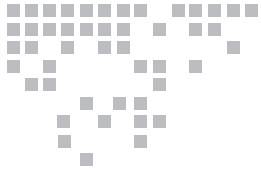
第一部分 Part 1

高效 Web 前端开发综述



Chapter 1 第 1 章

Web 前端开发概述

本章首先会简单介绍 Web 前端开发的历史由来，以及前端开发的概念及其范畴，让读者对前端开发有个整体的认识，同时还会对前端开发的一些错误认识进行更正。其次会介绍一下 Web 前端的现状，包括互联网整体环境、浏览器的发展、网站的设计和开发现状等，让读者认识到前端开发的现状和趋势，以便更好地提高自身的前端开发技能。

1.1 Web 前端开发的范畴

1.1.1 什么是 Web 前端开发

什么是 Web 前端开发？也许有人会说 Web 前端开发就是网页设计，就是美工，也有人会说 Web 前端的工作就是使用工具拖曳生成各种界面，然后导出为网页。其实这些都是对 Web 前端的误解，或者说还是停留在过去对前端开发的认识上。记得某个求职节目中，某互联网公司负责人对前来求职的前端工程师说他们理解的 Web 前端是需要用 Java 的，是需要用这种语言编写的。一个互联网公司的负责人竟然对自己公司使用的核心技术概念知之甚少，这实在是让人匪夷所思，更是闹了不小的笑话。此事件也激起了业内针对 Web 前端概念的大讨论，不过也从侧面说明 Web 前端这一概念还没有深入人心。的确，Web 前端兴起的时间还很短，这也就导致了很大一部分人不太了解这个新型的职业，所以我在这里再次解释一下 Web 前端这个词语的来龙去脉。

对于 Web 前端，业内公认的说法是从 2005 年开始兴起的。2005 年以前，因为 Web 网页主要以展示为主，内容基本都是静态的，所以客户端开发工作的目的就是让页面展现得更加整齐和漂亮，没有太多花哨的内容。网站的用户也只是以浏览为主，并不会有复杂的交互。正因为如此，一般的美工仅依靠 Photoshop 和 Dreamweaver 等工具就可以制作出外观漂亮的静态网页。2005 年之后，互联网进入 Web 2.0 时代。Web 2.0 更注重用户的交互作用，用户不再只是读者，同时也是作者。用户不再是仅仅阅读静态的网页，同时也为网站贡献内容。随着这一概念的发展，人们开始重新审视网站的设计，制作的网页慢慢变得生动起来，页面也有了大量的交互，不再是简单地展示静态的文字和图片。Google Gmail 的发布，使得 AJAX 技术大红大紫，这也把对 Web 2.0 的概念的认识推上了一个新的高度。AJAX 无刷新技术极大地增强了网页的用户体验，使得用户操作页面更流畅，操作体验更接近于本地应用。此外，搜索引擎的普及使得网站的搜索引擎优化受到了重视。搜索引擎对网站的外观并不感兴趣，只识别网站的 HTML 代码，这就要求网站的设计者和开发者不仅要重视网站外在的用户体验，还要重视网站内在的代码质量。

随着网站功能的丰富、设计风格的发展以及网站代码质量的要求，网页端的开发也变得复杂起来，其代码量和逻辑复杂度都增加不少，同时还需要考虑网站的性能、浏览器兼容及网站安全性方面的问题。传统的网站开发者仅仅会使用网页制作工具已经不能够满足目前的需求了，此时的网站开发更接近于后端开发，需要有专门的软件开发工程师来做网站开发相关的工作，于是原来的网页制作这一职业就演变成了 Web 前端开发。从职责上讲，Web 前端开发要涉及网站开发的方方面面，从前端 UI 到和后端的数据交互都属于前端开发的范畴。因此，Web 前端开发是兼具艺术气息和逻辑思维的综合体，既要考虑页面的美感和操作体验，又要关注前端代码的质量。

1.1.2　Web 前端开发需要具备的技能

由于 Web 前端技术兴起的时间不长，因此它还没有明确的界限定义，不同的 Web 项目中可能要求的 Web 前端开发技术会有所不同。例如，某些项目可能需要前端开发人员了解一些后端技术，这样前端开发人员才可以更好地与后端开发人员配合，如在页面上留下一些后端需要调用的“钩子”等，而某些项目可能需要前端开发人员懂一些 UI 设计、Photoshop 工具的使用方法等，以便于和 UI 设计师沟通和配合。虽然 Web 前端开发的范畴广泛，并且界限模糊，但是以下 7 点是 Web 前端开发必备的技能。

1. 页面标记（HTML）

由于页面 HTML 代码结构基本固定，HTML 的标签数量也不多，因此，从学习的难易程度来说，HTML 应该是前端技术中非常容易学习的技术。即使是一个新手，也能在较短的时间里学会编写一个结构良好的页面。虽然说入门容易，但是要编写语义良好、简洁整齐的 HTML 代码则需要大量的实践才能掌握。HTML 是页面的基本结构组成部分，是网站的基础，臃肿混乱的 HTML 代码不但会影响其本身的展现，而且与其对应的 CSS 与 JavaScript 代码也会变得难以编写和维护。

2. 页面样式

CSS 是 Cascading Style Sheet（层叠样式表）的简称。在标准页面设计中，因为 CSS 负责网页内容的表现，所以 CSS 也是前端开发需要掌握的核心内容之一。丰富的 CSS 样式能让平淡的 HTML 展现出绚丽的效果，使得页面更为友好。好的样式可以让用户在页面上停留的时间更久一些，也可以帮助用户更好地阅读网站内容，同时，还可以让用户在不同浏览器上有着相同的体验。CSS 和 HTML 代码一样，没有复杂的逻辑，上手也比较容易，其主要的难点在于如何合理地利用 CSS 的组合和继承特性来编写简洁、可维护性好的 CSS 代码。以上这两项基本技能是前端 UI 开发必备的技能。

3. 前端编程

前端编程技能主要是指 JavaScript 编程。JavaScript 是一种基于对象和事件驱动的客户端脚本语言，是页面实时动态交互的技术基础。相较于 HTML 和 CSS，编写 JavaScript 代码更能让前端开发人员找到后端程序员的感觉。JavaScript 是非常灵活的脚本语言，包含了高阶函数、动态类型以及灵活的对象模型等强大的语言特性，当然，JavaScript 的灵活性也可能导致代码不易维护。此外，浏览器的兼容性也增加了 JavaScript 编码的难度。同一个功能，可能在不同的浏览器中有不同的实现。例如，在 IE 浏览器中，事件绑定使用的是 attachEvent() 方法，但其他浏览器则使用的是 addEventListener() 方法。开发人员在熟悉 JavaScript 基本语法和基本的编码规范之外，还应该了解并解决在不同浏览器中的 JavaScript 的兼容性问题。另外，作为前端开发工程师，必定会涉及后端的编程，一些原因是目前流行的 Web 编程方式会有部分后端代码存在于前端页面中，和前端的 HTML、JavaScript 等混合在一起，如 PHP、JSP、ASP.NET 等，因此，前端开发工程师也有必要了解一些后端编程技术。

4. 跨平台，跨浏览器

前端代码本来不存在跨平台方面的问题，但是随着移动 Web 平台的兴起，跨

平台的问题就逐渐显现出来了。移动设备（如智能手机和平板电脑）在近几年发展迅猛，用户通过移动设备访问 Web 站点的比率也是逐年增高。如何在众多移动平台、众多屏幕尺寸上展现友好的 Web 站点成为一项前端技能。不过，目前跨浏览器没有像几年前表现得那么突出了，这主要是因为 IE 6、IE 7 浏览器的占有率下降和众多浏览器对标准的重视，另外，目前流行的前端框架已经很好地解决了浏览器的兼容问题。尽管如此，但是还需要熟悉常见的浏览器兼容方法，主要包括：IE 7、IE 8 的兼容，HTML5 中新特性的兼容等。

5. 前端框架

各种前端框架的出现，在很大程度上降低了前端开发的难度。框架统一了编码的方式，封装了浏览器兼容问题并添加大量的扩展功能。如今的 Web 项目中前端框架应用非常广泛，在开源社区 GitHub 上排名靠前的开源框架也是以前端框架居多。优秀的前端框架可以在很大程度上缩短项目开发的周期，尤其是 jQuery，几乎成为 Web 项目默认的前端框架。但是，前端框架的接口众多，各种框架的使用方式和编码方式也不尽相同，作为前端开发工程师，需要熟悉一些常用框架的使用方法，并且要了解如何编写常用框架的扩展插件，如 jQuery、YUI、Ext JS 等。

6. 调试工具

对于前端代码，在调试过程中需要查看页面的 HTML 结构变化、CSS 渲染效果、JavaScript 代码的执行情况以及 HTTP 请求和返回的数据，并且要了解网站各个部分的性能等，甚至需要动态更改 HTML、CSS 代码来查看预期的效果，模拟发起 HTTP 请求来查看后端返回的数据。主流浏览器都会有对应的浏览器插件来辅助完成这些工作，如 IE 中的 IE Dev Toolbar、Chrome 中的 Developer Tools、Firefox 中的 Firebug 等，此外还有 HTTP 请求监控和模拟工具，如 Fiddler 等。开发工程师需要熟练使用这些工具来辅助完成前端代码的调试。

7. 沟通能力

沟通是开发人员必备的一项基本技能，尤其是对于前端开发工程师来说。Web 前端开发介于 UI 和后端逻辑开发之间，因此，Web 前端工程师在开发过程中必定会和 UI 设计师及后端工程师合作：前端工程师需要和 UI 设计师沟通，确定效果是否可以实现以及实现的代价，并对 UI 设计提出建议；还需要和后端工程师沟通，确定前后端交互的接口以及传输的数据实体的结构等，良好的沟通会让这些过程变得轻松许多。

1.2 Web 前端开发现状

前端开发虽然起步时间晚，但是发展势头迅猛，在各种新技术、新标准的推动下，各大互联网公司也开始重视 Web 产品的前端重构与开发，如淘宝、腾讯、新浪、百度、搜狐等都对自己的网站进行了重构并同时使用了 HTML5 中的新特性。现在开发的 Web 新产品的页面交互也越来越丰富，视觉效果也越来越绚丽。互联网公司除了重视前端开发之外，还在积极推动互联网技术的发展，2001 ～ 2012 年，腾讯、百度及奇虎 360 公司先后加入了万维网联盟（W3C），共同参与互联网技术标准的研究和制定，促进了国内互联网产业的发展。

在 Web 前端发展的过程中，浏览器的发展起着至关重要的作用。浏览器厂商在这场 Web 浪潮中表现突出，具体表现在浏览器的版本升级周期短、对前端标准的支持也越来越好、浏览器的性能也有大幅的提升等。IE 浏览器也开始发力，最新发布的 IE 10 浏览器无论是标准的支持还是性能都表现突出。浏览器的快速发展使得低端的浏览器如 IE 6、IE 7 等快速退出市场。

下面看一下全球范围内浏览器的占有率情况。根据 StatCounter 的统计，2012 年 10 月份，全球浏览器占有率前 5 位的分别是 Chrome、IE、Firefox、Safari 及 Opera，占有率分别为 34.77%、32.08%、22.32%、7.81% 和 1.63%。按照浏览器的版本统计，在低端浏览器中，IE 8 占有率为 12%，IE 7 的占有率仅为 0.88%，IE 6 的占有率则更低。

此外，移动设备上的浏览器在整个浏览器占有率中上升速度很快。根据 StatCounter 的统计，Safari iPad 在浏览器的占有率已经达到了 2.76%。目前移动设备的普及率极高，用户已经逐渐习惯于通过移动设备来浏览网页。主流的移动平台主要为 Android 和 iOS 系统，对应的浏览器都是以 WebKit 为核心的，性能和对标准的支持都不错。

以上介绍的是浏览器在全球范围内的占有率情况。从上面的统计可以看出，在全球范围内，低端浏览器的占有率已经很低了，但国内浏览器占有率的状况则比较复杂，低端浏览器如 IE 8、IE 7 和 IE 6 等还占有不小的份额。根据 StatCounter 的统计，2012 年 10 月份，国内浏览器占有率前 5 位分别为 IE 8、IE 9、IE 6、搜狗浏览器和 360 安全浏览器，占有率分别为 48.7%、14.09%、8.29%、5.74% 和 2.72%。其中低端浏览器 IE 8、IE 7 和 IE 6 的总占有率大约为 58%，低端浏览器依然占有“半壁江山”，而高级浏览器如 Chrome、Safari、Firefox 等占有率则很低。值得称道的是，360 安全浏览器和搜狗浏览器这两款浏览器都是基于开源的浏览器内核，对标准支持较好，性能表现突出。这两款浏览器的占有率目前虽然不高，但发展势头迅猛，是推动老旧浏览器退出市场的中坚力量。

除各大互联网公司和浏览器厂商的积极推动外，Web 前端开源社区的参与热

情也极为高涨。在著名的网络代码托管系统 GitHub 上，开源的项目仍然是以前端相关项目居多，按语言排名，前 3 位分别为 JavaScript、Ruby 和 Python；按项目排名，靠前的大部分也是前端项目，如 Prototype、Bootstrap、jQuery 及相关插件等，具体内容可参考 GitHub 官方网站。国内的一些前端社区则发展较为缓慢，多是以推广 HTML5 为主，靠一些酷炫的效果来吸引眼球，反而关注前端基础的较少。国内 Web 互联网公司（如淘宝、百度、腾讯等）也有部分优秀的开源前端框架，这些公司的前端技术分享氛围不错。整体来讲，国内前端技术发展还是比较迅速的，也不乏顶级的前端技术人员，但普遍缺乏的是开放的心态和与国内外同行交流的能力，国内前端技术的发展任重道远。

Web 前端整体技术的发展和前端工程师个人的能力是相辅相成的。目前，前端工程师很多是“半道出家”，一部分是从页面 UI 开发转行为 Web 前端开发的，一部分则是由后端工程师转行而来，所以前端工程师普遍自学成才，并没有受过足够的专业训练，也缺乏实际的项目经验。

