TURING 图灵程序设计丛书

Apress®

Pro CSS for High Traffic Websites

高流量网站CSS开发技术

Web设计最强音，打造和维护高流量网站的必杀技

[英] Antony Kennedy
[葡萄牙] Inayaili de León 著
大胖 王永强 译

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Pro CSS for High Traffic Websites

高流量网站CSS开发技术

[英] Antony Kennedy
[葡萄牙] Inayaili de León 著
大胖 王永强 译

人民邮电出版社
北京

图书在版编目（CIP）数据

高流量网站CSS开发技术 / (英) 肯尼迪(Kennedy,A.)，(葡) 利昂著；大胖，王永强译. -- 北京：人民邮电出版社，2013.10
（图灵程序设计丛书）
书名原文：Pro CSS for high traffic websites
ISBN 978-7-115-32935-6

Ⅰ. ①高… Ⅱ. ①肯… ②利… ③大… ④王… Ⅲ. ①软件工具－程序设计 Ⅳ. ①TP311.56

中国版本图书馆CIP数据核字(2013)第206814号

内 容 提 要

不论是前端工程师还是后端工程师，编写 CSS 可不只是码几行代码那么简单的事儿，他们需要面对的情况通常非常复杂。出色的 CSS 开发人员知道如何应对和避免跨浏览器陷阱，懂得处理语义化、无障碍访问、搜索引擎优化，以及相关文档不完整带来的各种问题。

一套优秀的 CSS 框架可以显著提升网站性能，但如何创建一套优秀的 CSS 框架呢？单纯使用一些新特性只会让情况变得更糟糕，要确保 CSS 不会成为开发过程或网站性能的瓶颈，请遵循本书的指导！

本书针对高流量网站开发人员及项目经理，当然那些在小型团队工作，以及缺乏相关经验的开发人员更应自觉学习本书的真知灼见。

◆ 著　[英] Antony Kennedy　[葡萄牙] Inayaili de León
译　大　胖　王永强
责任编辑　刘美英
责任印制　焦志炜

◆ 人民邮电出版社出版发行　北京市崇文区夕照寺街14号
邮编　100061　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京艺辉印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：21.5
字数：508千字　2013年10月第1版
印数：1－3 000册　2013年10月北京第1次印刷

著作权合同登记号　图字：01-2011-3260号

定价：69.00元

读者服务热线：(010)51095186转604　印装质量热线：(010)67129223
反盗版热线：(010)67171154
广告经营许可证：京崇工商广字第 0021 号

站在巨人的肩上

Standing on Shoulders of Giants

www.turingbook.com

版权声明

Original English language edition, entitled *Pro CSS for High Traffic Websites* by Antony Kennedy and Inayaili de León, published by Apress, 2855 Telegraph Avenue, Suite 600, Berkeley, CA 94705 USA.

Copyright © 2011 by Antony Kennedy and Inayaili de Leon. Simplified Chinese-language edition copyright © 2013 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由Apress L.P.授权人民邮电出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

版权所有，侵权必究。

此为试读，需要完整PDF请访问：www.ertongbook.com

序

这年头，任何人都想鼓动大家抛弃老浏览器，拥抱新技术。虽然拥抱新技术是件好事，值得称赞，但很多开发者还是得支持老浏览器，或者要抄近路来赶工，不然项目就要延期，老板就要发飙。尤其在高流量的场合，不妥协是不行的。

基于这个出发点，本书能让我们透透气儿。作者 Antony 和 Inayaili 没有责怪你的意思（你是迫于公司要求不得已才做了那些事），而是尽可能地告诉你如何更好地找到平衡点，帮你选择正确的战场。编写 CSS 不光是处理代码，想要成为开发者中的大牛，你要知道它内部的工作原理、浏览器显示网站的不同方式、可能要面对的不同环境、性能、无障碍访问、语义化、可用性……

