CSS Mastery

Advanced Web Standards Solutions, Third Edition

精通CSS

高级Web标准解决方案

（第3版）

[英] 安迪·巴德 [瑞典] 埃米尔·比约克隆德 著

李松峰 译

- CSS畅销经典全面升级，充分展示现代CSS实践技巧
- 直接提供常见问题的解决方案，让前端架构更上一层楼

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

作者简介

安迪·巴德
(Andy Budd)

数字设计咨询公司Clearleft联合创始人，Web标准倡导者。目前专注于用户体验设计领域。

埃米尔·比约克隆德
(Emil Björklund)

数字设计咨询公司inUse技术总监，拥有十余年专业Web开发经验，从客户端JavaScript到服务器端Python都有深入研究。

译者简介

李松峰

360奇舞团高级前端开发工程师、前端TC委员、W3C AC代表，360 Web字体服务“奇字库”作者，翻译出版过40余部技术及交互设计专著，如《JavaScript高级程序设计》《简约至上》等。

数字版权声明

TURING 图灵程序设计丛书

CSS Mastery
Advanced Web Standards Solutions, Third Edition

精通CSS
高级Web标准解决方案
（第3版）

[英] 安迪·巴德 [瑞典] 埃米尔·比约克隆德 著
李松峰 译

人民邮电出版社
北 京

图书在版编目（CIP）数据

精通CSS : 高级Web标准解决方案 : 第3版 / (英)安迪·巴德 (Andy Budd) , (瑞典) 埃米尔·比约克隆德 (Emil Björklund) 著 ; 李松峰译. -- 北京 : 人民邮电出版社, 2019.2
（图灵程序设计丛书）
ISBN 978-7-115-50690-0

Ⅰ. ①精… Ⅱ. ①安… ②埃… ③李… Ⅲ. ①网页制作工具 Ⅳ. ①TP393.092.2

中国版本图书馆CIP数据核字(2019)第020503号

内 容 提 要

本书是 CSS 设计经典图书升级版，结合 CSS 近年来的发展，尤其是 CSS3 和 HTML5 的特性，对内容进行了全面改写。本书介绍了涉及字体、网页布局、响应式 Web 设计、表单、动画等方面的实用技巧，并讨论了如何实现稳健、灵活、无障碍访问的 Web 设计，以及在技术层面如何实现跨浏览器方案和后备方案。本书还介绍了一些鲜为人知的高级技巧，让你的 Web 设计脱颖而出。

本书适合具备 HTML 和 CSS 基础知识的读者阅读。

◆ 著　[英] 安迪·巴德
　　[瑞典] 埃米尔·比约克隆德
　译　李松峰
　责任编辑　温　雪
　责任印制　周昇亮

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路11号
　邮编　100164　电子邮件　315@ptpress.com.cn
　网址　http://www.ptpress.com.cn
　北京　印刷

◆ 开本：800×1000　1/16
　印张：24.25
　字数：573千字　2019年 2 月第 1 版
　印数：1－4 000册　2019年 2 月北京第 1 次印刷

著作权合同登记号　图字：01-2016-8299号

定价：99.00元

读者服务热线：(010)51095186转600　印装质量热线：(010)81055316

反盗版热线：(010)81055315

广告经营许可证：京东工商广登字 20170147 号

版 权 声 明

本书献给我在 Clearleft 的同事——过去的，还有现在的。没有他们的支持和智慧，就没有这本书。

——安迪·巴德（Andy Budd）

献给我最怀念的祖父 Sven Forsberg（1919—2016），一位工程师、艺术家、终身手艺人。

——埃米尔·比约克隆德（Emil Björklund）

前　言

回想 2004 年，在写本书第 1 版的时候，市面上已经有两本 CSS 方面的书了。当时，我对读者是否需要第三本并没有把握。毕竟，CSS 那时候还算小众技术，只有博主和 Web 标准的狂热粉丝才会研究。当时的大部分网站用的还是表格和框架。本地开发者邮件列表中的小伙伴都说我疯了，他们认为 CSS 只是一个美丽的梦。他们并没有想到，那时候 Web 标准运动的序幕即将拉开，而本书的出版恰逢这个领域爆发之际。在接下来的几年里，本书一直名列出版商最畅销图书榜单。

等到本书第 2 版出来的时候，CSS 的地位已经无法撼动。本书的作用也从向人们展示 CSS 的魅力，转变成帮人们更有效地使用 CSS。于是，我们找到各种新技术、解决方案，还有“黑科技”，希望打造出一本 Web 设计师和前端开发者的权威指南。当时 CSS 的发展似乎已经趋于稳定，而本书好像也能卖相当长一段时间。事实证明，我们错了。

CSS 的发展并未停滞，近几年的情况表明，CSS 最终兑现了其最初的承诺。我们进入了 Web 标准的黄金时代，即浏览器支持程度已经足够好的时代。因此，我们终于可以放弃原来那些“黑科技”，转而把时间和精力花在为最大、最复杂的网站编写优雅、巧妙、容易维护的代码上。

于是，本书第 3 版应运而生：该把所有新工具、新技术和新思路写成一本新书了。为了完成这个任务，我把好朋友埃米尔·比约克隆德拉了进来。他是一位技术与才能俱佳的开发高手，为本书加入了对现代 CSS 实践的深刻理解，还会告诉大家怎么利用新技术写出高度灵活的代码，并且让这些代码以最优雅的方式在不同浏览器、不同屏幕和不同平台上跑起来。

应该说，我们俩通力合作，基本上完全重写了本书，并且添加了覆盖 Web 排版、动画、布局、响应式设计、组织代码等主题的新章节。这一版仍然继承了前两版的写法，整合了实例、语言解读和跨浏览器的巧妙解决方案。谙熟各路“黑科技”或者任意属性都能信手拈来，这些不再是精通 CSS 的标志。今天的 CSS 已经分化为几十个规范，演化出了几百个属性，恐怕没有谁能够对其无所不知！因此，这一版不追求让读者对 CSS 无所不知，而是强调灵活性、稳健性，并确保代码在花样不断翻新的浏览器、设备和使用场景下都能欢快地跑起来。虽然本书不会一一介绍所有语言特性，但会让你知道有什么可用，告诉你一些鲜为人知的基本技术，还有对 CSS 未来的展望。

要想真正看懂这本书，读者至少应该懂得 CSS 的基本原理，比如自己写过 CSS，甚至用它

设计过一两个网站。本书前三章是科普性质的，讲了一些给网页添加样式的最基础的知识，也算是照顾一下基础不牢的读者吧。从第 4 章开始，每一章都会介绍不同的 CSS 新特性，给出的例子也会越来越复杂。相信即使是 CSS 的老手，也能从本书中学到解决常见问题的实用技术。当然，这样的读者就不用按部就班地从头看到尾了，感觉哪一章有意思就看哪一章吧。

最后，我们希望无论读者的基础如何、经验多寡，都能够借助本书领略到 CSS 的无穷魅力，最终成为真正精通 CSS 的大师。

本书源代码地址：https://github.com/Apress/css-mastery-16。[①]