除了以上这些 Web 前端开发的外在环境之外，Web 前端在技术方面也存在着大量的挑战，大量旧的网站需要重构来提高网站用户体验和性能等。这些网站的前端代码普遍存在的问题有：代码组织混乱，CSS 代码和 JavaScript 代码混合在 HTML 代码中；代码的格式问题突出，不够整洁；页面布局随意，HTML 代码不符合标准；网站整体性能差，还没有意识到要去应用诸如缓存、动态加载、脚本压缩、图片压缩等提高性能的技术。

此种状况并非是个案，查看目前访问量颇高的某网站的首页源代码，可以很直观地看出其中不规范的内容：页面没有 body 的闭合标签，页面代码缩进随意，页面中大量使用内联样式并且页面中还在使用诸如 <marquee>、<font> 等标准不推荐的标签。以下是其中的代码片段：

```
    <marquee  width="260"  ONMOUSEOUT="this.start()"
ONMOUSEOVER="this. stop()" scrollamount=3 scrolldelay=100 >
                              <font color=blue style="font-size:14px;">[redacted]
</font>    
<a  href="http:[redacted]"  target="_blank"><font  color=blue
style="font-size:14px;">[redacted]
</font></a>                                  </marquee>
```

在这段代码中，<marquee> 和 <font> 标签已经不再被 W3C 制定的标准推荐使用，<marquee> 所呈现的效果完全可以通过 JavaScript 代码实现，并且实现的效果可以更好。<font> 标签完全可以通过 CSS 样式代码实现。

此外，此站点首页 HTML 代码中包含大量不必要的标签和多余的样式设置，代码的可读性较差。再来看一段代码：

```
    <!—第一段代码片段 -->
    <div id="slideshow">
       <div class="current"><img src="./images/pic011.gif" width="189"
height="176" />
    </div>
      <div><img src="./images/pic012.gif" width="188" height="176" /></div>
      <div><img src="./images/pic013.gif" width="188" height="176" /></div>
    </div>
    <!—第二段代码片段 -->
    <div><br/>
        <a href="./gljd/gywm/">关于我们</a>  |  <a href="./gljd/wzls/">网站
声明</a>
        <br />
         版权所有 &copy; [illegible] <br /><a href=
"[illegible]" target="_blank">[illegible]</a>
         </div>
```

在上面的代码段中，第一段代码中的 <img> 标签外的 标签是可以省略的，<img> 标签内的 width 和 height 属性可以统一通过 CSS 代码设置，<img> 标签内需要添加必要的 alt 属性来说明图片的信息。在第二段代码中，依靠
 标签来增大距离是不合语义的做法，应该通过 CSS 样式来设置。因此，以上两段 HTML 代码至少可以简化为：

```
    <!—第一段代码片段 -->
    <div id="slideshow">
       <img src="./images/pic011.gif" class="current" alt="img info" />
       <img src="./images/pic012.gif" alt="img info" />
       <img src="./images/pic013.gif" alt="img info" />
    </div>
    <!—第二段代码片段 -->
    <div>
       <a href="./gljd/gywm/">关于我们</a>  |  <a href="./gljd/wzls/">网站声
明</a>
       版权所有 &copy; [illegible]
       <a href="[illegible]" target="_blank">[illegible]</a>
    </div>
```

另外，站点中的 CSS 样式文件和 JavaScript 脚本文件，没有任何文件被压缩和合并。推荐的做法是分别压缩合并样式文件和脚本文件，在缩小文件大小的同时也减少了文件的 HTTP 请求次数，提高了性能。

综合以上的这些信息，Web 前端开发的现状可以概括为：前端技术发展迅速，但起步较晚，基础薄弱；前端工程师热情高涨，但缺乏足够技能培训，对代码规范重视不足，对一些基础原理的理解不够深刻。

1.3　规范的 Web 前端代码：更易维护、更高性能和更安全

规范的代码，这是所有软件开发中对代码的基本要求，前端开发也是一样的，要求编写规范的 HTML、CSS 和 JavaScript 代码。

什么样的前端代码才能称得上规范的代码？探讨这个问题之前，首先需要强调的是规范不是标准，不是放之四海而皆准的，不同的项目中的代码规范是有可能有差异的，比如命名，有些项目规定 HTML 标签的 id 必须要以控件的缩写名作为前缀，如按钮的 id 名以“btn”作为前缀，有些只是规定命名有意义就可以。再比如有些项目规定 JavaScript 代码语句结尾必须添加分号，但是有些项目并不要求，大名鼎鼎的 Bootstrap 源代码中并没有给 JavaScript 语句结尾添加分号。因此，规范的作用只是让同一个团队代码风格统一，减少协作时的复杂性，确保后续的维护和修改方便。不同团队中遵循的规范有可能存在部分差异，但是在同一个团队中规范必须是统一的，团队中的成员应该严格遵循。澄清了规范的作用后，接着上面的问题来探讨什么才是规范的前端代码。

Web 前端的代码规范主要针对的是 HTML、CSS 和 JavaScript 代码。尽管前端代码规范在不同场合会有差异，但是规范的前端代码应该具有如下特征。

（1）符合标准

所谓的标准指的是 W3C 制定的 Web 标准，这是从百度百科找到的关于 W3C 的介绍：

万维网联盟（World Wide Web Consortium，W3C），又称 W3C 理事会。1994 年 10 月在麻省理工学院计算机科学实验室成立。建立者是万维网的发明者蒂姆·伯纳斯 – 李。

万维网联盟是国际著名的标准化组织。1994 年成立后，至今已发布近百项相关万维网的标准，对万维网发展做出了杰出的贡献。目前，万维网联盟拥有来自全世界 40 个国家或地区的 400 多个会员组织，已在全世界 16 个地区设立了办事处。2006 年 4 月 28 日，万维网联盟在中国内地设立首个办事处。

W3C 制定的标准包括使用语言的规范、开发中使用的原则和解释引擎行为等，主要由 3 个部分组成：结构（Structure）标准、表现（Presentation）标准和行为（Behavior）标准。结构标准包括 XML 标准、XHTML 标准和 HTML 标准，目前使用的标准版是 HTML 4.01 标准，HTML5 是 HTML 和 XHTML 的最新标准，还没有发布最终版；表现标准主要指的是 CSS 样式标准，目前使用的标准版是 CSS2，CSS3 尚未发布最终版；行为标准主要包括 ECMAScript 标准和 DOM 标准。ECMAScript 是 ECMA（European Computer Manufacturers Association）制定的标准脚本语言（JavAScript）。DOM 是文档对象模型（Document Object Model）的缩写，是一个中立于语言的应用程序接口，允许程序访问并更改页面的内容、结构

和样式。目前推荐遵循的 ECMAScript 标准是 ECMAScript 262 第 5 版，DOM 标准是 DOM 级别 2，DOM 级别 3 目前还没有发布正式版本。W3C 推荐的这些标准受到了各浏览器厂商和 IT 互联网公司的欢迎，前端代码遵循标准则可以保证网页在不同浏览器上正常展现。

（2）格式规范统一

前端代码的格式主要包括命名、代码缩进、空格和空行的使用以及代码注释。命名主要有 HTML 元素的 id 和 class 名，JavaScript 中函数和变量的命名；HTML、CSS 和 JavaScript 代码中都需要通过代码的缩进来体现代码的层次关系；空格和空行主要用在 CSS 和 JavaScript 代码中，用来提高代码可读性，如操作符前后添加空格、不同代码段逻辑之间添加空行等；CSS 和 JavaScript 代码中都需要添加必要的注释来解释说明代码文件及代码段，HTML 代码中使用注释的情况较少。和其他代码格式规范一样，前端代码格式规范也是为了提高代码的可读性和可维护性。

（3）高性能

前端性能体现在浏览器的响应速度上，包括网页的加载速度和页面的交互响应速度。网页加载所占用的时间包括后端处理请求的时间、代码文件从服务器端传输的时间、HTML 和 CSS 的组合展现的时间以及 JavaScript 加载和运行的时间。除了第一条，其余都和前端代码有直接的关系，减少文件传输时间的最直接的方式就是减小文件的大小，越少的代码文件相对传输就会更快；简洁和符合标准的 HTML 和 CSS 代码能减少浏览器解析的时间，加快浏览器渲染过程；页面中请求数量越少相对页面加载时间也会越快；在 JavaScript 代码中选择性能更好的实现方案，如延迟加载、动态加载等技术，会让页面加载更快和交互更流畅。规范的前端代码应该针对这些方面来编写高性能的前端代码，提高用户的前端体验。

（4）高安全性

网站的安全有时很难引起一些互联网公司的足够重视，他们更看重的是站点的用户体验、性能等这些更直观的方面。2011 年，多个网站用户信息泄露风波震惊整个互联网界，网站安全也再次引起业内的重视。从技术上讲，网站的安全瓶颈主要在后端，但是随着前端技术的发展，富客户端应用越来越多，前端安全问题也随之增多，如跨站点攻击、Cookie 劫持等。这些攻击通过设置 JavaScript 变量、HTML 标签的值和属性、CSS 属性值等方式伪造恶意代码来达到攻击的目的，因此，规范的前端代码至少要针对这些方面做必要的安全校验和编码，提高代码的安全性。

前端代码如果能遵守如上的几个规范点，则基本上能称为高质量的代码。需要强调的是，酷炫的技术只是“浮云”，良好的编码习惯和意识才是真功夫。本书后续的章节将会针对如上的代码规范点展开详细讨论。

第 2 章 Chapter 2

高效 Web 前端开发

本章首先将概述 Web 前端开发中的相关最佳实践，如前端代码文件组织、前端代码重构、前端框架的选择，以及前端开发过程中实用的开发辅助工具等，帮助读者提高前端开发的效率。好的开发方式在项目中会起到事半功倍的效果，并且可确保开发过程中的代码结构清晰，易维护。本章然后会介绍前端代码的基本命名规范和格式规范，良好的命名规范和规整的格式让代码看起来干净整洁，也体现了开发者良好的职业素养，应该说命名规范、整齐的格式不仅是开发过程中的一种约定，而且是程序员之间良好沟通的桥梁。

2.1 前端代码的结构组织和文件的命名

代码的组织和代码文件的命名并没有最优的形式，但是无论什么代码，它们所遵循的原则是相同的，即在同一个项目中代码的组织结构一定要清晰，同类型的代码文件或者相同模块的代码文件尽量归类到相同的文件夹中，文件的命名规则须统一并且命名要有意义。

（1）代码文件组织结构

前端代码文件主要包含 JavaScript、CSS、HTML 等文件，以及这些代码文件相关的图片、Flash、音视频等资源文件。如何合理地组织这些文件是项目成败的关键因素之一，对于该文件，既要考虑结构清晰、一目了然，还要考虑代码的复用。基于这样的原则，惯用的做法是同类文件放在一起，并按模块划分文件结构。

如下是一种常用的前端文件的组织结构：

- ❑ js（放置 JavaScript 代码）
  - lib（放置框架 JavaScript 文件）
  - custom.js
- ❑ css（放置 CSS 样式代码）
  - lib（放置框架 CSS 文件）
  - images（放置用于样式中的背景图）
  - reset.css（统一元素默认样式的样式文件）
  - custom.css（业务相关样式文件）
- ❑ resource（放置页面图片文件以及其他类型资源文件）
- ❑ index.html

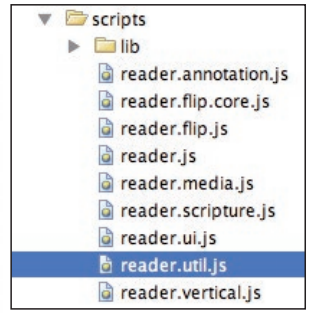


图 2-1 代码文件组织结构示例

代码文件整体按照文件类型的不同归类，同一类型的代码文件则需要按照具体的业务模块来划分，切忌把多个模块的代码编写到同一个文件中。划分的粒度以最大化代码复用为标准，这样做的优点是易于维护和管理。不同模块的代码放置到不同的文件中也更有利于多人协作开发。图 2-1 是实际项目中的一个示例。

如果每种分类下面的文件过多，则可以根据对应的模块来归类到不同的模块文件夹中。例如，某个项目业务模块很多，导致前端 JavaScript 脚本文件数量过多，如果这些文件放置到同一个文件夹中，将会增加维护的困难。如果按照模块的不同新建不同的文件夹，并把同一模块相关的代码文件放置在对应的模块文件夹中，则代码文件的结构更加清晰。

细心的读者可能会发现代码文件的细化会使代码文件的数量增多，相应地进行网站加载时请求的数量也会增加，这样会导致网页整体的加载时间变长。的确，加载文件数量增加会影响页面加载的速度，所以，发布时需要合并相应的代码文件。值得庆幸的是，有相关的工具或插件可以弥补这一不足，比较著名的有雅虎（Yahoo）公司开发的 YUI Compressor 和微软公司开发的 Web Optimization。YUI Compressor 是基于 Java 平台开发的，在 Java 平台下应用较多，而 Web Optimization 则是基于 .NET 平台开发，所以在 ASP.NET 开发的项目中广泛应用。

（2）代码文件的命名

代码文件命名的原则主要是名称需要表明文件对应的模块内容、对应的版本号和文件的格式等，如 jQuery 的文件命名为 jquery-1.8.2.min.js，其中，jquery 表明文件的内容，1.8.2 表明文件的版本号，min 表明此文件为文件的压缩格式版本。如果文件为子模块文件，则在文件名中用点号或短横线来表明父子关系，如 Bootstrap

框架中响应式设计模块对应的 CSS 文件的命名为：bootstrap-responsive.css。

2.2　前端代码重构

代码重构是业内经常讨论的一个热门话题。重构指的是在不改变代码外部行为的情况下进行源代码修改，重构之前需要考虑的是重构后如何才能保证外部行为不改变。对于后端代码来说，可以通过大量的自动化测试来确保重构后的代码逻辑，可对于普遍缺乏自动化测试的前端代码来说，重构之前一定要考虑再三才能下手。

我曾经有一次不算太成功的前端重构经历，所幸的是没有导致太大的问题，但教训是惨痛的。此次重构的项目本身没有足够的自动化测试，尤其是针对前端的自动化测试，其实在重构之前也预想到了重构的风险。先来介绍重构之前项目存在的问题：项目的前端代码没有统一的规范，不同人员开发的模块对应的代码风格完全不同，即使有相同的逻辑也是由完全不同的代码实现的；而且，不同模块的代码写到了相同的代码文件中，使得文件的代码量很大；项目的功能较多，前端代码量大，代码的文件也多，又由于需求变更频繁，因此代码中的无用代码量较大；网站首页加载时间过长，平均加载时间超过 5 秒；部分动态模块的加载会导致页面“假死”。总体来说，项目代码存在的问题就是维护性差和性能差，如果要持续地添加新功能，就有必要重构现有的代码。