在大公司，有太多开发者为了图自己方便，而把用户放在了次要的位置。本书建议你反其道而行，帮助你做出一些能让自己和公司引以为傲的好东西，同时又不会在任何方面偏于极端。这就是实战中的 CSS。

Bruce Lawson，*Introducing HTML5* 作者[①]，Opera 开放标准布道师

① 参见 http://introducinghtml5.com，中文版《HTML5 用户指南》（机械工业出版社，2011）。——译者注

前　言

既然你已经拿起这本书，不妨假设你正在一家具有相当规模的公司工作。这意味着你要面对官僚主义和红头文件，而且有各个层面的员工和你同场竞争，你不可能立刻修改软件或网站，因为还要遵循一大堆既有的协议和流程。如果这些描述不符合你的状况，那非常幸运，你的公司属于少数派。

这里提到的事情听上去都是负面的，但多数都不可避免——特别是在快速成长的公司里。公司越大，响应变化的速度通常也就越慢。Antony 还记得，曾经花几个月时间帮某上市公司做了一个项目。结果到项目完成参观客户办公室时才发现，这家公司本身就有现成的软件部门，完全有能力承担他刚刚完成的项目，而且还更合适。在大公司这种事很常见，它们太大了，大到员工之间无法有效交流工作。不同部门经常发现他们在做重复的事情，原因只是没有沟通渠道。

不同公司的驱动力量可能不同，有的公司是财务驱动，有的是声誉驱动，有的是品牌驱动，还有的是设计驱动，难以尽数。公司的驱动力定义了公司的商务活动和文化。除非是品牌或设计驱动的公司，否则不太可能从一开始就考虑所开发软件的视觉框架。

在这些公司的网站开发团队中，CSS 开发者经常会被忽视。很多人把 CSS 当成简单易学的东西，把它交给服务器端开发者而不是 Web 方面的专家来做。此外，他们经常在最后一分钟才考虑 CSS 相关的框架和静态文件的问题，而不是一开始就把它们当做基础设施的一部分。尽管 CSS 的语法简单，但实现所带来的影响却不可小觑。一位出色的 CSS 开发者有经验应对跨浏览器陷阱并知道如何加以避免，懂得处理语义、无障碍访问、搜索引擎优化，以及由于 CSS 编写糟糕或缺少文档所带来的各种问题。一套完善的 CSS 框架可以显著提升网站性能。此外，一套完整的 CSS 框架文档可以方便多位开发者以一致的风格工作。特别是在网站生命周期的开始阶段，在起跑时就正确完成这些事情，最终肯定能受益。

我们不会承诺能让你的公司实现快速响应、良好沟通，提高员工幸福度。但是贯穿本书的目标是通过某些可供遵循的流程，帮你打一个坚固的地基，确保 CSS 不会成为开发过程或网站性能的瓶颈。在最后一章和附录中，我们会提供本书涉及内容的应用示例，示例代码请参考本书网站（http://www.procssforhightrafficwebsites.com/）。

致　谢

编写本书的过程是漫长而令人激动的，不眠了多少个夜晚，放弃了多少个周末，多少的耐心和坚持啊，终于——我们华丽转身，成了 CSS 图书的作者。但是，如果没有下面这些人的卓尔不凡、忍耐宽容，以及最重要的激励鼓舞，这一切都是不可能的。

感谢在这一奋斗过程中支持过我们的每个人，你们的鼓励让我们怀揣这个在某次饭局之后冒出来的小小梦想前进，特别感谢一直激励我们走到底的人（包括每次我们心灰意冷都能帮我们重拾梦想的人）。

非常感谢我们的技术审稿人 David Storey，他的意见弥足珍贵，很大程度上提升了本书的质量。感谢推荐序作者 Bruce Lawson 为我们提供的宝贵意见。