① 读者可前往本书图灵社区页面（http://www.ituring.com.cn/book/1910）下载源代码，查看本书更多信息，并提交中文版勘误。——编者注

致　　谢

我们要感谢 Jeffrey Zeldman、Eric Meyer 和 Tantek Çelik 的不懈努力，如果没有他们，“Web 标准运动”就永远不会发生。我们还要感谢后来参与这场运动的人，比如 John Allsopp、Rachel Andrew、Mark Boulton、Doug Bowman、Dan Cederholm、Andy Clarke、Simon Collison、Jon Hicks、Molly E. Holzschlag、Aaron Gustafson、Shaun Inman、Jeremy Keith、Peter-Paul Koch、Ethan Marcotte、Drew McLellan、Cameron Moll、Dave Shea、Nicole Sullivan 和 Jason Santa-Maria，他们迎接挑战并齐心协力将 CSS 变成今天的主流。最后，我们要感谢所有坚持不懈的设计师和开发者，他们接过了接力棒，并让 CSS 成为我们今天所知道的现代设计语言。虽然难免挂一漏万，但这里还是要提几位近年来对我们的实践产生了极大影响的人，包括 Chris Coyier、Vasilis van Gemert、Stephen Hay、Val Head、Paul Lewis、Rachel Nabors、Harry Roberts、Lea Verou、Ryan Seddon、Jen Simmons、Sara Soueidan、Trent Walton 和 Estelle Weyl。我们还要感谢那些在 Twitter 和 Slack 小组中给过我们帮助和启发的设计师和开发者。

我们要感谢帮助这本书冲过终点的每一个人，包括为写作本书提供赞助的 inUse 公司。特别感谢技术编辑 Anna Debenham，如果书中存在任何错误，那么一定是我们没搞好，并且导致她没有发现。我们还要感谢 Andy Hume，他在本书开始写作时提供了专业的意见，为这个新版本确定了方向。此外，我们要感谢 Charlotte Jackson、Peter-Paul Koch、Paul Lloyd、Mark Perkins 和 Richard Rutter，感谢他们审校本书草稿、贡献想法，并提供宝贵的反馈。

书中或示例中的照片多数是我们自己制作或者从公共领域搜集的。以下图片获得了 Creative Commons Attribution 2.0 许可授权：*Portrait*，由 Jeremy Keith 提供；*A Long Night Falls on Saturn's Rings*，由美国国家航空和航天局戈达德太空飞行中心提供。

最后，我们俩都想感谢各自的伴侣，写那么多页花了太长时间，感谢她们的耐心和支持。

目　录

第 1 章 基础知识

人类天生好奇，喜欢摆弄东西。那天在办公室收到了新的遥控四轴飞行器 Parrot AR Drone，我们都没看说明书，就开始动手组装起来。我们喜欢自己琢磨，喜欢按照自己认为的事物运作方式来思考和解决问题。零件只要能拼上就行，除非拼不上了，或者事实跟我们的想法相背离，才不得已去翻翻安装指南。

学习层叠样式表（CSS，cascading style sheets）最好的方法也一样，就是不管三七二十一，直接上手写代码。事实上，可能很多人都是这么学习编程的。比如在某个博客上看到了一些建议，又比如通过开源代码库研究自己喜欢的设计师创造的某个特效的源代码。几乎没人是在完整地看了一遍规范全文之后才开始动手写代码的，因为那样的话，人也早睡着了。

自己动手写代码是最好的开始方式，只是如果不够细心，可能会误解某个重要概念，或者给以后写代码埋下隐患。我们对此深有体会，因为自己给自己挖坑的事儿已经干了不是一回两回了。因此，这一章就来回顾一些基础但容易误解的概念，讲一讲怎么让 HTML 和 CSS 保持组织分明且结构良好。

本章内容：

- 可维护性的重要意义
- HTML 和 CSS 的不同版本
- 未来友好的代码与向后兼容的代码
- 使用新的 HTML5 元素为 HTML 赋予意义
- 在 HTML 中为添加样式设置恰当的接入点
- 通过 ARIA、微格式、微数据扩展 HTML 的语义
- 浏览器引擎模式与验证

1.1 组织代码

人们通常不会注意到建筑物的基础。可是，没有坚实的基础，建筑物的主体就不可能稳固。虽然本书讲的是 CSS 的技术和概念，但这些内容的讲解与实现必须以结构良好且有效的 HTML 文档为前提。

本节会介绍结构良好且有意义的 HTML 文档对基于标准的 Web 开发的重要性，以及如何让文档更有意义、更灵活，从而减轻开发者的工作负担。但首先我们要讲一个对任何语言都同样重要的概念。

1.1.1　可维护性

可维护性可以说是所有优秀代码最重要的特点。如果你的代码已经看不出结构，变得难以阅读，那么很多问题就会接踵而至。开发新功能、修复 bug、提升性能，所有这些操作都会因为代码可读性差以及代码脆弱而变得复杂，而且结果难料。这最终会导致开发人员一点代码也不敢改，因为每次只要改一点，就会出问题。于是，没人愿意再维护这个网站，更糟糕的情况是只能走严格的变更控制流程，每周发布一次，甚至每月才能发布一次！

如果你开发的网站最终要交付给客户或者另一个开发团队，那么可维护性就更重要了。这时候，代码是否容易看懂，是否意图明确，是否**为将来的修改做过优化**，都至关重要。哪个项目没有持续不断的变更需求？哪个项目不需要一直开发新功能？哪个项目不需要不断修复 bug？因此，“唯一不变的就是变化”。

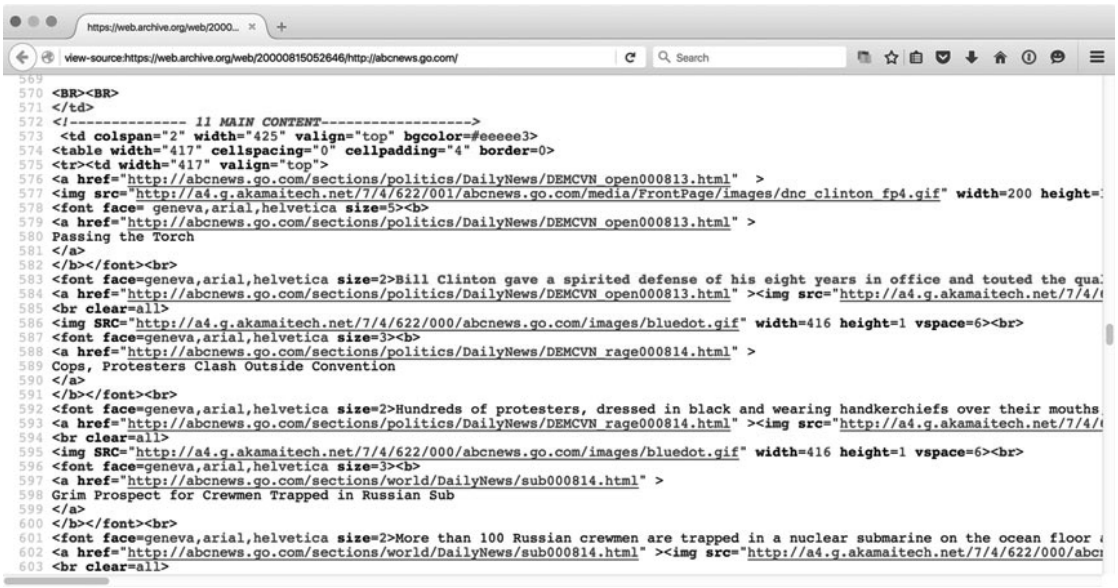
相对来说，CSS 是随着代码量增加而最难保持可维护性的语言之一。即使网站规模不大，样式表也会很快变得难以控制。现代编程语言都有内置的变量、函数和命名空间等特性，这些特性都有利于保持代码的结构和模块化。这些特性 CSS 都没有，所以要按照使用这种语言和组织代码的特殊方式来管理它。本书后面在讨论各种主题时，大都会涉及可维护性。