鉴于项目代码的实际情况，并且考虑到缺乏足够的自动化测试，因此，我权衡再权衡，最终决定先尝试对前端的部分代码进行重构。重构的过程主要分为如下几个步骤。

1）删除无用代码，精简代码。无用的代码主要集中在 CSS 和 JavaScript 文件中，包括已经不起作用的 CSS 样式和废弃的 JavaScript 函数。这个重构过程是比较麻烦的，因为修改的过程需要小心翼翼，不停地查看函数和变量在上下文的调用关系，以免错删代码。总体来说，这个阶段的重构还是利大于弊，虽然后续发现了一些由于误删而导致的 Bug，但是删除了废弃的代码，提高了后续的可维护性。

2）前端代码规范化。页面的 HTML 标签中还有大量的内联 CSS 样式，有些页面的 head 部分也有 CSS 样式，需要把这些 CSS 放到独立的文件中；调整代码的层次缩进格式，不同层级按照 4 个空格来缩进；更改标准已不推荐的标签，如 、<b> 等，改为由 CSS 样式控制；统一命名规则，这里主要涉及 HTML 中的 id 和 class 名称；另外，在 JavaScript 中集中定义局部变量，并把部分全局变量转变为局部变量，缩小变量的作用域。

3）整理基础类库。网站初期为了加快开发的进度而引入了多个框架，其中包括 Ext JS、jQuery 以及多个 jQuery 插件。Ext JS 和 jQuery 中重复的功能较多，项目中很多相同的实现使用的是不同类库中的功能，如 DOM 选择、AJAX 请求等。部分 jQuery 的 UI 插件的功能也和 Ext JS 中的功能重复。鉴于项目的状况，只能保守选择同时保留 Ext JS 和 jQuery。重构的内容是统一 UI 插件的使用、统一基础方法的使用。

4）前端代码模块化。按照模块归类 CSS 代码，放到模块对应的单独 CSS 文件中；按照模块分离 JavaScript 代码，按照模块定义不同的命名空间；将 JavaScript 代码中的公共方法归类到独立的共通文件中，同时引入面向对象的思想来重构 JavaScript 代码，进一步明确公有接口和私有接口。

5）提高页面加载性能。将部分不影响首页展示的 JavaScript 文件延迟到页面加载后加载；删除页面中初始隐藏的区域，改为通过 JavaScript 按需动态生成；页面中的部分图片延迟加载；调整 CSS 和 JavaScript 文件的引用顺序，即 CSS 在前 JavaScript 在后；给静态文件设置缓存；使用 CSS Sprite，合并首页背景图；合并和压缩发布后的 CSS 和 JavaScript 代码文件。

从上面的重构内容可以看到，这次重构的目的是明确的，即提高前端代码的可维护性和性能。从中也暴露了重构过程中的一些问题：最主要的问题还是低估了重构的风险，导致后续发现了不少新 Bug ；没有足够的自动化测试，而冒然去修改代码，很难保证最终修改代码的正确性；其次是对于性能优化，事先没有量化，没有做任何性能方面的监控，没有抓住性能主要的瓶颈，所以性能部分的重构就显得比较盲目。

有了这次的重构实践经历，再加上学习业内的一些经验，我归纳了如下的前端重构最佳实践。

- ❑ 重构前一定要预估风险，如果没有足够的自动化测试，最好是先完善自动化测试代码。
- ❑ 重构的目的和范围要明确，切忌盲目修改。前端代码的重构目的主要是提高代码的可维护性、可读性和性能。
- ❑ 最好是先易后难，循序渐进。首先修改诸如命名、格式等不涉及具体逻辑的内容，然后考虑模块化和性能提升等与具体逻辑相关的内容。
- ❑ 重构过程中要持续测试，在多个浏览器中测试，确保重构的部分功能正确。切忌在大量重构后再进行测试，因为大量重构后基本很难记得重构的逻辑，也就有可能遗漏部分测试用例。
- ❑ 如果是性能提升，要事先检测网站的整体性能并量化，找出性能瓶颈。重构过程中要持续监控性能，并对比性能提升的效果。

2.3　合理使用前端框架

JavaScript 本身是一种很强大的脚本语言，但是 JavaScript 固有的灵活性和由于使用多浏览器而产生的复杂性，在使用时并不能得心应手。此种状况下，JavaScript 框架的重要性就显现出来了。JavaScript 框架是 JavaScript 代码的工具集和函数集，一般包括 DOM 元素操作、DOM 事件的封装、AJAX 操作、UI 控件封装以及一些功能算法的扩展等，如 string、array 等的功能扩展。好的 JavaScript 框架已经经过了大量的功能测试、性能测试，也经过了各主流浏览器的兼容测试，在项目中应用前端框架，不仅加快了项目的开发进度，同时还避免了出现各种浏览器的兼容问题。

前端代码天生的开源优势使得前端开源框架异常丰富，全球最大的社交编程及代码托管网站之一 GitHub 中收录的开源项目也以 JavaScript 项目居多，但是正因为前端框架多种多样，各个框架的功能侧重点也不同，如何选取合适项目的框架，就成为前端开发人员的一项技能。

给项目选择一个前端框架，至少需要考虑如下几个因素。

1）项目的需求。这是选择前端框架的最主要因素之一。选择框架之前，先要调查项目的基本需求，调查项目是否有 AJAX 操作？是否需要模板化？数据传输的格式是 JSON 还是 XML 等？项目需要支持的浏览器有哪些？项目是移动应用还是普通的网站？项目中需要哪些主要的 UI 模块？常用的 UI 模块有模态窗口、滑块控件、进度条、提示框、分割框、幻灯显示以及自动填充等。

2）项目的特点。项目本身的特点也是选取框架的一个重要因素，这些特点包括项目开发团队的前端技术能力、项目的开发周期，以及是否是长期维护的产品等。使用框架是有学习成本的，需要开发团队先学习框架的使用方法，如果某个框架对开发团队来说是不易上手和使用的，那么就可能需要考虑其他相对容易上手的框架。项目开发周期的长短也会影响框架的选择，开发周期短则优先考虑一些容易上手、能快速开发的框架，或者选择使用多个框架。如果项目是需要长期升级维护的，则选择框架时需要考虑框架本身的升级、框架本身的扩展性等。

3）框架的特点。分析项目的状况后，则可以开始选择框架了。从框架的角度出发，需要调查框架是否满足需求？框架的性能是否可以接受？框架是否持续开发中？文档和例子是否充足？是否有技术支持？是否有版权问题？如果收费，价格是多少？有些框架属于轻量级的，功能集中但文件小，加载快；有些框架功能多，面面俱到，但文件相对较大，加载也慢一些；有些框架容易上手，但是运行速度慢，会影响页面整体的渲染，所有这些性能因素要尽早分析清楚。历史悠久的框架已经经过了很多项目实际的检验，这些框架的质量高，Bug 会少一些。如果框架有官方正式的 API 文档、对应的代码示例以及技术支持渠道，则会帮助

使用者快速地掌握框架的使用方法。

考量前端框架主要就是如上介绍的这些原则。需要强调的是，框架虽然非常有用，但是不要过分依赖，不能只是为了框架中的某一个功能就引入整个框架，这样会导致前端代码过于庞大，影响站点的性能。鉴于框架在项目中的重要性，一般在大型的 Web 项目中需要有专门的个人或团队来管理前端框架及其扩展。

了解前端框架的实际使用情况也有助于我们选择合适的框架。在 2012 年 10 月份，国外知名的前端工程师 PPK（Peter-Paul Koch）曾经发起过一个有关前端框架使用的调查，调查结果显示在过去一年中 95% 的开发者使用过 JavaScript 框架，91% 的调查者表示在超过一半的项目中使用过框架。对单独的框架，PPK 提出了两个问题：

1）在过去一年内你使用过“xx 框架”吗？

2）你是否在多于一半的项目中使用了“xx 框架”？

调查的结果还显示，使用率较高的完备框架有 jQuery、YUI、Prototype、Ext JS、MooTools 等。尤其是 jQuery，有超过 91% 的调查者在过去的一年中使用过 jQuery，称其为 Web 项目标配的前端框架一点也不为过。此外，使用率较高的轻量级框架有 Modernizr、Underscore.js、Backbone.js、Raphael.js 等。

前端工程师有必要了解和学习这些常用的框架，理解这些框架的功能、性能、设计原理等，在实际的项目中选择合适的框架就会得心应手，不至于盲目选择。

2.4 多浏览器测试：多测试，早测试

多浏览器的兼容一直是前端开发中比较头疼的一个问题，尤其是 IE 6 和 IE 7 浏览器盛行的阶段。以目前的趋势来看，随着 IE 6 和 IE 7 等浏览器逐渐退出市场以及各新版本浏览器对标准的支持会越来越好，浏览器的兼容问题也随之慢慢变小。此外，各种成熟的前端框架已经兼容了几乎所有浏览器，如果使用了这些框架则会减少浏览器的兼容问题。对 Web 应用开发者来说，这是一件值得庆幸的事，我们再也不用经常解决各种浏览器兼容的问题了，可以把更多的精力放在具体的业务逻辑上。虽然浏览器兼容问题已不是前端开发中的主要问题，但是还是会有一些兼容问题需要解决。目前多浏览器的兼容问题主要集中在如下几个方面。

1. IE 7、IE 8 的兼容

IE 6 已经基本退出了历史舞台，在开发过程中可以不再考虑兼容 IE 6 的问题。针对 IE 6 用户，仅显示提示升级信息即可。IE 7 和 IE 8 的占有率目前还很高，但其表现和标准的浏览器有一定的差距。这里并不打算介绍 IE 系列浏览器的具体兼容问题，但需要强调的是，IE 7 和 IE 8 的兼容问题是目前浏览器兼容的

主要关注点。

2. HTML5 的兼容

最近几年，HTML5 发展迅速，其大量的新特性也广泛应用。尽管各浏览器也加快了对新标准支持的脚步，但支持的步调并不一致，总有一些特性并不能被所有的浏览器支持。此外，使用 HTML5 新特性的同时也必定要考虑旧浏览器兼容的问题，如果要使用 HTML5 中的新特性，就必须明确浏览器支持的范围，考虑对于不支持的浏览器如何做到平稳降级。

浏览器兼容问题直接影响到方案的选择问题，应该在开发阶段就进行浏览器的兼容性测试，不仅要在多个浏览器中测试，甚至还包括在相同浏览器的不同版本中测试，只有这样才能确保设计的效果可完美地呈现在所访问的浏览器中。此外，为了把浏览器的兼容问题降到最小，在开发过程需要遵循以下一些准则。

（1）编写高质量、标准的 HTML

不标准或者无效的 HTML，可能在某个浏览器中可以正常显示，但是很难保证在所有的浏览器或者将来新版本的浏览器中正确显示。要让网页在所有浏览器中的显示效果相同，最有效的方法就是编写高质量和标准的 HTML 与 CSS 代码。为了确保代码的高质量和标准，可以使用一些标准验证工具，如 W3C 提供的 HTML 和 CSS 的标准验证工具以及检查 JavaScript 代码规范的 JSLint 等。

（2）明确支持的浏览器范围

多浏览器的支持一直是 Web 开发中的难点，浏览器的版本众多并且对标准的支持并不相同，很难在主流浏览器中做到网页的展示和交互都完全符合设计要求。因此，在 Web 项目开发之初就应该明确浏览器的支持范围，并在不支持的浏览器访问的时候引导用户下载和使用支持的浏览器。在支持的浏览器中也并非能做到所有浏览器中的展示效果和交互都满足设计要求。如下是浏览器支持的 4 个标准，可供在确认浏览器支持情况时参考。

- ❏ 外观和交互符合设计需求。
- ❏ 外观上有所差异，但不影响整体的效果。
- ❏ 外观和设计不符合，但不影响页面的交互。
- ❏ 完全不支持设计的功能。

在明确浏览器支持时，可按照此 4 类浏览器支持的标准制定浏览器支持情况列表，供开发和测试参考。

多浏览器测试是个耗费精力的工作，不太可能同时测试所有的浏览器以及浏览器的各种版本，在开发阶段只需要重点测试网站支持的浏览器就足够了。对于 Chrome、Safari、Firefox 等高级浏览器，它们的更新频繁，一般只需要支持最新版本即可。Chrome 和 Safari 浏览器有相同的 WebKit 浏览器内核，在开发过

程中可以重点测试其中一个浏览器。IE 系列浏览器各版本之间的差异较大，开发和测试过程中需要测试所有支持的 IE 浏览器版本。IE Tester 是个很好的 IE 兼容测试工具，可以模拟各个版本的 IE 浏览器，当然也可以简单地更改 IE 浏览器渲染模式来查看低版本 IE 浏览器中渲染的效果。另外，IE 9 浏览器自带的 IE Dev Toolbar 可以方便地切换各版本模式，如从项目开始就得明确要支持的浏览器范围，当有不支持的浏览器访问时要引导用户下载和使用支持的浏览器，如图 2-2 所示。

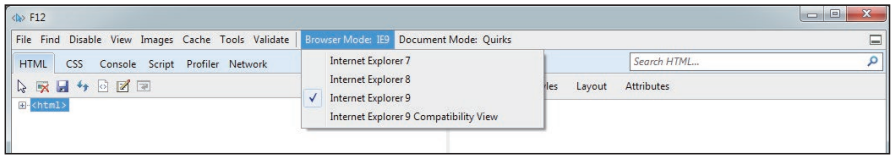


图 2-2 IE Dev Toolbar 工具中的浏览器版本切换菜单

（3）尽量避免出现浏览器兼容问题

浏览器兼容技能属于开发技巧，不属于开发技术。开发技巧在某些场合的确能起到事半功倍的效果，但带来的问题会是代码的难理解和难维护。例如，为了使 CSS 兼容 IE 7 而在 CSS 属性前面添加的星号（*），增加了代码量同时降低了可读性。因此，遇到兼容问题，首先考虑是否可以更改方案，使用没有兼容问题的代码等，其次再考虑如何兼容的问题。兼容 IE 系列的浏览器，建议使用 IE 特有的条件表达式，让兼容相关的代码独立，提高代码的可维护性。