特别感谢：Adam Lang、André Luís、Andrew Fox、Andy Beeching、Anthony Killeen、Calum Land、Christian Heilmann、Dan Jeffrey、Darren Waddell、Edwina Arney、Felix Kennedy、Gavin Williams、Geoffrey Robichaux、Gernot Poetsch、Gonçalo Melo、Ian Pouncey、Isa Costa、Ivo Gomes、Jade Thomas、Jamie Newman、James Newbery、Janak Patel、Kushal Pisavadia、Leah Davenje、Marc Tobias Kunisch、Mark Stickley、Matt Gaddis、Nick Holmes、Nicklas Persson、Paddy Donnelly、Paul Stanton、Sijmen Mulder、Stephanie Hobson、Talia Kennedy、Than Khine、Velcro、MF Boat 上的每个人、The Mailing List™ 中的每个人，以及所有在 Twitter 上帮助过我们的人。

还要感谢一些人的工作，不但为我们，而且为世界各地的 CSS 作者、设计师和开发者提供了灵感：Andy Budd、Eric Meyer、Nicole Sullivan、Paul Irish、Steve Souders、Tantek Çelik，等等。

当然，如果没有 Apress 的支持和相关人员的参与，这本书也无法完成，特别感谢本书编辑 Ben Renow-Clarke 和 Mary Tobin。

最后作者 Antony 和 Inayaili 在这里互致感谢：写一本书很难，但也很好玩。

背　景

本书假定读者已经具备某些知识。在这里我们会对预备知识做出说明，并解释整本书所涵盖的内容。如果你是多年的程序员，或在 Web 行业有相当的工作经验，则可以略过这一部分直接阅读第 1 章。

本书写给谁

虽然书中涉及的方法和主题都配有相关的 CSS 代码示例，但这些示例并不是为了演示最新的 CSS 选择器和属性，而是为了演示如何对代码进行格式化和注释，以保持清晰的条理，并符合最佳流程实践。为此，本书会同时兼顾整个团队和具体开发人员的需求，以期对所有人都有相应的价值。

本书适合以下读者：

- 所有在高流量网站工作的人，所谓高流量指独立访问用户数日均 1 万以上，或峰值超过 1 万；
- 所有在大型网站工作的人，大型网站至少要包括 2000 个独立网页，或 30 个子网站；
- 所有在公司负责网站开发的人，所在公司有大量员工在同一个代码库上工作，参与修改 CSS 的程序员超过 30 人；
- 所有在有潜力的公司工作、希望为公司 Web 开发建立一套良好流程的人；
- 所有缺少大型团队工作经验的开发人员。

同时，书中内容也可作为大小网站均可采用的最佳实践。

本书写什么

通过阅读这本书，你将了解以下内容：

- 流程的价值；
- 如何在员工和团队之间共享知识；
- 如何尽快让新来的 CSS 程序员上手工作；
- 如何将 CSS 修改纳入构建、部署流程；
- 如何编写可重用、模块化的 CSS；

- 如何最大限度提升网站性能；
- 如何保持品牌的一致性；
- 跨浏览器和无障碍访问的 CSS 最佳实践；
- 动态 CSS 技术。

最后一章提供了一套简单的 CSS 框架，是我们为本书定制的，其中演示了大量书中讨论过的内容，包括创建这个框架时所遵循的流程。书后的四个附录提供了规范和流程的具体例子，供读者参考。

本书哪里与众不同

着手写作之前，我们做了大量调查，查找可用的替代资源。而那些讲解 CSS 基础知识、CSS 高级技巧、CSS3 选择器/属性，以及各种 CSS 设计模式的图书，可谓浩如烟海，读者唾手可得。

本书无意在这些领域标新立异，而是探讨了大型团队或多个团队一同工作的情况，以及处理包含众多页面或子站点且访问量相当可观的网站时，所要面对的挑战。我们不介绍如何用最新最炫的技术实现图片替换或跨浏览器的圆角，而将关注点放在让团队中的新人更容易理解原有代码，并在其上添砖加瓦，让网页中的 CSS 代码从一开始就经过深思熟虑，且以完善、高效的方式构建。