1.1.2　HTML 简史

> Web 的威力源自其普适性。即使残障用户也能使用是题中应有之义。
>
> ——Tim Berners-Lee

Tim Berners-Lee 在 1990 年发明了 HTML，当时是为了规范科研文档的格式。HTML 是一种简单的标记语言，为文本赋予了基本的结构和意义，比如标题、列表、定义等。这些文档通常没有什么装饰性的元素，可以方便地通过计算机来检索，而人类可以使用文本终端、Web 浏览器，或者必要时使用屏幕阅读器来阅读它们。

然而，人类是视觉发达的生物。随着万维网被越来越多的人所接受，HTML 也逐渐增加了对展示效果的支持。除了用标题元素标记文档标题，还可以使用粗体标签和不同的字体来创建特殊的视觉效果。本来用于展示数据的表格（`table`），却成了页面布局的手段；块引用（`blockquote`）也经常被用来缩进文本，而不是只用来标记引文。HTML 很快就偏离了为内容赋予结构和意义的初衷，变成了一堆字体和表格标签。Web 设计者给这种标记起了个名字，叫“标签汤”（见图 1-1）。



```
<BR><BR>
</td>
<!-------------- 11 MAIN CONTENT------------------>
 <td colspan="2" width="425" valign="top" bgcolor=#eeeee3>
<table width="417" cellspacing="0" cellpadding="4" border=0>
<tr><td width="417" valign="top">
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_open000813.html"  >
<img src="http://a4.g.akamaitech.net/7/4/622/001/abcnews.go.com/media/FrontPage/images/dnc_clinton_fp4.gif" width=200 height=
<font face= geneva,arial,helvetica size=5><b>
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_open000813.html" >
Passing the Torch
</a>
</b></font><br>
<font face=geneva,arial,helvetica size=2>Bill Clinton gave a spirited defense of his eight years in office and touted the qua
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_open000813.html" ><img src="http://a4.g.akamaitech.net/7/4/
<br clear=all>
<img SRC="http://a4.g.akamaitech.net/7/4/622/000/abcnews.go.com/images/bluedot.gif" width=416 height=1 vspace=6><br>
<font face=geneva,arial,helvetica size=3><b>
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_rage000814.html" >
Cops, Protesters Clash Outside Convention
</a>
</b></font><br>
<font face=geneva,arial,helvetica size=2>Hundreds of protesters, dressed in black and wearing handkerchiefs over their mouths
<a href="http://abcnews.go.com/sections/politics/DailyNews/DEMCVN_rage000814.html" ><img src="http://a4.g.akamaitech.net/7/4/
<br clear=all>
<img SRC="http://a4.g.akamaitech.net/7/4/622/000/abcnews.go.com/images/bluedot.gif" width=416 height=1 vspace=6><br>
<font face=geneva,arial,helvetica size=3><b>
<a href="http://abcnews.go.com/sections/world/DailyNews/sub000814.html" >
Grim Prospect for Crewmen Trapped in Russian Sub
</a>
</b></font><br>
<font face=geneva,arial,helvetica size=2>More than 100 Russian crewmen are trapped in a nuclear submarine on the ocean floor
<a href="http://abcnews.go.com/sections/world/DailyNews/sub000814.html" ><img src="http://a4.g.akamaitech.net/7/4/622/000/abcn
<br clear=all>
```

图 1-1 ABC News 网站 2000 年 8 月 14 日头条新闻的标记，使用了表格布局，标题为大号粗体文本；代码没有结构，难以理解

正当 Web 变得一团糟之际，CSS 作为解决方案面世了。CSS 的初衷是把跟 HTML 混在一起的表现性标记提取出来，使其自成体系，达到**结构与表现分离**的目的。这就让有意义的标签或者说语义悄悄返回了 HTML 文档。`font` 之类的表现性标签可以不用了，而表格布局也可以被逐步取代。对大多数网站而言，CSS 都能提升其可访问性和加载速度。不仅如此，CSS 还给 Web 设计和开发人员带来了更多好处：

- 一种专用于控制视觉样式和布局的语言；
- 在同一网站中更易于重用的样式；
- 通过关注点分离得到了良好的代码结构。

关注点分离

关注点分离（separation of concerns）是软件开发行业的一个常见概念。对Web开发而言，关注点分离不仅适用于标记和样式，同样也适用于编写样式的方式。事实上，关注点分离也是确保代码可维护性的一种主要方法。

Unix开发社区有一句话很好地诠释了关注点分离的思想，即“分成小块，松散结合”（small pieces, loosely joined）。其中，每一“小块”都是一个模块，专注于做好一件事。而且，因为这个模块跟其他组件是“松散结合”的，所以它可以方便地在系统的其他部分中重用。Unix中的一“小块”可能是一个字数统计函数，可以应用于传入的任何文本片段。而在Web开发中，这一“小块”可能就是一个商品列表组件，如果能做到“松散结合”，就可以

在一个网站的多个页面或者在同一布局的不同区块中重用。

可以把代码中的这些“小块”想象成积木。每一块积木都很简单，但把很多块积木以不同方式组装起来，就可以创造出无比复杂的东西。第12章还会讨论这个话题，届时会讲一讲怎样以结构化的方式应用这个策略。

1. HTML 和 CSS 的版本

CSS 有很多版本，或者“级别”。了解这些版本产生的背景，有助于了解应该或不应该使用哪些 CSS 特性。万维网联盟（W3C，World Wide Web Consortium）是制定 Web 技术标准的组织，该组织制定的每一个规范要经历几个阶段之后才能成为 W3C 推荐标准。CSS1 是在 1996 年底成为 W3C 推荐标准的，当时只包含字体、颜色和外边距等基本的属性。CSS2 在 1998 年成为推荐标准，增加了浮动和定位等高级特性，此外还有子选择符、相邻选择符和通用选择符等新选择符。