（4）使用 HTML5 新特性时做好平稳降级

目前，在前端开发中对 HTML5 兼容，实属无奈之举。HTML5 的兼容包括：新标签的兼容、CSS3 的兼容，以及新增加的 API 的兼容。HTML5 中的新标签并不能被 IE 7 和 IE 8，以及其他旧浏览器所支持，当然可以通过一些技巧让新标签在这些浏览器中正常展现，如可以使用 html5shiv 框架[⊖]。CSS 本身的渲染特点使得在开发中可以直接使用 CSS3 中的新特性，如果浏览器不支持 CSS3 的新特性，则会直接忽略，不会产生语法错误。HTML5 新增加接口的使用过程中，必须添加条件判断。框架 Modernizr[⊜]已经很好地封装了对 HTML5 新特性的判断，推荐在实际项目中使用。在使用新特性时有必要了解特性在各浏览器中的支持状况，推荐使用“Can I use”[⊜]网站查询新特性的浏览器兼容情况，如查询浏览器对 CSS3 中的圆角支持情况（见图 2-3）。

⊖ https://github.com/aFarkas/html5shiv

⊜ http://modernizr.com

⊜ http://caniuse.com

图 2-3　CSS3 Border-radius 对应的浏览器支持

2.5　Web 前端代码开发和调试

2.5.1　Web 前端集成开发环境

很多集成开发环境（IDE）都集成了前端代码 IDE，如 Visual Studio、Eclipse 等，但在纯粹的前端开发中，这些 IDE 显得不够强大而且不够轻量。这里推荐两款强大的 IDE：Aptana Studio[⊖]和 WebStorm[⊜]。

Aptana Studio 是一个开源的 Web 开发工具，有非常强大的 JavaScript 编辑器和调试器（见图 2-4）。它的主要特性包括：

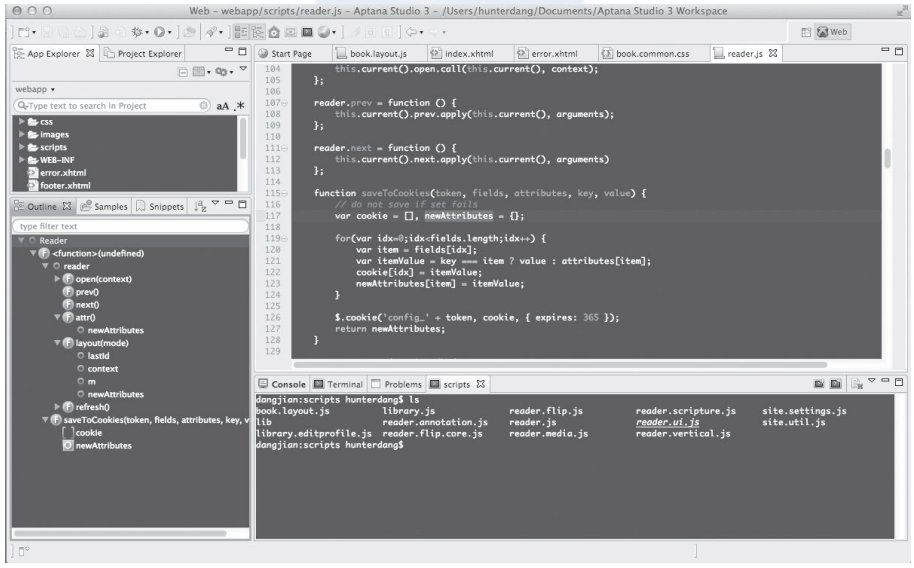


图 2-4　Aptana Studio 编辑器的界面

⊖ http://www.aptana.com/products/studio3

⊜ http://www.jetbrains.com/webstorm/

❑ JavaScript 函数、HTML 及 CSS 的 Code Assist 功能。

❑ 显示 JavaScript、HTML 及 CSS 代码的结构。

❑ 代码语法错误提示。

❑ 调试 JavaScript 代码。

❑ 支持流行的 AJAX 框架的智能提示：jQuery、Prototype、YUI、Ext JS 等。

WebStorm 是 JetBrains 专门为 Web 开发人员提供的一款商业的 JavaScript 开发工具（见图 2-5）。其功能和 Aptana Studio 类似，主要有智能代码提示、代码格式化、代码调试、代码结构展示、代码重构等。

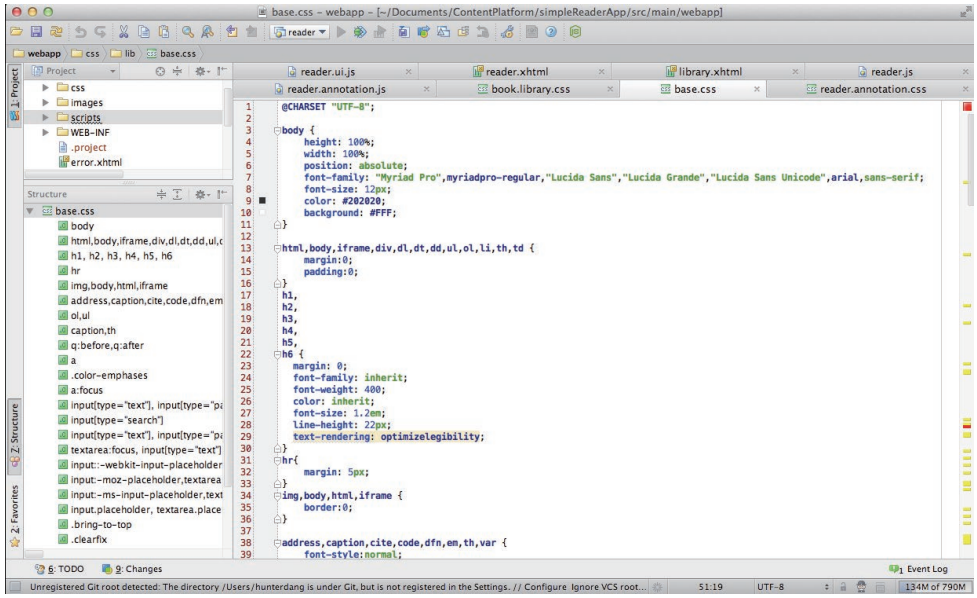


图 2-5 WebStorm 编辑器的界面

2.5.2 Web 前端代码调试

各主流浏览器都自带的有调试工具，如 IE 浏览器自带的 IE Dev Toolbar，Chrome 浏览器自带的 Developer Tools，Firefox 浏览器插件 Firebug，Safari 浏览器自带的 Develop 系列工具。这些工具的功能类似，都可以查看 HTML、CSS 代码，动态更改页面的 HTML 和 CSS 代码并实时查看修改效果，也可以调试 JavaScript 代码的执行。这些自带的工具基本可以满足前端的代码调试工作。

图 2-6 展示的是 Chrome 自带的 Developer Tools。

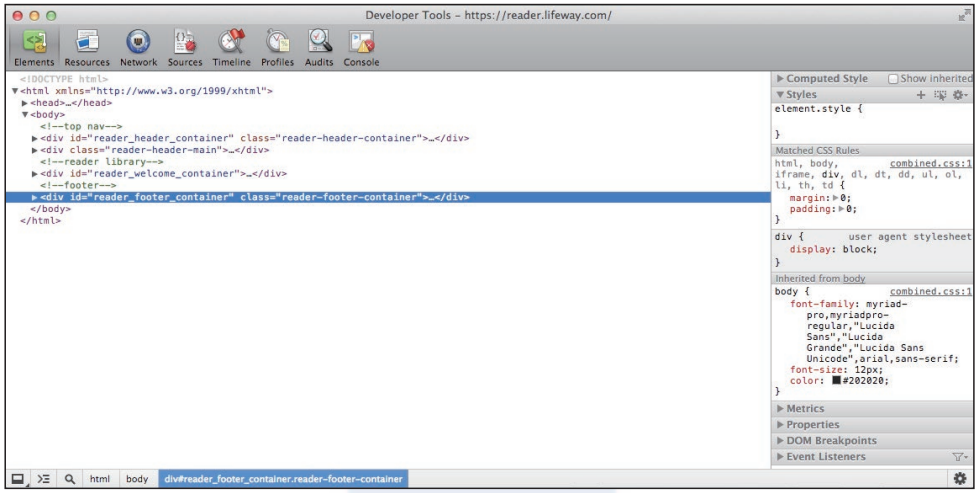


图 2-6　Chrome 浏览器中的 Developer Tools

2.5.3　Web 前端性能分析

常用的 Web 前端性能分析工具有 YSlow[⊖]、PageSpeed[⊜]及各浏览器自带的开发工具等。YSlow 是由雅虎公司开发的一款免费工具，主要的特性有：

- ❑ 基于不同的规则评定网站整体性能评分。
- ❑ 给出提高网页性能的建议。
- ❑ 统计页面加载的组件。
- ❑ 页面的统计信息视图。
- ❑ 相关性能分析的工具集，如 JSLint、Smush.it 等。

YSlow 可以以插件的形式安装在多个浏览器上，如 Firefox、Chrome、Opera、Safari 及一些移动浏览器等。

图 2-7 所示为在 Firefox 浏览器中 YSlow 作为 Firebug 的一部分。

YSlow 的性能分析基于 23 条规则，这 23 条规则是雅虎公司性能改进团队制定的 34 条提高 Web 性能“军规”中的一部分。YSlow 性能评分也是针对这 23 条规则逐项评分的。其官方网站上列出了这 23 条规则及具体说明，值得前端开发者仔细阅读和学习。

PageSpeed 出自于 Google 公司，也是一款免费的工具，提供了在线工具和浏览器插件两种方式。功能和 YSlow 类似，只是检查的规则稍有不同。

⊖ http://yslow.org

⊜ https://developers.google.com/speed/pagespeed

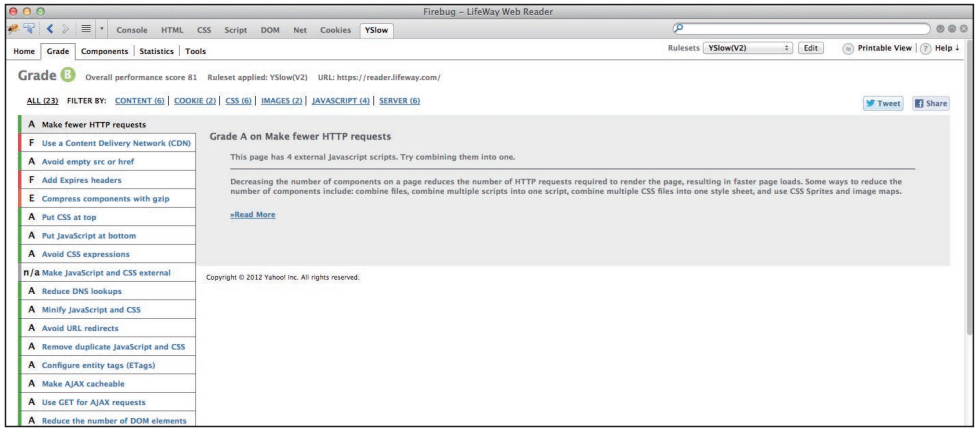


图 2-7 YSlow 性能分析工具

各主流浏览器自带的开发工具也可以辅助检测网页的性能，比如 Chrome 的开发工具。Chrome 的开发工具中有 Network、Timeline、Profiles 和 Audits 4 个菜单项，Network 中可以查看各资源请求和下载所用的时间，Timeline 中可以查看网页渲染和交互过程中各步骤所花费的时间，从资源的加载到 JavaScript 的解析执行、样式的应用及绘制，Profiles 中可以查看网页的 CPU 及内存占有情况报告，Audits 中提供了各种资源和配置优化的建议和未使用 CSS 规则的列表。这 4 项功能会极大地帮助开发者检测并优化网页性能。

2.5.4 代码和资源的压缩与合并

在网站发布时，一般会压缩前端 HTML、CSS、JavaScript 代码及用到的资源文件（主要是图片文件），目的是加快文件在网络中的传输，让网页以更快的速度展现。当然，CDN 分发、缓存等方式也是加快代码或资源文件传输的方式，但压缩代码和资源文件是最简单有效的手段之一。代码压缩的优点逐渐得到了大部分开发者的认可，并已经成为 Web 前端开发中不可或缺的一个步骤。以下是代码和资源压缩的具体实践方法。

（1）Web 服务器开启 Gzip 压缩

在 HTTP 中允许客户端可以选择从服务器上下载压缩的内容，Gzip 就是其中一种支持的格式。Gzip 是一种常见的压缩算法，可以参考维基百科中关于这种算法的详细介绍[⊖]。目前，Gzip 已经得到了大部分的主流 Web 服务器（比如 IIS、Nginx、Apache、Lighttpd 等）和主流的浏览器的支持。服务器配置人员可以查看服务器对应的配置文档，开启 Gzip 压缩。服务器启用了 Gzip 压缩后，代码文件有更小的体积，尤其是文本文件。

⊖ http://zh.wikipedia.org/wiki/Gzip

Web服务器开启Gzip压缩后，会在Response的header中增加Content-Encoding:gzip。可以通过检查此header项来判断服务器是否开启了Gzip压缩。

（2）JavaScript代码压缩

JavaScript压缩的原理一般是去掉多余的空格和回车、替换长变量名、简化一些代码写法等。JavaScript代码压缩工具很多，有在线工具、应用程序、编辑器插件。其中使用最多的是UglifyJS[㊀]、YUI Compressor[㊁]和Closure Compiler[㊂]。UglifyJS不仅仅是一个压缩工具，同时具有JavaScript语法分析和代码美化功能，包括代码缩减、代码转化等。如：

```
var a = 10; var b = 20; ==> var a=10,b=20;
if (foo) bar(); else baz(); ==> foo?bar():baz();
if (foo) bar(); ==> foo&&bar();
new Array(1, 2, 3, 4)  => [1,2,3,4]
```