本书对新手和专家同样有用。不过，若是你已经有一些 HTML 和 CSS 的使用经验，或至少熟悉相关语法，那再好不过了。对代码重用、模块化、健壮性和实用性的讨论，将贯穿本书各章。

关注点分离

关注点分离是编写 CSS 时要理解的一个重要概念。程序和系统的体系结构多种多样，要论证它们的优劣远远超出了本书的范围。然而，对多层体系结构做个简单说明还是很有必要的，其逻辑思想对于 CSS 和浏览器应用程序非常适用。

多层体系结构设计方法是指在设计中将逻辑、数据和表现分离[①]。通常用它来描述的客户端/服务器系统，就包括一个客户端程序和一个服务器端程序。大部分的处理过程由服务器完成，客户端侧重于信息显示以及给用户提供操作界面。此时的多层结构中包括：

- 客户端应用（表现层）；
- 应用（逻辑层）；
- 服务器（数据层）。

对这三层进行分离，使得代码、功能、任务的任何一部分各归其位、各司其职，也为重用和维护提供了方便。

这就是传统的客户端/服务器程序的运作方式。“浏览器—服务器—数据库”同样也是明显的多层结构，但我们可以更进一步，对运行在 Web 浏览器中的代码做同样的划分。在前端代码中，

① 模型—视图—控制器（MVC）是一个例子。

通常会使用 HTML、CSS、JavaScript 三种“语言”。这三者可能彼此混杂，类似下面的例子：

```
<a id="clickableAnchor" href="#" style="color:blue;" onclick="alert('点到了!');">点这里</a>
```

单就技术而言，这里并无不妥，运行结果就是蓝色的“点这里”，用户点击即运行 JavaScript。然而这样做属于不良实践，有以下几个原因。

- 不同类型的代码没有明确区分。如果 CSS 程序员要改变文本的颜色，就不得不修改这个网页。如果 JavaScript 程序员想要改变提示框的文字，也不得不直接修改这个网页。
- 这样的代码没法重用。假如所有的代码都这么写，一旦需要让所有的链接文字都变成红色，就只好逐个修改每处链接。
- 这样的代码在网页上越多，用户要下载的网页文件就越大。在某种程度上，这也印证了上一条理由，因为可重用的外部代码只需要下载一次。进一步说，网页文件越大，越不利于搜索引擎优化（SEO）。
- 在同一文件中，代码的种类越少，代码就越容易阅读、分析和修改。

一种更合适的写法如下所示。

在<head>部分：

```
<style>
        #clickableAnchor {color:blue;}
</style>
<script>
        $('#clickableAnchor').click(function(){
                alert('clicked!');
        });
</script>①
```

在<body>部分：

```
<a href="#" id="clickableAnchor">点这里</a>
```

这样的写法分开了三种语言，因而就整洁得多。但还有更好的解决方案。

在<head>部分：

```
<link rel="stylesheet" href="/css/screen.css" />
<script src="/js/main.js"></script>
```

在<body>部分：

```
<a href="#" id="clickableAnchor">点这里</a>
```

这就使三个层次彻底分开了，最终实现了关注点分离。今后就很容易在多个页面共享这些文件，通过缓存获得更好的性能。在前端领域，层次关系类似这样：

- HTML，数据层；
- CSS，表现层；
- JavaScript，行为（逻辑）层。

我们再来多看一点细节。

① 本例使用了 jQuery 来简化代码，并不代表在众多 JavaScript 框架中我们更推崇 jQuery。

此为试读，需要完整PDF请访问：www.ertongbook.com

数据层

对于网站而言，我们一般不称“数据”而称“内容”。经常听人说“内容为王”，这是不错的建议。应该使用语义相关的标记，并将其编排得结构良好，使得网页文件本身就是一个可供阅读、内容有效、顺序合理的文件。这样就可以提供机会，让机器与人类一样不仅了解文档内容的意图，甚至明白字里行间的意思。如果某段内容要作为标题，就应该标记为标题，而不是标为粗体段落之类的。如果很明显是项目列表，就应该标记为列表，而不是一系列 div 或其他相关性更差的元素。当机器理解了我们的意图，就有能力做到以下几点。