相比之下，CSS3 则采用了完全不同的模式。实际上**不存在**所谓的 CSS3 规范，因为 CSS3 指的是一系列**级别**独立的模块。如果规范模块是对之前 CSS 概念的改进，那就从 3 级（level 3）开始命名。如果不是改进，而是一种全新的技术，那就从 1 级（level 1）开始命名。而我们所提到的 CSS3，则是指所有足够新的 CSS 规范模块，比如 CSS Backgrounds and Borders Level 3、Selectors Level 4 和 CSS Grid Layout Level 1。这种模块化的方式可以让不同的规范有自己的演进速度。有些 3 级规范，比如 CSS Color Level 3，已经成为推荐标准。而另外一些可能还处于候选推荐阶段，很多甚至还处于工作草案阶段。

虽然 CSS3 的制定工作在 CSS2 发布后就开始了，但这些新规范一开始的制定速度很缓慢。为此，W3C 在 2002 年发布了 CSS2 Revision 1。CSS 2.1 修正了 CSS2 中的一些错误，删掉了支持度不高或者并非所有浏览器都实现了的一些特性，总体来说就是把 CSS 规范做了一番清理，好为浏览器实现提供更精准的蓝图。CSS 2.1 在 2011 年 6 月成为推荐标准，此时距离 CSS3 启动已经有 10 多年了。由此可见，标准制定主体和浏览器开发商为了确保相应的特性得以原原本本地实现，需要花多么长的时间。不过，浏览器开发商经常会在标准还处于草案阶段时，就发布一些实验性的实现。这样，等到了候选推荐阶段，相应的实现就已经非常稳定了。换句话说，很多 CSS 特性早在相应模块成为推荐标准前就可以使用了。

HTML 的历史也很复杂。HTML 4.01 在 1999 年成为推荐标准，与此同时 W3C 也把注意力转向了 XHTML 1.0。本来接着要发布 XHTML 1.1，但其严格程度在实践中暴露了无法落地的问题，最终被 Web 开发社区抛弃。于是，这个 Web 主要语言的发展停滞了。

2004 年，有几家公司共同组建了 Web 超文本应用技术工作组（WHATWG，Web Hypertext Application Technology Working Group），并致力于开发新的规范。2006 年，W3C 肯定了它们工作的必要性，并欣然加入该工作组。2009 年，W3C 完全放弃 XHTML，正式接纳 WHATWG 制定的新标准，这就是后来的 HTML5。起初，WHATWG 和 W3C 都基于标准调整自己的工作，但后来它们的关系又变得复杂起来。今天，它们分别在编辑两份标准。WHATWG 那份就叫 HTML，

而 W3C 那份则称为 HTML5。没错，这种分裂确实不好。但万幸的是，这两份标准的内容相当接近，因此只讲 HTML5 是没有问题的。

2. 应该使用哪个版本

设计者和开发者经常问的一个问题就是应该使用 HTML 或 CSS 的哪个版本。这个问题不好回答。虽然规范反映了标准和 Web 技术开发的进度和焦点，但它其实跟设计者和开发者日常的工作关系不大。真正重要的是知道 HTML 和 CSS 的哪些部分已经在浏览器中实现了，以及这些实现是否稳健，有没有 bug。比如，浏览器提供的这些特性是不是实验性特性，使用时需谨慎？或者，这些特性到底靠不靠谱，是不是已经得到了多数浏览器的支持？

今天，使用 CSS 和 HTML 就要了解浏览器对其中特性的支持程度。有时候我们会觉得技术发展太快，必须拼命追赶才不会落伍；有时候我们又会觉得技术发展太慢。本书会随时提示不同 HTML 和 CSS 特性的浏览器支持情况，还会给出在什么情况下可以使用它们的建议。不过显而易见的是，印在纸上的东西注定会过时，所以你得学会自己更新这方面的信息。

要了解浏览器支持情况，推荐几个不错的地方。对于 CSS 属性，可以访问“Can I use”网站（https://caniuse.com）。这个网站可以搜索属性或属性组，结果配有统计信息，显示支持它们的浏览器百分比，包括桌面浏览器和移动浏览器。另一个非常有想法的项目是 https://webplatform.github.io/，是 W3C 和几家浏览器厂商及行业巨头合作搞出来的，目标是收集合并它们所有关于 CSS、HTML、JavaScript API 等支持情况的文档。不过，就跟很多大型项目一样，最终要完成那么庞大的 Web 技术文档的聚合，需要花很长时间。此外，Mozilla 的开发者文档，即 MDN，也是一个非常好的参考。

说到浏览器支持，关键要明白，并非所有浏览器都一样，实际上从来就没有完全一样的浏览器。某些 CSS3 特性只得到了少数浏览器的支持。比如，Internet Explorer 11 和 Safari 6.1 之前的版本都没有正确支持 Flexbox（Flexible Box Layout）。不过，就算需要支持老版本的浏览器，也不意味着不能使用 Flexbox。核心布局上可以不用 Flexbox，但对于某些特定的组件，Flexbox 可能就非常合适。只要为不支持它的浏览器准备好可以接受的后备代码就行了。判断一个人是不是 CSS 大师，很大程度上要看他能否游刃有余地处理向后兼容的代码与未来友好的代码。

1.1.3 渐进增强

平衡向后兼容性与最新的 HTML 和 CSS 特性，涉及一种叫作**渐进增强**（progressive enhancement）的策略。所谓渐进增强，大意就是“首先为最小公分母准备可用的内容，然后再为支持新特性的浏览器添加更多交互优化”。使用渐进增强策略，意味着代码要分层，每一层增强代码都只会在相应特性被支持或被认为适当的情况下应用。听起来有点复杂，而实际上 HTML 和 CSS 的实现已经部分内置了这一策略。

对 HTML 而言，这意味着浏览器在遇到未知元素或属性时并不会报错，而且也不会对页面产生什么影响。比如，可以在页面里使用 HTML5 定义的新 `input` 元素。假设表单中有一个电子

邮件字段的标记如下：

```
<input type="text" id="field-email" name="field-email">
```

要使用新的 input 元素，应该把 type 属性改成这样：

```
<input type="email" id="field-email" name="field-email">
```

尚未实现这个新字段类型的浏览器碰到它只会想：“这是啥意思呀？不明白。”然后回退为默认的 text 类型，结果和上面的第一行代码一样。而实现了这个类型的新浏览器则知道 email 想让用户在这里填写什么样的数据。而在很多移动设备中，相应的软键盘还会针对输入电子邮件地址调整界面布局。假如你还在这里使用了内置的表单验证，那么支持它的新浏览器也会帮你做验证。这样，我们既**渐进增强**了页面，也不会对旧版本浏览器产生不好的影响。

另外一个简单的变化就是，HTML5 把文档类型声明更新为新的简短形式。所谓文档类型，就是位于 HTML 文档第一行的代码，供机器识别当前文档使用的标记语言版本。以往的 HTML 和 XHTML 版本中，这行代码很长很复杂，但在 HTML5 中，它已经简化成了下面这样：

```
<!DOCTYPE html>
```