Closure Compiler出自Google公司，功能和UglifyJS类似，只是压缩的算法不同。

YUI Compressor出自雅虎公司，此工具不同于以上两个工具。UglifyJS和Closure Compiler不仅是Compressor，同时还是Compiler，会针对代码进行优化，而YUI Compressor仅仅是压缩。

这3个工具都有对应的在线版本，也可以本地安装后使用命令行方式。如果不想手动压缩代码，则可以选择编辑器自动化集成方式，比如在WebStorm编辑器中配置File Watchers。

（3）CSS代码压缩

CSS代码压缩原理和JavaScript代码压缩的原理类似，也是去掉不必要的空格、回车、注释等，并同时优化合并一些CSS规则定义，让规则更简洁。比如下面的代码：

```
background-color:#fff;
background-image: url(image.gif);
background-repeat:no-repeat;
background-position: top left;
```

会优化为：

```
background: #fff url(image.gif) no-repeat top left;
```

CSS代码压缩工具也有在线、本地应用程序和编译器插件。比较有名的在线

㊀ https://github.com/mishoo/UglifyJS

㊁ http://developer.yahoo.com/yui/compressor

㊂ https://developers.google.com/closure/compiler

工具是 CSS Compressor[㊀]，压缩时可以选择各种配置模式。自动化工具可以选择上面提到的 JavaScript 压缩工具 YUI Compressor，它也可以压缩 CSS 代码。使用相同的工具压缩 JavaScript 和 CSS 代码，省去了安装多个工具的麻烦。

（4）HTML 代码压缩

压缩 HTML 代码的争议很大，反对的一方觉得压缩的作用不大，不像压缩 JavaScript 和 CSS 代码，压缩 HTML 代码仅仅是去掉空格、回车、注释等无关字符，并不会简化 HTML 代码本身，所以觉得在服务器开启 Gzip 就足够了。另外，压缩 HTML 代码也有一些限制，比如网站开发时，HTML 页面并不是单纯的 HTML 代码，而是和服务器端脚本混合在一起，比如 PHP、ASP、JSP、ASP.NET 等。压缩类似这些页面时，压缩的规则就比较复杂，也容易破坏代码。目前可以使用的 HTML 压缩工具并不多，使用最多的是这款工具：HtmlCompressor[㊁]。使用此工具的时候需要仔细调查和测试，避免压缩工具破坏页面的代码。

（5）图片资源压缩

除了代码的压缩外，网页中使用最多的资源文件就是图片。在一般的网站中，图片资源所占的比重还是挺大的。图片压缩工具主要是在线工具和本地应用程序，还没有一款使用较为广泛的编辑器插件可用。还好，在一般项目中，图片的变化并不是很大，图片的自动化压缩工具的需求并不是很迫切。

图片压缩工具也很多，以在线工具居多。在线工具中，推荐使用 TinyPNG[㊂]工具压缩 PNG 格式图片，推荐使用 JPEGmini[㊃]压缩 JPG 格式图片。本地应用程序，则推荐 ImageOptim[㊄]，这是一款强大的工具，集成了多种压缩工具，可以压缩多种格式（如 PNG、JPEG 及 GIF 动画）的图片，支持拖放操作，使用也非常方便。

以上就是网站发布时可以使用的各种资源压缩工具。除了在项目中使用合适的压缩工具之外，何时压缩代码也很重要。因为压缩后的代码会影响代码的调试，使得开发者不易确定出错代码的位置，调试 JavaScript 代码时也不易设置断点，所以推荐在开发后期，甚至是在网站的发布阶段做代码和资源的压缩。如果放在网站发布阶段压缩，则开发者在开发过程中不需要考虑代码压缩的问题，方便了开发者的开发与调试。发布阶段压缩代码的方案也很成熟，比如使用 ANT 工具。ANT 是一个基于 Java 的构建工具，在这个工具中可以构建代码和资源压缩的任务：使用 YUI 压缩程序合并和压缩 JavaScript 和 CSS，使用 jpegtran 和

㊀ http://www.csscompressor.com

㊁ https://code.google.com/p/htmlcompressor

㊂ http://tinypng.org

㊃ http://www.jpegmini.com

㊄ http://imageoptim.com

OptiPNG 优化 JPG 和 PNG 文件，使用 HtmlCompressor 压缩 HTML 及移除调试代码。

目前比较流行的前端自动化构建工具 Grunt[㊀]也可以集成代码和资源的压缩工具。在 Grunt 的插件列表页面上，有很多的压缩代码和资源文件的插件可用。比如，压缩 JavaScript 的 grunt-contrib-uglify[㊁]插件、压缩 CSS 代码的 grunt-contrib-cssmin[㊂]插件及压缩图片的 grunt-contrib-imagemin[㊃]插件等。

2.6　前端代码基本命名规范和格式规范

命名规范和格式规范是代码规范中最基本的规范，任何代码的混乱都是从命名和格式的混乱开始的，而意义明确的命名和规整的代码格式则提高了代码的可读性与可维护性，给代码的阅读者和维护者留下了良好的第一印象。命名规范和格式规范没有一个统一的标准，不同的人可能有不同的认识，但是在同一个项目中，必须严格遵守统一的命名和格式规范。以下推荐的规范是在实际项目中认同度较高的代码规范，供读者参考。

2.6.1　HTML 命名规范及格式规范

HTML 代码所有的标签名和属性应该都为小写，虽然 HTML 代码是不区分大小写的，但是 W3C 的规范建议小写；属性值应该使用双引号闭合。

不推荐示例：

```
<!-- 不推荐示例：标签名称大写，或者大小写混合；属性值没有用双引号闭合 -->
<IMG src=demo.jpg alt='test' />
```

推荐示例：

```
<!—推荐示例：标签名称小写；属性值用双引号闭合 -->
<img src="demo.jpg" alt="test" />
```

给所有的关键元素定义元素的 id 和 class，便于和 CSS、JavaScript 交互。因为 id 名称和 class 名称有可能作为检索值用在 JavaScript 代码中，所以命名一定要规范，这样才能保证不会出现不必要的重复而导致 Bug 的产生。

推荐的做法是根据语义和 DOM 树的层级关系来定义合适的名称，名称中全部使用小写，id 名称中的关键词用下划线（_）连接，class 的关键词用中划线（–）

㊀ http://gruntjs.com

㊁ https://github.com/gruntjs/grunt-contrib-uglify

㊂ https://github.com/gruntjs/grunt-contrib-cssmin

㊃ https://github.com/gruntjs/grunt-contrib-imagemin

连接，这样可以最大限度地保证命名的不重复。

不推荐示例：

```
<!-- 不推荐示例：命名简单随意，很难保证命名不重复 -->
<div id="Reader">
    <div id="introduce" class="Introduce ">
        …
    </div>
</div>
```

推荐示例：

```
<!-- 推荐示例：根据语义以及上下层级关系定义名称 -->
<div id="reader">
    <div id="reader_introduce" class="reader-introduce">
        …
    </div>
</div>
```

如果 class 名称仅作为 JavaScript 调用的“钩子”，则可在名称中添加“js”前缀。

示例代码：

```
<!--class 名称仅作为 JavaScript 调用的 " 钩子 "，可在名称中添加 "js" 前缀 -->
<ul id="js_reader_menu">
    <li class="menu-toc js-active">Toc</li>
    <li class="menu-store js-active">Store</li>
    <li class="menu-library">Library</li>
    <li class="menu-news">News</li>
</ul>
```

HTML 代码的层级缩进为 4 个空格。如果元素包含子元素，则此元素对应的起始标签和闭合标签分别单独占用一行。

不推荐示例：

```
<!-- 不推荐示例：标签树形层级之间没有缩进或者缩进混乱 -->
<ul>
<li>item1</li><li>item2</li>
<li>item3</li><li>item4</li></ul>
```

推荐示例：

```
<!-- 推荐示例：利用缩进体现元素的层级关系 -->
<ul>
    <li>item1</li>
    <li>item2</li>
    <li>item3</li>
    <li>item4</li>
</ul>
```

给 HTML 代码添加必要的注释。页面 HTML 代码的注释不宜过多，添加的原则是在保证代码维护性的基础上尽量让 HTML 代码简洁。基于这样的原则，可以在页面的公共部分（如页面的头部、尾部以及侧边栏等）、页面经常变化的部分（如广告栏）以及需要后端代码注入的部分添加注释。注释添加的位置在要注释的代码上部并单独占用一行，不要在代码行的后面直接添加。

示例代码：

```
<body>
   <!--main header-->
   <div id="reader_header">
      ...
   </div>
    <!--main content-->
   <div id="reader_content">
      ...
      <!-- 动态绑定列表 : toc-->
      <ul id="reader_content_toc">
      </ul>
   </div>
    <!--main footer-->
   <div id="reader_footer">
      ...
   </div>
</body>
```

2.6.2　CSS 命名规范及格式规范

推荐的 CSS 类的命名规则和元素的 id 命名规则相似，只是组成类名称的关键字的连接符为中划线（-）。

示例代码：

```
.reader-content-title {
   ...
}
```

为了避免 class 命名的重复，命名时取父元素的 class 名作为前缀。

```
/* 父元素的样式声明 */
.reader-content {
   ...
}
/* 子元素的 class 名称以父元素中的 class 名称作为前缀 */
.reader-content-body {
   ...
}
```

在 CSS 样式定义中，左大括号放置在选择器的同一行，并和选择器之间添加

一个空格分隔，在保证可读性的基础上缩短代码的行数；在样式声明中，属性名称和值之间用一个空格分隔，提高代码可读性。

不推荐示例：

```
/* 不推荐示例：CSS 样式定义中的左大括号单独占一行；样式声明没有缩进或缩进混乱；属性名称和值之间没有用空格分隔 */
.reader-content-title
{
background:#FFF;
   ...
}
```

推荐示例：

```
/* CSS 样式定义中的左大括号放置在选择器的同一行；样式声明中属性名称和值之间用一个空格分隔 */
.reader-content-title {
   background: #FFF;
   ...
}
```

多个选择器具有相同的样式声明时，每个选择器应该单独占一行，便于阅读和维护。

不推荐示例：

```
/* 不推荐示例：多个选择器具有相同的样式声明时，所有选择器放置于同一行 */
h1,h2,h3 {
   font-weight: normal;
   line-height: 1.2;
}
```

推荐示例：

```
/* 推荐示例：多个选择器具有相同的样式声明时，每个选择器应该单独占一行 */
h1,
h2,
h3 {
   font-weight: normal;
   line-height: 1.2;
}
```

样式声明的顺序按字母顺序排列，不考虑浏览器前缀。单纯靠手写代码并保证样式声明按照一定的顺序是不现实的。建议使用一些 CSS 美化工具做样式声明排序的工作。

示例代码：

```
/* 样式声明的顺序以字母序排列 */
.reader-content-title {
```

```
    background: #FFF;
    border: 1px solid;
    -moz-border-radius: 4px;
    -webkit-border-radius: 4px;
    border-radius: 4px;
    color: black;
    text-align: center;
}
```

样式定义按照模块来分组，相同模块的样式定义放在一起，不同模块的定义之间用一个空行分割。

示例代码：

```
/* reader header*/
.reader-header-title {
    ...
}
.reader-header-introduce {
    ...
}

/*reader footer*/
.reader-footer-copyright{
    ...
}
.reader-footer-links {
    ...
}
```

CSS 中的注释非常重要，能对 CSS 样式起到解释和说明的作用，提高了 CSS 代码的可读性。有些开发者可能担心添加过多的注释会让 CSS 文件行数增多，其实不用担心，可以在发布网站的时候对 CSS 文件进行压缩，这个过程中会去掉所有的注释。在 CSS 样式文件中添加注释主要有两种类型：文件头部的文件信息注释和正文中的解释说明性注释。文件信息一般包括文件版本、版权信息以及作者等；解释说明性的注释有给模块的注释和单独给选择器的注释，模块的注释则需要添加注释表明模块样式定义的开始和结束，CSS 选择器的注释需要添加在选择器的上一行，而不是和选择器相同一行。

示例代码：

```
/* 注释规范说明：文件头部的文件信息注释 */
/*!
 * reader content v1.0
 *
 * Copyright 2012
 * Dual licensed under the MIT or GPL Version 2 licenses.
 *
 * Designed and built by dangjian
```

```
 */

/* 注释规范说明：模块样式定义的开始和结束 */
/* Content containers start */
/* 注释规范说明：注释需要添加在选择器的上一行，而不是和选择器相同一行 */
/* content title */
.reader-content-title {
   ...
}
...
/* Content containers end */
```

2.6.3 JavaScript 命名规范及格式规范

JavaScript 局部变量命名采用首字母小写，其他单词首字母大写的方式。命名时建议采用有意义的单词命名，不推荐使用标识变量类型的前缀，如 int、str、obj 等。不推荐使用单词缩写命名，变量以缩写命名则降低了其可读性。如果认为变量名太长而使 JavaScript 脚本文件变大，则可以在发布阶段通过 JavaScript 脚本混淆压缩等手段来缩小文件。

不推荐示例：

```
// 不推荐示例：变量命名首字母大写
var ReaderBookmark = 'bookmark';
// 不推荐示例：变量命名意义不明确
var object = {};
// 不推荐示例：变量命名以类型作为前缀
var strName = 'Note';
// 不推荐示例：变量命名使用语义不明确的缩写
var newNT = function(){
    ...
}
```

推荐示例：

```
// 推荐示例：变量命名语义明确
var bookmarkDefaultTitle = 'Untitled Bookmark';
```

现在流行 JavaScript 的面向对象编程，那么就会有公有或私有接口的概念。原则上公有接口的命名为首字母大写，私有接口的命名为首字母小写。

示例代码：

```
Reader.Content = function(){
   // 私有变量
   var info, title;
   // 私有方法
   var getContent = function(){
      ...
```

```
    };

    return {
        // 公有方法
        SetTitle: function(contentTitle){
            title = contentTitle;
        },
        // 公有属性
        ContentInfo: info
    }
}();
```

jQuery 框架在项目中使用广泛，推荐给 jQuery 类型变量添加“$”作为前缀。

示例代码：

```
var $tocTitle = $('.reader-toc-title');
```

左大括号应该在行的结束位置，而不应该单独一行，因为这样增加了不必要的行数。应该一直使用大括号括起逻辑块，即使逻辑只有一行，也应该用大括号括起来，以便提高代码的可读性和可维护性。