- 明确网页的用途，哪块重要，哪块不重要，这对搜索引擎和辅助设备尤其重要。
- 在不影响可读性的前提下改变页面的显示方式，例如用户可以定制样式表。这也意味着网站可以良好地向下兼容旧版浏览器，即使网页显示有所不同也不会妨碍用户对内容的理解。
- 允许非线性跳读内容，这样又可以让辅助设备从中获益。

表现层

在这一层只考虑页面的外观，涉及外表的、纯粹装饰的部分全放在这里。如要选择字体、页面布局或改变颜色，这些适合放在表现层。如果公司的品牌形象要求把内容改成小写字母，或者把标题缩小，你就可以在不触及页面“真实”内容语义的情况下通过修改 CSS 来实现。

行为层

行为层有时也称为逻辑层、业务层或业务逻辑层。当要在页面中实现元素的拖放或 AJAX（异步 JavaScript 与 XML 技术，一种不需要刷新整个页面就能从服务器取得信息的方法）等非标准的交互方式时，就要使用 JavaScript 来完成这部分的工作。

前端开发不拘一格

虽然分层模式的好处显而易见，但实际使用中很难像我们刚刚看到的那样划分得如此泾渭分明。CSS 文件里可以包括运算表达式（但我们不提倡这样做），JavaScript 可以完成表现工作（例如菜单的外观和动画），有时 JavaScript 还要负责为页面添加新的内容，诸如此类，不胜枚举。尽管这样，分层的思路还是应该作为一个基本方向。此外，以分层模式来进行 Web 开发往往能引出种种奇思妙想，同时还能避免内容重复和保持较高的性能。

了解了这些背景知识，我们就可以上路了。第 1 章将介绍流程的重要性，这虽然不算严格意义上的 CSS 话题，却是构建项目的重要基础。

目　录

第1章 流程的价值

本章我们集中讨论团队和公司的流程如何才能促进团队效率，而不是束缚生产力。其中很多话题并不是专门针对CSS（层叠样式表）的，但如果要写出容易扩展、能够跨团队、跨网站共用的CSS，拥有一致、可靠的流程和标准至关重要。本章将会探讨开发方法、代码风格一致性、源代码管理、开发工具以及命名约定等。在本书的最后，有一份示例流程，可供实践参照。

本章我们着眼于：

- 团队组成；
- 业务扩张；
- 员工流失；
- 代码一致性；
- 开发工具；
- 版本控制；
- 备份；
- 原型设计；
- 开发方法。

1.1 团队

在CSS开发中，可以认为最重要的因素就是程序员身处的工作团队。不单是成员性格、人数多少，还包括管理制度、技术水平，每个团队都各有千秋。有的团队由几名服务器端程序员和一名前端程序员组成，有的团队则囊括了设计师、前端程序员、服务器端程序员、数据库工程师、测试人员，几乎涵盖Web开发相关的所有环节。

这样的大团队优势是不同专业领域沟通便利，但人多也有坏处：人越多对需求的响应就越慢。原因多种多样，研究人员将其中的主要因素称为“责任分散”。意思是说，在较大的团队中，针对某个特定时间的特定任务，大家都会以为别人正在做这事儿，而不会自己承担责任。一旦出现意想不到的问题，或者计划外的工作任务，如果缺乏明确适用的准则，就很难确定谁应当负责。结果是，某些事情即使提了出来，也会被抛在脑后。在较小的团队中，沟通是立竿见影的，个人的表现也会更受关注，人们会更加主动地承担责任。

此为试读，需要完整PDF请访问：www.ertongbook.com