今后，只要这样声明 HTML 文档类型就好了，因为这个 HTML5 语法的 doctype 是向后兼容的。后面几节我们会再介绍一些 HTML5 中出现的新元素，但如果要更深入地学习如何使用 HTML5 标记，建议看一看 Jeremy Keith 的 *HTML5 for Web Designers*。

CSS 中的渐进增强同样也反映在浏览器如何对待新属性上。任何浏览器无法识别的属性或值都会导致浏览器丢弃相应的声明。因此，只要同时提供合理的后备声明，使用新属性就不会带来不良后果。

举个例子，很多现代浏览器支持以 rgba 函数方式表示的颜色值。这种方式可以分别传入红、绿、蓝通道，以及阿尔法（alpha，即透明度）通道的值。我们可以这样使用它：

```
.overlay {
  background-color: #000;
  background-color: rgba(0, 0, 0, 0.8);
}
```

这条规则定义了类名为 overlay 的元素背景为黑色，但随后又用 rgba 声明背景色应稍微透明。如果浏览器不支持 rgba，那么相应元素的背景色就是不透明的黑色。如果浏览器支持 rgba，那么第二条声明就会覆盖第一条。也就是说，即使并非所有浏览器都支持 rgba，我们也可以使用它，只是要先为它声明合适的后备代码。

1. 厂商前缀

浏览器厂商也基于相同的原理为自家浏览器引入实验性特性。实验性特性的标准名称前面会加上一个特殊字符串，这样他们自己的浏览器就能识别该特性，而其他浏览器则会忽略该特性。有了这个方案，浏览器厂商就可以添加规范中没有或者尚不成熟的新特性，样式表作者也可以安心地试用这些新属性，不用担心浏览器因不认识它们而破坏页面。比如：

```
.myThing {
  -webkit-transform: translate(0, 10px);
  -moz-transform: translate(0, 10px);
  -ms-transform: translate(0, 10px);
  transform: translate(0, 10px);
}
```

这里使用了几个不同的前缀，给相应的元素应用了变换（第 10 章会介绍）。以`-webkit-`开头的适用于基于 WebKit 的浏览器，如 Safari。Chrome 和 Opera 都基于 Blink 引擎，而 Blink 最初也是基于 WebKit 开发的，所以`-webkit-`前缀通常也适用于这 3 个浏览器。`-moz-`前缀适用于基于 Mozilla 的浏览器，如 Firefox。`-ms-`前缀则适用于微软的 Internet Explorer。

最后我们又加了一条不带前缀的声明，这样那些支持标准属性名称的浏览器就不会漏网了。过去经常出现开发人员漏加不带前缀的标准声明的情况。为此，有些浏览器厂商也开始支持竞争对手引擎特定的前缀，以便让流行的网站能在自己的浏览器上打开。但这样做也造成了混乱，于是多数浏览器厂商抛弃了厂商前缀。那实验性特性呢？有的厂商选择把它们隐藏在 chrome://flags 中，有的选择只在特定的预览版中提供。

本书中绝大多数示例都使用不带前缀的标准属性名称，因此建议大家经常查一查 http://caniuse.com，以了解相应的支持情况。

2. 条件规则与检测脚本

如果希望根据浏览器是否支持某个 CSS 特性来提供完全不同的样式，那么可以选择`@supports`块。这个特殊的代码块称为条件规则，它会检测括号中的声明，并且仅在浏览器支持该声明的情况下，才会应用块中的规则：

```
@supports (display: grid) {
  /* 在支持网格布局的浏览器中要应用的规则 */
}
```

条件规则的问题是其自身也很新，只能将它应用于新的浏览器中，因为旧版本浏览器不支持（比如第 7 章要介绍的网格布局）。此外，还可以通过 JavaScript 来检测支持情况，比如使用 Modernizr 这个库。Modernizr 的原理是为 HTML 添加支持提示信息，然后可以依据这些信息来编写 CSS。

随后几章还会更详细地介绍类似的策略和工具，这里关键是要知道：渐进增强可以让我们放下对版本号和规范的很多担忧。只要加点小心，就可以在适当的时候使用一些簇新的特性，同时又不会丢掉使用旧版浏览器的用户。

1.2 创建结构化、语义化富 HTML

语义化标记是优秀 HTML 文档的基础。**语义**就是以系统方式表示的含义。对于根据一个形式符号的集合人工创造出的语言（比如 HTML 语言，及其元素和属性）来说，语义指的就是通过使用某个符号想要表示的含义。简而言之，**语义化标记**意味着在正确的地方使用正确的元素，

从而得到有意义的文档。

有意义的文档可以确保尽可能多的人都能够使用，无论他们用的是最新版本的 Chrome，还是 Lynx 这样只能处理文本的浏览器，甚至是屏幕阅读器或盲文点触设备之类的辅助技术。无论将来项目中会增加多么花哨的图形或交互，文档的基础语义都应该永远——而且必须永远——不打折扣。

结构良好的标记也意味着内容更对机器的胃口。机器？对，特别是 Googlebot 这种搜索引擎爬虫，对它胃口的内容可以让你的页面在 Google 搜索结果中排名更靠前。这是因为，Googlebot 从你的页面中获得的相关数据越多，它对你的页面的索引和排名可能就越准确。于是，你的页面在搜索结果中出现的位置就可能更靠前。

对于 CSS 来说更重要的是，有意义的标记本身为添加样式提供了方便。这些标记不仅描述了文档的结构，而且还为我们继续装扮它提供了底层的框架。

实际上，编写 CSS 的最新实践都建议先给网站一组“基础”样式。图 1-2 是 Paul Lloyd 的样式指南，包括他个人博客中可能用到的所有元素的样式说明。每个元素都有使用的方式和情境。他的样式表可以确保，无论以后他给页面添加什么元素，都无须再另写样式。