示例代码：

```
// 左大括号应该在行的结束位置，而不应该单独一行
for (var i=0; i<100; i++) {
    doSomething(i);
}

// 应该一直使用大括号括起逻辑块，即使逻辑只有一行
var isFound = false;
if (statement) {
    isFound = true;
}
```

JavaScript 中可以用单引号或者双引号定义字符串，但是因为习惯于定义 HTML 的元素属性值时使用双引号，而 JavaScript 中又经常包含 HTML 代码，所以字符串定义使用单引号也可方便于在字符串内部包含含有双引号的 HTML 代码。

示例代码：

```
var content = '<span id="main_content">…';
```

空格的作用是提高代码的可读性，在函数参数的逗号后面使用一个空格，在操作符前后各使用一个空格。另外，使用一个空行来区分业务逻辑段。

示例代码：

```
doSomething(myChar, 0, 1);

while (x === y){
```

```
    ...
}
```

JavaScript 语句结束时应该添加一个分号。语句结束是否添加分号这个话题曾经引起很大的讨论，大名鼎鼎的 Bootstrap 框架中的 JavaScript 语句结束就没有添加分号。著名的框架都不在语句行尾添加分号，这里有必要简单介绍一下在行尾推荐添加分号的理由。首先来看看 JavaScript 是如何看待分号的。JavaScript 有自动插入分号的算法，在没有添加分号的 JavaScript 语句的结束处会自动添加一个分号，但是如果语句的下一行以“[”、“(”、“+”、“-”、“/”开头则不会在此语句后面添加分号。看似合理的设计，但其实如果应用不慎就会导致一些莫名其妙的错误，如下这个示例是由于自动添加分号而导致的逻辑错误。

错误示例：

```
return
{
    a + b
}
```

按照自动添加分号的算法，会在 return 后面添加一个分号，代码等价于：

```
// return 后面会添加一个分号
return ;
{
    a + b
}
```

其结果自然会返回 undefined，而不是期望的值。其实这个诡异的问题可以通过规定左大括号必须放置在前一个语句结尾处的方式来解决。

上面的例子是在不想添加分号的地方被自动添加了分号，而下面的例子则是因为没有在该添加分号的地方添加分号而导致的逻辑错误。

错误示例：

```
var b = function(){
    return function(){return 1}
}
var a = b
(function(){
    ...
})()
```

根据自动添加分号的算法，“var a = b”这行语句的后一行代码以左小括号开头，不会为这行语句自动添加分号，此行代码等价于：

```
var b = function(){
    return function(){return 1}
}
```

```
var a = b(function(){
    ...
})()
```

这完全背离了代码表达的初衷。当然，可以给以“[”、“(”、“+”、“-”、“/”开头的语句前添加一个分号来避免出现这样的逻辑错误，但是这也是一种“丑陋”的方案。

JavaScript 这种有缺陷的自动添加分号的算法希望开发者格外小心。开发者明白这些缺陷则有助于在实际的开发过程中避免犯错误。尽管在语句结尾添加分号和不添加分号都会有一些问题存在，但是考虑到大多数开发者已有的习惯，还是建议给语句的结尾添加分号。

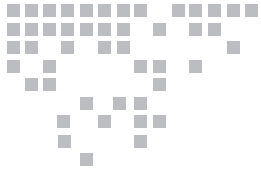
因为 JavaScript 代码在前端中是逻辑性最强的，所以需要添加足够的注释来保证代码的可读性。在 JavaScript 代码中，如果注释未占有多行，那么建议使用 //，不推荐使用 /**/。注释应该单独占用一行，而不是写在和代码相同一行的右边。和 CSS 代码的注释规范相似，JavaScript 代码的注释主要也是文件信息注释和代码逻辑注释。

示例代码：

```
/* 文件头部的文件信息注释 */
/*!
 * reader content v1.0
 *
 * Copyright 2012
 * Dual licensed under the MIT or GPL Version 2 licenses.
 *
 * Designed and built by dangjian
 */
Reader.Content = (function(){
   return {
      // reader 初始化
      Init: function(){
         ...
      };
    };
})();
```

第二部分 Part 2

HTML 最佳实践



Chapter 3 第 3 章

标准的 HTML 代码

标准的 HTML 代码指的是 HTML 代码符合 W3C 的最新标准，而在页面的 HTML 代码中包含有任何规范之外的或者是不推荐的标签和属性都是不符合标准的。W3C 定义了所有规范的 HTML 标签及其属性，目前正式的版本是 HTML5，HTML5 中定义的大部分内容已经被所有的高级浏览器支持，所以并不妨碍在页面中使用这些已经广泛被支持的新标签和新属性。本章将贴近 W3C 标准，介绍如何构建标准的页面 HTML 代码。

3.1 验证代码是否符合标准

一个符合 W3C 标准的网页会有什么重要的意义呢？这是一个讨论了很多年的话题，虽然有很多人在强调 Web 标准的重要性，可是比较遗憾的是这个话题并没有引起足够的重视，甚至有一些大的互联网公司的官方网站也很难通过 W3C 标准的检查。除了一些客观历史原因（如页面建立的时间久远、页面维护成本等）之外，我想这主要是前端技术还不够成熟的表现，只关注页面外在的表现而忽略了页面本身代码的质量。这些大的互联网公司的网站都没有起到带头的作用，其他的一些网站的页面代码质量就可想而知了。因此，在讨论页面标准化之前，先来说说一个标准的页面具有哪些优点。

1. 标准的页面会保证浏览器正确地渲染

尽管主流浏览器有很好的纠错能力，使得一些错误并不会影响页面的显示，

但是不同的浏览器处理错误的方式不尽相同，很难保证所有平台的所有客户端都能正确地兼容不规范的或者错误的代码。而使用符合标准的 HTML 标签和 CSS 样式，能最大限度地保证页面在不同浏览器正常地进行解析，同时还能最大限度地保证在未来的各种客户端中正常解析。

2. 网页能更容易被搜索引擎搜寻，提高网站的搜索排名

这一个优点是大家反复提到的。随着搜索引擎的普及，越来越多的用户从各种搜索引擎中查询需要的信息，网站的维护者也是想尽各种办法来提高在搜索引擎中的排名，从而提高网站的用户访问量。据统计，编写标准的页面基本上完成了一半的搜索引擎优化工作。各搜索引擎使用的网络“爬虫”不同于 Web 浏览器，“爬虫”的目的是读懂网站的内容，并找出网站中的关键字。良好的页面结构可以帮助搜索引擎准确地理解网站的内容，比如合理使用标题标签 <h1> 到 <h6>、设置 <img> 标签中的 alt 属性、选择更标准且更有语义化的标签等。网络“爬虫”会根据标签的语义和标签上的一些属性值来判断标签的内容所要表达的意思。

3. 提高网站的易用性

提高网站的易用性指的是让网站能被更多的用户访问，尤其是被一些视力或者肢体障碍用户等所访问。美国的某些地方甚至有专门的法律来要求发布的网站必须要达到一定的易用性。这样就可让那些特殊用户通过辅助的设备来阅读网站，这些辅助设备只会关注页面中的主要内容，并把内容以独特的形式（如通过语音阅读等）传达给用户。而符合标准的 HTML 和 CSS 组成的网页则会让辅助设备更容易识别，更准确地提取页面的主体内容。

4. 网页更好维护和扩展

这一点是针对网页开发的。W3C 的 Web 标准是被普遍接受的标准，页面的多个开发者如果遵循统一的标准，则会更好地理解和维护已有的页面。标签、样式以及行为分离的标准页面显然具有好的扩展性。

既然标准页面有这么多的好处，那么在前端的页面开发中如何做到编写的代码符合标准呢？编写标准的页面代码，首先要熟悉标准。W3C 定义了很多相关的标签、样式和行为标准，开发者只有深刻地理解了这些标准，才能编写出高质量的前端页面代码。除此之外，最直接有效的方式是使用工具来持续验证页面代码（主要是 HTML 和 CSS 代码）的标准性。

早在 2009 年，在 W3C 的社区中出现过一项有关是否需要验证页面的调查[㊀]，

㊀ http://www.w3.org/QA/2009/01/valid_sites_work_better.html

结果是大部分人赞成验证，并且有很多理由来说明验证页面代码的好处。除了包括如上标准页面本身的优点之外，验证页面的过程也使得开发者提高了自身的技术技能和职业素养。下面是常用的页面验证方式。

验证页面代码最直接的方式是用W3C提供的一项免费的验证服务W3 Validator[⊖]。它提供了3种验证方式：URL、文件上传、直接输入代码（见图3-1）。

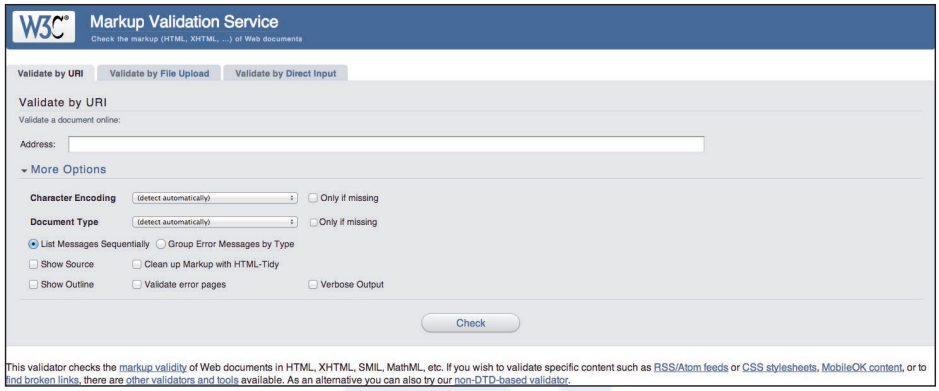


图3-1 W3 Validator工具使用界面

它验证的结果非常详细，每项错误或者警告都有对应的解释，因此它也是不错的学习工具。它的缺点是不够方便，尤其是在开发过程中需要持续验证的情况下。个人推荐使用的工具是HTML Validator，它是一个Firefox浏览器的插件，其方便之处在于可以在查看页面的同时验证页面。此插件的核心是基于一个开源的项目HTML Tidy[⊖]，是由W3C的Dave Raggett开发的。HTML Tidy相比于W3 Validator验证，强大的地方在于它不仅验证代码是否标准，还会自动纠正和美化代码。单击图3-2右侧的“Clean up the page”按钮，就可以美化页面的代码。

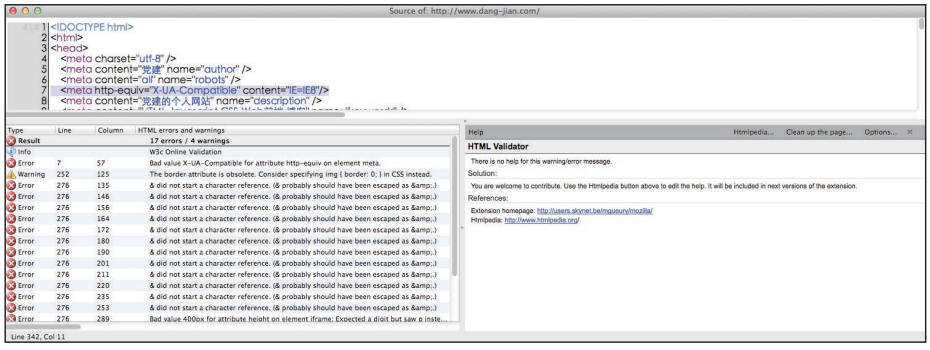


图3-2 HTML Tidy工具使用界面

⊖ http://validator.w3.org

⊖ http://www.w3.org/People/Raggett/tidy

需要强调的是，验证页面并不能保证开发者编写高质量的代码。验证页面的代码就像在写文章的过程中检查错别字以及语法错误一样，目的只是找到页面中直观的错误。页面代码符合标准只是基本要求，编写高质量的页面代码，还要注意很多细节，后续的章节会详细讨论如何编写高质量的页面代码。

3.2　标准的 HTML 页面结构

每个 HTML 文档都会包括两个部分：“ head ”和“ body ”。head 部分是以 <head> 开始并以 </head> 结束的，而 body 部分是以 <body> 开始并以 </body> 结束的。这两个部分由一对 <html>…</html> 包含在一起，可见一个基本的 HTML 文档结构就是：

```
<html>
    <head>
    </head>
    <body>
    </body>
</html>
```

head 部分包含整个文档的一些基本信息，如网站的编码格式、网站的标题、网站引用的样式和脚本等。body 部分包含用户在浏览器中看到的内容。

为了使得 HTML 文档能兼容标准格式，还需要给文档添加一个文档类型声明（DTD），当浏览器解析 HTML 文档时会遵循指定的类型声明标准。

HTML4 的规范中定义了多个规范的文档声明，如下是一个典型的使用示例：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
```

在 HTML5 的规范中简化了文档类型声明，省略了 DTD 的引用，表明文档以标准模式解析，示例代码如下：

```
<!DOCTYPE html>
```

因为浏览器可以识别 HTML5 规定的这种声明，所以只要 HTML 文档的内容够规范就不用担心浏览器是否能识别。

head 部分包含文档的标题，文档的标题是作为站点的名称和简短描述显示在浏览器的标题栏上的，示例代码如下：

```
<title> 网站标题 </title>
```

如果引用 JavaScript 和 CSS 外部文件，则需要把外部文件的链接添加到 head 部分。

示例代码：

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

此外，head 部分还会包含一些必要的 meta 标签，是对 HTML 文档内容的描述，用来表明文档的编码、关键字、介绍、作者等信息。

示例代码：

```
<meta name="keywords" content="HTML, web" />
<meta name="description" content=" 一个展示 HTML 页面的例子 " />
```

body 部分则包含所有在浏览器上展示的内容，如文本、图片、表格、音视频等。

示例代码：

```
<a href="/news">News</a>
<img src="green.jpeg" alt="Green" />
<video src="tgif.vid" autoplay onerror="failed(event)"></video>
```

综合这些页面的主要组成部分来看，一个稍微完善的页面会具有如下类似的结构：

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Sample page</title>
        <link rel="stylesheet" href="my_style.css" />
    </head>
    <body>
        <h1>Sample page</h1>
        <p>This is a <a href="demo.html">simple</a> sample.</p>
        <!-- this is a comment -->
    </body>
</html>
```

3.3 正确闭合 HTML 标签

HTML 元素的内容模型定义了元素的结构，表明元素可以包含哪些内容以及元素可以有哪些属性。元素可以包含的内容包括其他元素和字符，但是也有一些元素是空元素，即不能包含任何内容，这些元素对应的 HTML 标签也称为自闭合标签，下面列出了 HTML 中所有的自闭合标签：area、base、br、col、command、embed、hr、img、input、keygen、link、meta、param、source、track、wbr

W3C 制定了多个版本的 HTML 规范，目前流行的 HTML 规范有 XHTML 1.0、HTML 4.01 和 HTML5。规范中规定了所有 HTML 标签的语法，其中规定非自闭合标签必须有开始标签和结束标签，而自闭合标签没有闭合标签。示例代码

如下：

```
<!-- 非自闭合标签必须有开始和结束标签 -->
<a href="demo.html" title="demo">simple</a>
<!-- 自闭合标签必须没有结束标签 -->
<img src="demo.png" alt="demo" />
```

有关自闭合标签中是否应该添加符号“/”，在 XHTML 1.0、HTML 4.01 和 HTML5 的规范中稍有不同。XHTML 的规范最严格，必须在自闭合标签中添加“/”来表明标签的结束。在 HTML 4.01 的规范中，不推荐在自闭合标签中添加“/”。而 HTML5 最宽松，自闭合标签添加“/”和不添加“/”都符合规范，在自闭合标签中添加符号“/”是可选的。示例代码如下：

```
<!-- 如下写法符合 XHTML 1.0、HTML 4.01 和 HTML5 的规范，但在 HTML4 中不推荐 -->
<img src="demo.png" alt="demo" />
<!-- 如下写法不符合 XHTML 1.0 规范，但符合 HTML 4.01 和 HTML5 的规范 -->
<img src="demo.png" alt="demo">
```

如下的这几种写法不符合规范，应该严格禁止：

```
<!-- 错误：非自闭合标签没有结束标签 -->
<a href="demo.html" title="demo">simple</a>
<!-- 错误：非自闭合标签使用自闭合标签的语法 -->
<a href="demo.html" title="demo" />
<!-- 错误：自闭合标签使用非自闭合标签的语法 -->
<img src="demo.png" alt="demo"></img>
```

当元素的起始标签和结束标签不在同一个元素的内容中时，则会出现交叉嵌套。应该严格禁止标签之间的交叉嵌套。

下面的例子中起始标签“”在元素 div 的内容中，而结束标签“”则在 div 元素的内容之外。

```
<!-- 错误：起始标签 "<span>" 在元素 div 的内容中，结束标签 "</span>" 则在 div 元素的内容之外 -->
<div>foo <span>bar</div>baz</span>
```

一般通过编写层次缩进良好的 HTML 代码能够最大化地避免出现这样交叉嵌套的错误代码。

3.4　停止使用不标准的标签和属性，简化 HTML 代码

W3C 在制定的 HTML4 和 HTML5 标准中有独立的章节来说明哪些是不被推荐的标签和属性，这些标签和属性是 Web 互联网发展早期 HTML 标准混乱和浏览器“各自为政”的产物，有些标签甚至使用率颇高，比如滚动效果标签<marquee>，尽管在现在看来其效果不好，但在当时几乎所有的网站都会使用这

样的效果，风靡一时。但是随着 Web 互联网的飞速发展，人们对 Web 的认识也逐渐提高，也开始关注如何让网页 HTML 代码更统一、更简洁、更易理解等，不在局限于单纯的外观。在此过程中，一些标签和属性逐渐被更好的方案代替，这些标签也不被标准所推荐，甚至是从标准规范中移除，有些特性在浏览器中也不被继续支持。从这些标签和属性的作用来看，导致不被推荐的原因主要有如下几个。

（1）标签没有实际的语义，仅仅是用于设置样式

不推荐使用单纯设置样式的标签，如 basefont、big、center、font 等。应该通过 CSS 设置样式，让 HTML 标签功能更单一。不推荐的示例如下：

```
<!-- 不推荐代码：不推荐使用单纯设置样式的标签，应该通过 CSS 设置样式 -->
<font color=blue>don't use it!</font>
<big>don't use it!</big>
<center>don't use it!</center>
```

不推荐在 HTML 标签中添加样式属性，如 iframe、img、input、div 等标签中的 align 属性，body 标签上的 background 属性，td 和 tr 标签上的 height、width、nowrap、bgcolor、valign 等属性，iframe 标签中的 frameborder、marginheight、scrolling 等属性。此类属性应该废弃，并通过添加 CSS 样式来实现相同的效果。不推荐的示例如下：

```
<!—不推荐代码：标签中添加 border、width、height 等样式属性 -->
<img src="#" alt="demo" border="1" width="194" height="37" />
<div id="focusViwer" align=center></div>
```

不推荐使用 <blink> 或 <marquee>（闪动或滚动）。这两个标签的职能已经超出了 HTML 本身，并且也存在浏览器的兼容问题。以如今的审美来说，这两个标签实现的效果不佳，如果一定要这样的效果，可以通过 JavaScript 代码来实现，并且效果会更好，如可以使用 jQuery Marquee 插件[⊖]。不推荐的示例如下：

```
<!-- 不推荐代码：效果不佳，并且存在浏览器兼容问题，不推荐使用，如果需要实现这样的效果，可以通过 JavaScript 代码来实现，并且效果会更好 -->
<blink>don't use it!</blink>
<marquee scrollamount=3 scrolldelay=100 >don't use it</marquee>
```

（2）让 HTML 标签具有更好的语义

不推荐使用 <b> 和 <i> 显示黑体字和斜体，推荐使用更具有语义的 和 ，如果单纯是为了样式，推荐用 CSS 样式定义 font-weight 和 font-style，让页面更简洁。类似的不推荐标签还有 <s> 和 <strike>，这两个标签是给文字添加删除线的，可以用 <del> 和 <ins> 来代替。

不推荐示例：

⊖ https://github.com/aamirafridi/jQuery.Marquee

```
<!-- 不推荐代码：无语义的标签，单纯设置样式 -->
<b>don't use it!</b>
<i>don't use it!</i>
<s>don't use it!</s>
<strike>don't use it!</strike>
```

推荐示例：

```
<!-- 推荐代码：使用具有语义的标签，如果单纯为了样式，则应该通过 CSS 来设置 -->
<strong>important</strong>
<em>emphasize</em>
<del>deleted</del>
<ins>insert</ins>
```

（3）移除不常用的 HTML 标签

此类标签包括 acronym、applet、dir 等，废弃的原因是使用率极低或者是语义有歧义，并且有其他更好代替方案可以使用。例如，表达缩写的标签 <acronym>，其语义模糊，开发者更常用 <abbr> 来代替；开发者更喜欢使用 <ul> 而不是 <dir>。

3.5　样式与结构分离

网页的结构、表现和行为分离是 W3C 制定的 Web 页面标准的精髓，一般的页面结构和表现对应着代码中的 HTML 和 CSS，这意味着在页面中 HTML 和 CSS 分离是符合标准的做法。在讨论页面中 HTML 和 CSS 组合的最佳实践之前，先来简单说明一下 HTML 和 CSS 所有的组合方式。

把 CSS 样式应用于 HTML 总共有 4 种方式。

（1）在 HTML 页面中链接一个 CSS 文件

文件以 link 的形式添加到 <head> 部分，在 link 中可以设置 media 属性来表明样式使用的场景。例如，media 设置为 screen 表示正常浏览时的 CSS 样式，设置为 print 则表示页面打印时的 CSS 样式。

示例代码：

```
<link rel="stylesheet" href="default.css" />
<link rel="stylesheet" href="print.css" media="print" />
```

这种模式是最佳实践模式，做到了 CSS 和 HTML 的完美分离。如果需要修改页面的样式，则只需要修改 CSS 文件。此方式最大的优点是提高了性能。当多个页面共用一个相同的样式时，只会在首个页面访问时需要下载 CSS 文件，随后其他页面访问则只需要下载页面本身，CSS 文件可直接从缓存中取得，不用再次从服务器端下载，从而提高了页面加载的速度。

（2）在 HTML 页面中内嵌 CSS 样式

此方式是把 CSS 样式直接内嵌到 HTML 页面 <head> 中，所有的 CSS 样式放置在 style 标签中，示例代码：

```
<style>
    body { }
</style>
```

此方式虽然也做到了 CSS 和 HTML 的分离，但是因为 CSS 和 HTML 放置到了同一个文件中，不利于样式的重用，也不利于多人维护。如果 CSS 样式用在多个页面时，CSS 样式和 HTML 分离，可以利用 CSS 文件的缓存，提高多个页面整体的加载速度。当然，此方式也有一些优点，如果 CSS 样式只是用到单一的页面，则当 CSS 和 HTML 放置在同一个页面，会减少加载 CSS 样式文件的请求数量，加快页面加载速度。此外，在一些动态生成页面的场合，比如，在博客系统的自定义页面部分，将 CSS 和 HTML 放置在一起，方便输出和生成自定义页面。新浪网的首页正是把脚本和样式直接放置到了页面中，提高了页面加载速度。如果要合并 CSS 和 HTML 代码，一种好的实践是：在开发阶段，将 CSS 和 HTML 放置于单独的页面中，方便后续开发和维护。当代码发布时，通过工具合并 CSS 和 HTML 代码，提高页面实际运行时的性能。

（3）在 HTML 标签中添加内联 CSS 样式

此方式是样式直接添加到元素的 style 属性中。

示例代码：

```
<span style="color: green; background: transparent">green</span>
```

此种方式非常不利于页面的维护和样式的重用。很多糟糕的页面都是因为这种样式内嵌在元素中而导致页面代码庞大，难以维护。这种方式也降低了页面的可用性，并且也不利于搜索引擎对页面的识别。当然，此方式方便于通过 JavaScript 代码动态更改元素样式，如通过 JavaScript 代码给元素设置 display 样式值为 none 来控制元素的隐藏。jQuery 框架中也通过大量利用此方式来给元素动态设置样式，从而实现一些动态效果。

（4）在 CSS 样式文件中加载 CSS 样式文件

示例代码：

```
@import "mystyle.css";
@import url("mystyle.css");
```

这种方式是使用得较少的一种方式。每当看到前端性能相关的最佳实践时，总能看到一条：不要使用 @import。此方式最大的缺点是引用的 CSS 文件不能同时并行下载，这样延长了整个页面的下载过程，影响了页面加载性能。

以上 4 种引用 CSS 样式的方式中，第一种方式是作为最佳实践的方式推荐使用，第二种方式可以在某些场合有选择地灵活使用，第三种方式应避免在页面中直接使用，第四种方式在任何场合都应该避免使用。

3.6　添加 JavaScript 禁用的提示信息

作为 Web 编程语言，JavaScript 使得网页的交互更丰富，用户体验更好。大部分网页都会使用 JavaScript 开发页面上的一些交互功能。但不幸的是，并不是所有的浏览器都支持 JavaScript，比如 Kindle 内置的浏览器对 JavaScript 的支持就很弱，并且一般的浏览器也会提供让用户禁用 JavaScript 的选项。虽然目前 JavaScript 不起作用的场景很少了，但当 JavaScript 在浏览器上不起作用时，页面上的某些功能会不能正常工作，这样就影响这些少部分情况下的用户体验。如果开发的模块是个关键的模块，就需要考虑在 JavaScript 失效时如何让对应的功能正常使用。

最常用的方式是使用 <noscript> 标签。<noscript> 标签就是在当 JavaScript 被禁用或者不被支持时提供的一种代替方式，即 <noscript> 标签中的内容会在此时被浏览器解析，作为 JavaScript 不可用时的备选方案。<noscript> 在 HTML4 中引入，并在 HTML5 的规范中继续被支持。在 HTML4 中，<noscript> 只在 body 中起作用，在 head 中不起作用，但在 HTML5 中则允许 <noscript> 出现在 head 中。相比较 HTML4 规范中的定义，HTML5 规范中的定义提高了 <noscript> 标签使用的灵活性。

<noscript> 标签常规的用法是当 JavaScript 不可用时显示提示信息。

示例代码：

```
<script type="text/javascript">
    // 一些操作
</script>
<noscript>
    <p> 浏览器不支持 JavaScript</p>
</noscript>
```

从 <noscript> 的作用来看，这种方式很不灵活，使用上也有一些缺陷：<noscript> 标签只支持 HTML，不支持 XHTML，并且只能作为一个块元素；<noscript> 只在 JavaScript 被禁用时才起作用，在 JavaScript 因为诸如防火墙拦截等情况下不可用时，并不会起作用；<noscript> 也并不能很好地解决 JavaScript 不可用时网站的可用性问题，尤其是对于大量使用 JavaScript 及 AJAX 技术更新页面内容的网站。下面来看看 W3C 对此的态度，如下是从 W3C 官网上摘录的一段话[⊖]：

⊖ http://www.w3.org/TR/html5/the-noscript-element.html#the-noscript-element

The noscript element is a blunt instrument. Sometimes, scripts might be enabled, but for some reason the page's script might fail. For this reason, it's generally better to avoid using noscript, and to instead design the script to change the page from being a scriptless page to a scripted page on the fly.