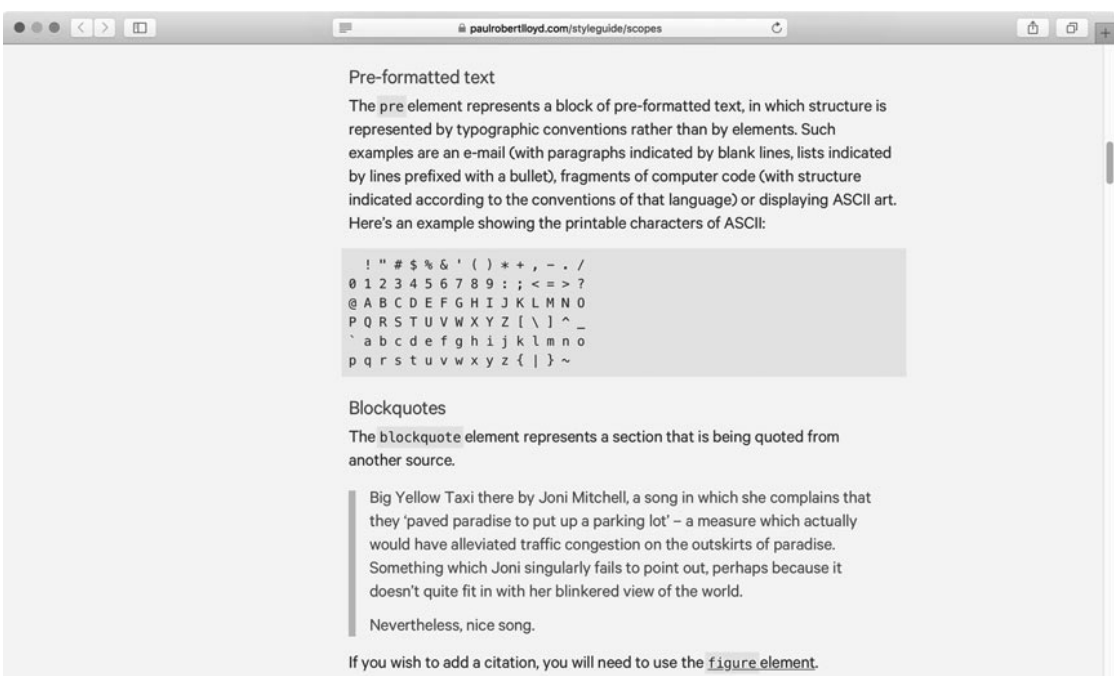


图 1-2　Paul Lloyd 网站的样式指南

Paul 的样式指南包含所有语义明确的元素，比如：

- h1、h2 等
- p、ul、ol 和 dl
- strong 和 em
- blockquote 和 cite
- pre 和 code
- time、figcaption 和 caption

其中还包括表单、表格及其相关元素的基础样式，比如：

- fieldset、legend 和 label
- caption、thead、tbody 和 tfoot

设立这么一套基础样式的价值非常之大。虽然实际设计和开发中，它们很快会被继承和覆盖，但有了这么一套基础样式，将来的工作就会有条不紊。这套样式也可以作为校准样式来使用。在不断修改 CSS 的过程中，可以时不时对照一下样式指南中的组件，检查自己是否无意中覆盖了某些不该覆盖的样式。

1.2.1 ID 和 class 属性

有意义的元素提供了不错的基础，却没有提供应用视觉效果所必需的全部“接入点”。我们几乎总是要根据上下文来调整基础元素的样式。除了元素本身，我们还需要一种方式把样式“接入”到文档上，这就是 ID 和 class 属性。

为元素添加 ID 和 class 属性**不一定**能给文档增加含义或结构。这两个属性只是一种让其他因素来操作与解析文档的通用手段，CSS 也可以利用这一手段。我们可以设置这些属性的值，即为其起名字。

给属性起名字听起来简单，但在写代码时却是极其重要的（常常也是最难的）。“名不正则言不顺”，起什么名字意味着它**是**什么，或者应该怎么使用它。我们知道，写代码的时候，清晰和明确都是至关重要的原则。下面就以一个链接列表为例，看看怎么给它的 class 属性一个既容易辨识又好用的值：

```
<ul class="product-list">
  <li><a href="/product/1">Product 1</a></li>
  <li><a href="/product/2">Product 2</a></li>
  <li><a href="/product/3">Product 3</a></li>
</ul>
```

我们先利用 class 属性在文档中创造一个 product-list 模块。在 CSS 里，我们用类名来定义一类事物。这里的 product-list 就意味着它可以是任何商品列表。换句话说，为 product-list 写好样式后，不仅可以用在这里，还可以用在网站的其他地方，就像蓝图或者模板一样可以重用。

给元素添加类名时，即使类名明确用于样式，也不要体现出其视觉效果（第 12 章会详细讨论这一点，包括什么情况下类名可以体现视觉效果）。正确的做法是让类名表示组件的类型。比

如，这里的类名是 product-list，而非泛泛的 large-centered-list。

前面的例子只给元素添加了 class 属性，并没有添加 ID 属性。对于添加样式而言，ID 与 class 属性有一些重要的区别，但针对这个例子而言，最主要的区别是一个 ID 只能应用到页面中的一个元素。也就是说，不能像 product-list 那样使用 ID 把页面中的模块定义为可重用的“模板”。如果使用了 ID，那么相应的 product-list 在每个页面中只能出现一次。

本书提倡使用 ID 来标识特定模块的**特定实例**。比如，下面就是 product-list 模块的一个特定实例：

```
<ul id="primary-product-list" class="product-list">
  <li><a href="/product/1">Product 1</a></li>
  <li><a href="/product/2">Product 2</a></li>
  <li><a href="/product/3">Product 3</a></li>
</ul>
```

这是 product-list 的另外一个实例，它因为有同样的 class 属性而获得了相应的样式。但在这里，这个实例也被 ID 定义为 primary-product-list。每个页面通常只能有一个主要商品的列表，因此这个 ID 值还是比较恰当的。利用这个 ID，可以为这个模块实例添加额外的样式，可以增加一些 JavaScript 交互，还可以作为页内导航的目标。

实际开发中，一般不建议把 ID 属性作为 CSS 的“接入点”。利用类来添加样式往往能够让代码更简单也更容易维护。ID 可以用于在文档中标识元素，但通常不用于添加样式。第 12 章将详细介绍这方面内容。

1.2.2　结构化元素

HTML5 新增了一批结构化元素：

- section
- header
- footer
- nav
- article
- aside
- main

增加这些新元素是为了在 HTML 文档中创建逻辑性区块。它们可以用于包含独立内容（article）、导航组件（nav）、特定区块的头部（header），等等。其中，main 元素是最新增加的，用于高亮页面中包含主要内容的区域。关于如何正确使用这些新元素，建议看看这个网站：http://html5doctor.com。

除了 main 之外，所有其他新元素都可以在一个文档中多次出现，以便让机器和人更好地理解文档。在 HTML5 引入这些新元素以前，我们经常能够看到带有类似类名的 div 元素，比如下

面这篇博客文章的标记：

```
<div class="article">
  <div class="header">
    <h1>How I became a CSS Master</h1>
  </div>
  <p>Ten-thousand hours.</p>
</div>
```

其中的 div 元素对文档而言并没有语义价值，只是借助类名作为添加样式的“接入点”而已。这段代码中只有 h1 和 p 是有含义的。现在有了 HTML5 的新元素，这段标记可以改写成这样：

```
<article>
  <header>
    <h1>How I became a CSS Master</h1>
  </header>
  <p>Ten-thousand hours.</p>
</article>
```

经过修改，这段 HTML 的语义得到了增强，但同时也产生了意外的副作用。此时，我们只能通过 article 和 header 元素来添加样式了。添加样式的 CSS 选择符可能会是这样：

```
article {
  /* 样式 */
}
article header {
  /* 其他样式 */
}
```

但 article 和 header 都可能在同一个页面中多次出现，不一定是展示博文内容。如果**确实**存在这种重用的情况，而我们又通过元素选择符直接把样式绑定到了元素上，那么本来给博客文章应用的样式也会被应用到其他相同的元素上，不管合不合适。此时，更灵活也更有远见的做法是把这两个例子结合起来：

```
<article class="post">
  <header class="post-header">
    <h1>How I became a CSS Master</h1>
  </header>
  <p>Ten-thousand hours.</p>
</article>
```

相应的 CSS 规则就可以使用类名为这段标记应用样式了：

```
.post {
  /* 样式 */
}
.post-header {
  /* 其他样式 */
}
```

以上变化反映出了一个非常重要的概念，那就是，我们已经解耦了文档的语义与为文档添加样式的方式，从而让文档更便于移植、更具有目的性，因此也更容易维护。假如我们有一天觉得 article 并不是包含内容的最合适元素，或者由于 CMS 系统的固有限制，必须把它替换成 div

元素，那么只要改这一处就行了。因为样式是通过类名来应用的，所以无论出于什么原因修改了标签，都不会影响样式。

旧版本IE与新元素

在多数浏览器中使用新元素都没问题，只是IE8及更早的IE不会给自己不认识的元素应用样式。不过好在我们可以使用一个JavaScript“垫片”或“腻子”脚本来解决这个问题。

这里给大家推荐一个这样的脚本：https://github.com/aFarkas/html5shiv。

这个脚本其实也包含在前面推荐过的Modernizr库里面，后面几章也会用到这个库。

假如你有很多用户在使用旧版本的浏览器，那么在使用这些新元素的时候务必小心。这是因为，要确保一切正常，有可能需要引入上述JavaScript依赖才行。

1.2.3　div 和 span

既然有了新语义元素，那么为我们效力多年的 div 元素是否就多余了呢？不是的。在没有合适的语义元素的情况下，div 仍然是给内容分组的一个不错的选择。有时候，我们会纯粹出于添加样式的目的而在文档中添加一个元素。比如，为实现居中布局而在整个页面外部包装一个元素。

如果有更具语义的结构化元素，那么务必使用它们，需要添加样式时再给它们一个适当的类名。但是，如果你只需要一个无语义的元素作为额外的样式接入点，那就使用 div。以前有一个说法，叫作“div 麻疹”，意思是有些人写的 HTML 里几乎全部都是 div，也不管用得合不合适。因此，请确保只在额外提供样式接入的情况下才使用 div，但也不要因为用了几个 div 就感到难为情。在后面介绍的几个具体的例子中，我们会看到，额外添加的无语义 div 元素对保证代码的清晰和可维护性非常重要。

与 div 元素类似的还有 span。同样，在无须表示语义、仅需添加样式的情况中，可以使用 span。与 div 不同，span 是**文本级元素**，可以用于在文本流中建立结构。不过在使用无语义的 span 之前，也一样要确保真的不需要使用任何语义元素。比如，使用 time 标记时间和日期，使用 q 标记引用，使用 em 标记需要强调的内容，使用 strong 标记需要重点强调的内容：

```
<p>At <time datetime="20:07">7 minutes past eight</time> Harry shouted, <q>Can we just end
this, now!</q> He was <strong>very</strong> angry.</p>
```

1.2.4　重新定义的表现性文本元素

时至今日，<b>和<i>可以算是幸存的表现性标记了，它们以前分别用于将文本标记为粗体（bold）和斜体（italic）。你是不是以为新的 HTML5 规范会把它们剔除掉？没有，它们还在。这是因为，在旧有 Web 内容中，或者通过低水平 WYSIWYG 编辑器创建的内容中，这两个元素随处可见。HTML5 的编辑最终决定保留它们，但改变了它们的含义。

今天，<i>元素用于标识与周围内容不一样的内容，一般在排版上会显示为斜体。HTML5规范中给出的例子包括另一种语言中的习语，以及一艘船的名字。

<b>元素的含义和<i>几乎一样，只是针对习惯上标记为粗体的内容。相关的例子包括商品或品类名。

这样的定义确实不够明确，但关键是要知道，这两个元素与及的区别在于，它们没有任何强调自己所包含内容的意味。多数情况下，应该选择或，因为它们是用来强调及重点强调内容的语义正确的选择。

1.2.5 扩展 HTML 语义

长时间以来，Web 开发者一直在探索给 HTML 有限的词汇表添加新的语义和结构的方式。为内容添加更丰富的语义，对 Web 和基于其构建的工具而言意义重大。虽然建设语义 Web 王国任重而道远，但不管怎么说，还是有了实质性的进步。利用这些成果，HTML 编写者可以为自己的文档添加更细粒度、更具表达性的语义。

1. ARIA 的 role 属性

很多新的 HTML5 元素都考虑到了无障碍访问的场景。比如，如果屏幕阅读器能够理解页面中的 nav 元素，那么它就可以利用这个元素帮助用户定位到相应的内容，或者在必要时返回导航。

另一种实现这个目标的方式是利用无障碍富因特网应用（ARIA，accessible rich Internet application），它是对 HTML 规范的补充。ARIA 为我们提供了针对辅助访问设备添加更多语义的手段，方式就是为文档中的不同元素指定其包含什么内容，或者说它们提供什么功能。比如，role="navigation"这个“地标角色”属性用于声明一个元素具有导航的角色。其他角色有：

- banner
- form
- main
- search
- complementary
- contentinfo
- application

完整的 ARIA 角色及其定义，参见 ARIA 规范。

我们再推荐一个关于如何使用无障碍角色的简要分析，*Using WAI-ARIA Landmarks – 2013*，作者是来自 Paciello Group 的 Steve Faulkner。

ARIA 还支持让开发人员指定更复杂的内容片段和界面元素。例如，在使用 HTML 创建一个音量滑动条部件时，应该包含值为 slider 的 role 属性：

```
<div id="volume-label">Volume</div>
<div class="volume-rail">
  <a href="#" class="volume-handle" role="slider" aria-labelledby="volume-label"
  aria-valuemin="1" aria-valuemax="100" aria-valuenow="67" ></a>
</div>
```

属性 aria-labelledby、aria-valuemin、aria-valuemax 和 aria-valuenow 也分别提供了额外的信息，辅助阅读技术可以利用它们帮助残障用户使用这个滑动部件。

为 HTML 页面中的不同组件添加这些额外的语义属性，同样有助于为元素添加脚本和样式，是典型的多赢策略。

2. 微格式

目前最广泛采用的扩展 HTML 语义的方式是**微格式**。微格式是一组标准的命名约定和标记模式，可用于表示特定的数据类型。微格式的命名约定是基于 vCard 和 iCalendar 等已有的数据格式制定的。比如下面的联系人信息就是以 hCard 格式标记的：

```
<section class="h-card">
  <p><a class="u-url p-name" href="http://andybudd.com/">Andy Budd</a>
    <span class="p-org">Clearleft Ltd</span>
    <a class="u-email" href="mailto:info@andybudd.com">info@andybudd.com</a>
  </p>

  <p class="p-adr">
    <span class="p-locality">Brighton</span>,
    <span class="p-country-name">England</span>
  </p>
</section>
```

以微格式标记的联系人信息便于开发人员编写工具从中提取数据。比如可以编写一个浏览器插件，从你浏览的页面中发现微格式，然后让你把联系人信息下载到通讯录，或者把活动信息添加到你的日历应用。目前微格式支持的数据类型包括联系人、活动、菜谱、博文、简历，等等。微格式也可以用于表示关系，比如一段内容与链接到该内容的另一个 URL 之间的关系。