大概的意思是说，<noscript> 标签不够灵活，有些时候 JavaScript 不可用并不是因为脚本被禁用导致的。因此，最好是不要使用 <noscript> 标签，而是更改设计，让页面从无脚本模式过度到有脚本的模式，即从不支持脚本到支持脚本的渐进增强，从而保证两种模式下页面都可用。如下是给出的示例：

```
<form action="calcSquare.php">
  <p>
    <label for=x>Number</label>:
    <input id="x" name="x" type="number">
  </p>
  <input id="submit" type=submit value="Calculate Square">
  <script>
    var x = document.getElementById('x');
    var output = document.createElement('p');
    output.textContent = 'Type a number; it will be squared right then!';
    x.form.appendChild(output);
    x.form.onsubmit = function () { return false; }
    x.oninput = function () {
      var v = x.valueAsNumber;
      output.textContent = v + ' squared is ' + v * v;
    };
    var submit = document.getElementById('submit');
    submit.parentNode.removeChild(submit);
  </script>
</form>
```

在该示例中，当 JavaScript 可用时，在客户端运行 JavaScript 代码计算输入值的平方值。当 JavaScript 不可用时，把输入值提交到服务器端来计算输入值的平方值。这个示例很清晰地演示了 JavaScript 不可用和可用两种模式下，页面渐进增强的设计理念。

从上述示例中可以看出，W3C 也意识到了 <noscript> 的局限性，并给出了推荐的方案。但实际的项目中很少有网站的设计采用这一方案，不过，某新闻网站很好地采用了此方案，网站整体做到了 JavaScript 脚本启用时的渐进增强设计。图 3-3 显示网站脚本禁用时的效果。

图 3-4 是脚本启用时的显示效果，可以看到，当脚本可用时，整个新闻列表按照类型显示并拥有了一个幻灯播放效果。

该网站首先按照静态网页的布局设计了新闻的文字图片列表，然后在如上显

示的脚本中按照新闻类别分别添加了类别标题，并构建了新闻的滚动幻灯播放效果，很巧妙地实现了脚本启用时的渐进增强设计。

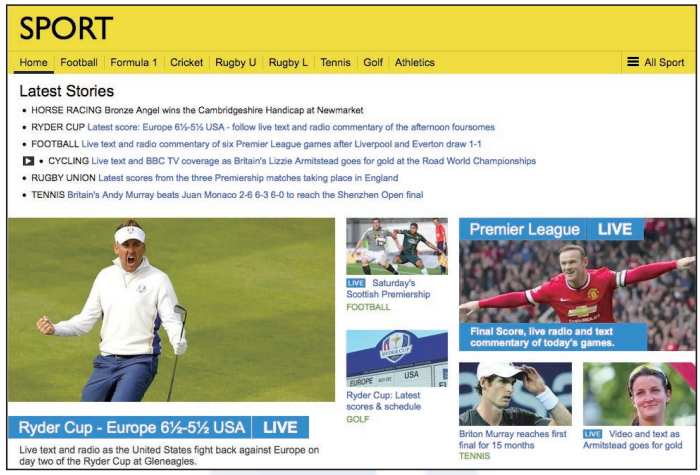


图 3-3　脚本禁用时，某新闻网站部分内容展示效果

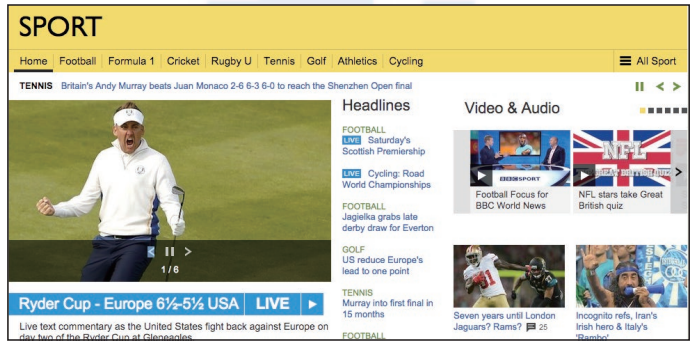


图 3-4　脚本启用时，某新闻网站部分内容展示效果

尽管有上述新闻网站等这样成功的案例，但是在实际的情况中，一般还是很难设计出合理的渐进增强效果，并且在大部分网站的使用过程中 JavaScript 脚本不可用的情况只占很小的比例，考虑到上述原因，推荐的做法是在技术实现的代价和网站的可用性之间做一个平衡。但可以肯定的是，针对 JavaScript 被禁用的客户端不做任何相应的设计是不专业的做法，降低了网站的可用性。那么实际的项目中应该如何应对 JavaScript 不可用的状况呢？

最佳的实践是，提示用户 JavaScript 已被禁用，并同时提供一个功能简单、不依赖于 JavaScript 的代替网站供用户继续浏览，做到平稳降级。如在首页中，当脚本不可用时会提示用户，同时会提供一个不依赖于脚本的移动站点作为代替的站点，图 3-5 给出了浏览器禁用脚本时 BBC 网站的展示效果。

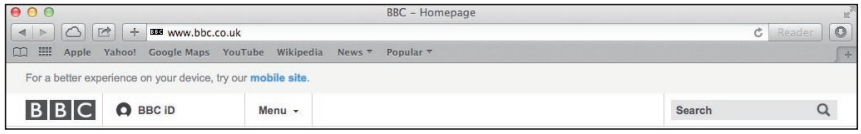


图 3-5 BBC 网站在脚本禁用时的展示效果

当然，也可以直接跳转到一个不依赖脚本的代替页面中。图 3-6 是百度首页在禁用脚本时的做法。

图 3-6 百度首页在脚本禁用时使用的代替网页

为了做到禁用脚本时页面自动跳转，百度首页中添加了如下的代码：

```
<noscript><meta http-equiv="refresh" content="0; url=/baidu.html?from=noscript"/></noscript>
```

为了使得页面在脚本不可用时还能正常展示，可能需要针对部分模块设置区别脚本的禁用和启用时页面的不同风格。常用的方式是给页面的 HTML 标签添加一个名为 no-js 的 class，并在脚本中添加移除此 class 的逻辑。在样式代码中可以这样设置不同状态下的样式：

```
/* 脚本启用时对应的样式 */
.product {

}

/* 脚本不可用时，通过覆盖以上定义的样式或者添加额外的样式来设置不同的外观 */
.no-js .product {

}
```

3.7　添加必要的 <meta> 标签

<meta> 标签放置在 HTML 页面的 head 中，主要用于标识网站。其中基本上包含了网站的一些描述信息，如简介、作者等。这些信息有助于搜索引擎更准确地识别网页的内容，也有助于第三方工具抓取网站基本信息。

按照 W3C 的标准介绍，<meta> 元素有 4 个属性：name、http-equiv、content 和 charset。<meta> 标签通过 name 属性来表述页面文档的元信息，通过 http-equiv 属性设置 HTTP 请求指令，通过 charset 设置页面的字符编码。按照属性设置分类，<meta> 可以分为三类：

（1）name 属性和 content 属性组合，构成名称 / 值对，用于描述网站信息

这是 <meta> 标签最主要的功能之一，使用广泛。标准的 <meta> 名称包括：application-name、author、description、generator 等。

示例代码：

```
<!—页面关键字 -->
<meta name="keywords" content="british,typeface,font,fonts" />
```

此类型 meta 使用最广泛，其中 keywords 和 description 这两个名称的使用率最高，是搜索引擎优化的主要手段之一，推荐读者使用。设置 keywords 和 description 这两个 meta 时，尽量使用简洁和语义明确的词语，下面的示例展示的是某新闻网站设置的 application-name、keywords 和 description 对应的 meta 信息：

```
    <meta name="application-name" content="BBC"/>
    <meta name="description" content="Breaking news, sport, TV, radio and a whole lot more. The BBC informs, educates and entertains - wherever you are, whatever your age." />
    <meta name="keywords" content="BBC, bbc.co.uk, bbc.com, Search, British Broadcasting Corporation, BBC iPlayer, BBCi" />
```

（2）http-equiv 属性和 content 属性组合，设置特定的 HTTP 指令

根据 W3C 制定的 HTML5 规范，指令型 meta 总共有 5 种，其中 content-type、default-style 和 refresh 已经确定，content-language 和 set-cookie 还未正式确定。

示例代码：

```
<!—页面加载 5 分钟后刷新 -->
<meta http-equiv="refresh" content="300" />
```

此类型 meta 应该谨慎使用。<meta http-equiv="content-type"> 可以使用下面介绍的更简洁的方式代替。不推荐使用 <metahttp-equiv="refresh">，某些搜索引擎遇到此 meta 时会停止解析页面剩余的部分。<metahttp-equiv="default-style"> 在实际的场景中很少使用。

（3）charset 属性，设置页面字符编码

此属性功能单一，提供了一种保存和传输文档的编码格式。

```
<!—声明文档为 UTF-8 格式 -->
<meta charset="utf-8">
```

这是在 HTML5 中新加入的 meta 类型，在 HTML5 的规范中，如下两种页面编码设置是等价的。

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

代码中的第二种形式更简洁好记，并且得到了所有主流浏览器的支持（尤其是 IE 8、IE 7 和 IE 6），所以不存在浏览器兼容问题，推荐在设置页面编码时使用。为了让浏览器能准确识别编码格式，务必在 <title> 标签之前设置 charset，保证标题能正确显示，示例如下：

```
<head>
    <meta charset="utf-8">
    <title>My home page</title>
</head>
```

以上介绍了三种规范中定义的 meta 格式，如果想要了解更详细的内容，可以参考 W3C 的规范，规范中很详细地描述了 meta 的种类及其作用。

除了规范中定义的这些 meta 之外，还有大量的自定义 meta 类型。这些 meta 类型主要是由互联网公司或者浏览器厂商为了实现特定的功能而定制的，W3C 规范中允许自定义 meta 类型，但为了防止自定义的 meta 名称重复，所有的自定义 meta 应该事先注册。目前已经有很多注册的自定义 meta，以及一些厂商自定义名称[⊖]和自定义的指令[⊜]。这些自定义的 meta 并不能通过 W3C 提供的标准校验，但并不是说这些 meta 不标准。下面介绍一些常用的 meta。

（1）设置 IE 浏览器的兼容模式

从 IE 8 浏览器开始支持一种设置页面兼容模式的 meta 类型，示例代码如下：

```
<meta http-equiv="X-UA-Compatible" content="IE=8"/>
```

根据 HTML 规范，浏览器是按照页面开头定义的文档类型来解析页面的。例如，使用 HTML5 的文档类型声明：

```
<!DOCTYPE html>
```

IE 就会以标准模式解析 HTML 文档。但是某些已有页面由于各种原因不能在最

⊖ http://wiki.whatwg.org/wiki/MetaExtensions

⊜ http://wiki.whatwg.org/wiki/PragmaExtensions

新标准模式下正确显示，只支持特定的标准。针对这种状况，IE 浏览器提供了一种兼容的方案，通过设置 X-UA-Compatible 指定页面在 IE 浏览器中渲染时执行的标准。在 IE 浏览器中，此设置的优先级高于通过 DOCTYPE 设置的文档标准。有关此 meta 的具体使用方式可参考微软公司网站中的相关介绍[⊖]。在实际的项目中应该谨慎使用此方式，当在 IE 浏览器中出现兼容问题时，最好的做法是修改页面，编写规范的 HTML 代码，让页面支持最新的标准，而不是设置 IE 的兼容模式。

此设置还有一种常见的设置值，即：

```
<meta http-equiv="X-UA-Compatible" content="chrome=1">
```

很疑惑吧，IE 定义的 meta 为什么会出现 chrome 呢？其实设置为 chrome=1 时，则会在 IE 9 及以下浏览器中激活 Chrome Frame[⊖]，强制 IE 使用 Chrome Frame 渲染页面。

（2）设置页面在移动设备中的显示

一般针对移动设备优化的网页都会添加如下一条 mata 设置，使得网页在移动设备中显示正常，设置的代码类似如下的代码语句：

```
<meta name="viewport" content="width=device-width, initial-scale=1,
maximum-scale=1" />
```

在后面讨论移动设备的前端开发时，会详细解释此设置的来龙去脉以及在不同的场景中如何正确设置 viewport。

（3）设置 IE 浏览器的“固定网站”功能

这是 IE 浏览器的独有功能，是从 IE 9 开始支持的，它能够将网站如同应用程序一样固定在 Windows Vista 及以上版本系统的任务栏中，并且在点击图标后显示网站相关的菜单列表。此功能可以通过定义网页 meta 来实现，如下代码用于展示人人网[⊜]首页中添加的 meta 定义。

```
<meta name="msapplication-task" content="name=新鲜事;action-uri=http://
www.renren.com/home;icon-uri=http://a.xnimg.cn/n/res/icons/newsfeed.ico" />
<meta name="msapplication-task" content="name=日志;action-uri=http://
blog.renren.com;icon-uri=http://a.xnimg.cn/n/res/icons/blog.ico" />
<meta name="msapplication-task" content="name=相册;action-uri=http://
photo.renren.com;icon-uri=http://a.xnimg.cn/n/res/icons/photo.ico" />
<meta name="msapplication-task" content="name=音乐;action-uri=http://
music.renren.com;icon-uri=http://a.xnimg.cn/n/res/icons/music.ico" />
<meta name="msapplication-task" content="name=分享;action-uri=http://
share.renren.com;icon-uri=http://a.xnimg.cn/n/res/icons/share.ico" />
```

⊖ http://msdn.microsoft.com/en-us/library/jj676917（v=vs.85）.aspx

⊖ http://www.google.com/chromeframe?hl=zh-CN&prefersystemlevel=true

⊜ http://www.renren.com

```
    <meta name="msApplication-ID" content="App" />
    <meta name="msApplication-PackageFamilyName" content="57722RenRenpreview.
RenrenHD_fknrsfzqca1jw" />
```

添加到系统的任务栏后的效果如图 3-7 所示，可以看到在图标上单击右键后弹出的快捷菜单中包含有 5 项定义的菜单项，单击这些菜单项则可以跳转到对应的页面中。

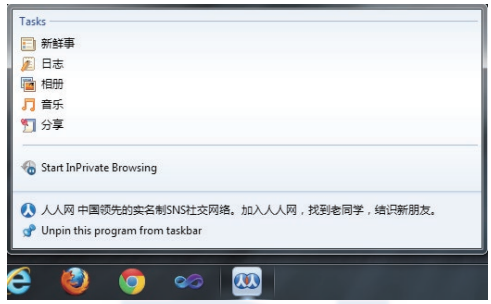


图 3-7 人人网网站定义的网站导航菜单

如果想更深入地了解具体如何设置对应的 meta 来实现此功能，则可以参考微软公司提供的详细文档[⊖]。

以上就是在页面的 meta 信息中使用较多的设置，随着移动设备的兴起，出现了很多针对移动设备的 meta 信息，比如针对 iOS、Android 及 Windows 8 系统的 meta 信息，这部分内容将在后面章节中介绍 Web 移动开发相关最佳实践时进行详细讨论。

⊖ http://msdn.microsoft.com/zh-cn/library/ie/gg491725（v=vs.85）.aspx