微格式得以流行的原因之一是容易实现，迄今已经被 Yahoo!和 Facebook 等内容平台采用，而且已经直接添加到了 WordPress 和 Drupal 等内容发布工具中。2012 年一项关于结构化数据实现的研究（http://microformats.org/2012/06/25/microformats-org-at-7）发现，微格式在 Web 上的应用最为广泛。不过最近微数据异军突起，也不容小觑。

3. 微数据

微数据是跟 HTML5 一起，作为给 HTML 添加结构化数据的另一种方式而推出的。它的目标与微格式非常相近，但在把微数据嵌入内容方面则有所不同。下面看一看用微数据标记同样的联系人信息会是什么样：

```
<section itemscope itemtype="http://schema.org/Person">
  <p><a itemprop="name" href="http://thatemil.com/">Emil Björklund</a></p>
    <span itemprop="affiliation" itemscope
```

```
      itemtype="http://schema.org/Organization">
        <span itemprop="name">inUse Experience AB</span>
      </span>
      <a itemprop="email" href="mailto:emil@thatemil.com">emil@thatemil.com</a>
    </p>
    <p itemprop="address" itemscope itemtype="http://schema.org/PostalAddress">
      <span class="addressLocality">Malm?</span>,
      <span class="addressCountry">Sweden</span>
    </p>
  </section>
```

通过这个例子可以看出，微数据的语法比微格式要烦琐一些，不过这是有原因的。由于微数据设计的时候考虑到了可扩展性，它可以用来表示任意类型的数据。微数据只定义一些语法来表示数据结构，但自身并未定义任何词汇表。相反，微格式则定义了具体的结构化数据，比如 hCard 和 hCalendar。

微数据把定义特定格式的事交给了使用者或第三方。上述例子中使用的格式就是由 Bing、Google 以及 Yahoo!等搜索引擎共同创建的 https://schema.org 中的一个词汇表。这几家搜索引擎使用它来辅助索引和排名页面，当然搜索爬虫也会使用这些词汇表，从你的内容中更高效地提取丰富的信息。

1.2.6 验证

即使经过深思熟虑，你的标记已经非常语义化了，其中也仍然存在输入或者格式错误的风险。这些隐患会带来无法预料的麻烦。这时候就要使用验证了。

现实中的多数 HTML 文档并不是真正有效的 HTML。用规范编写者的话说，就叫作“未遵行”（nonconformant）。这些文档中存在的问题有元素嵌套不对、包含未经编码的和号（&），以及缺少必要的属性等。浏览器对这类错误非常宽容，总会尝试猜测作者的意图。事实上，HTML 规范中也包含了如何处理无效 HTML 的规定，以确保浏览器厂商以一致的方式处理错误。

总体来说，浏览器如此大度地帮我们处理错误是个好事，但不代表我们可以因此而放弃自己的职守。我们都应该尽力写出有效的 HTML 文档，这样有利于更快地查找问题，避免错误泛滥。假如你碰到一个渲染或布局上的 bug，一时又找不出问题所在，最好先验证一下 HTML，以保证你的样式应用到了格式正确的文档上。

验证 HTML 的工具有很多。比如，可以使用 W3C 网站上的 HTML 验证器（http://validator.w3.org/），或者与之相关的插件。其中的 Web Developer 扩展，Firefox、Opera 和 Chrome 都支持。此外，如果你的项目有自动构建或测试环节，最好在其中加上 HTML 验证。

CSS 也是可以验证的。W3C 的 CSS 验证器地址是 http://jigsaw.w3.org/css-validator/。你可能认为验证 CSS 没有验证 HTML 那么重要，毕竟 CSS 中的错误一般不会导致 JavaScript 出错，或者导致屏幕阅读器无法打开页面。但是，我们还是建议你重视 CSS，保证其中没有什么低级错误，比如忘了写度量单位之类的。

根据你的 CSS 验证器设置而定，验证结果中可能包含很多关于厂商前缀的警告或错误。这些属性或值是浏览器厂商在实验性地支持某些 CSS 特性时使用的一种临时命名约定。比如，`-webkit-flex` 这个 `display` 属性的值，就是标准 `flex` 属性值在 WebKit 浏览器上的实验性版本。这些地方虽然会被验证器标记为警告甚至错误，但你的文件依然能正常使用。总之，只要明白验证器给出的这些标记的真正含义就行了。

验证并不是最终裁决，很多本身很好的页面也会验证失败，这是由于使用了来自第三方或低水准 CMS 的内容，或者使用了试验性 CSS 特性。此外，验证器本身也可能会跟不上标准更新和浏览器实现的步伐。因此不要过于激进，只要把验证当作事先帮我们发现一些低级错误的手段即可。

1.3　小结

本章介绍了一些必要的基础知识，包括 HTML 和 CSS 的一些重要概念。我们回顾了 HTML 和 CSS 的一些历史，知道了如何跟进它们的发展进度，以及如何做到既向后兼容又对未来友好。经过本章的学习，相信大家已经明白了编写可维护代码的重要性，也掌握了构建 HTML 结构以便一致地应用 CSS 的方法。

下一章，我们会重温关于 CSS 选择符的基础知识，然后再介绍 3 级和 4 级规范中的高级选择符。相关知识点还包括特殊性、继承和层叠，以及如何在样式表中有效使用这些特性。

微信连接

回复“CSS”查看相关书单

微博连接

关注@图灵教育 每日分享IT好书

QQ连接

图灵读者官方群I：218139230
图灵读者官方群II：164939616

图灵社区
iTuring.cn

在线出版，电子书，《码农》杂志，图灵访谈

“本书作者不但对CSS设计的底层技术和方法有着深刻的理解，而且更善于将这些知识娓娓道来。在跨浏览器支持问题上，无人可望其项背。”

——Molly E. Holzschlag，Web标准项目负责人，W3C HTML工作组专家

CSS作为Web标准的一部分，是现代网页设计中必不可少的关键元素。鉴于CSS标准化的快速发展，想要成为CSS高手，打造出令人惊艳、辨识度高的网站，就必须充分了解当前CSS规范各模块的新特性、新技术和新思想。

本书是经典CSS参考指南，自第1版出版至今一直畅销不衰。第3版针对当前浏览器支持度足够好的CSS特性全面改写，着眼于如何为更大、更复杂的网站编写优雅、巧妙、易维护的代码。两位作者均是技术与写作才能俱佳的开发高手，他们将自己对现代CSS实践的深刻理解融入书中，结合大量复杂实现情景，清晰展示了如何利用新技术写出高度灵活、易维护和可扩展的代码，并让这些代码在不同浏览器和不同平台上跑起来。

- 给网页添加样式的基础知识
- 基本的网页排版技术
- 通过背景、阴影、边框等属性美化元素盒子
- 常见内容布局技术及其应用场景
- 页面布局的系统方法及网格的使用
- 响应式Web设计细节剖析
- 表单与表格的样式化，让网页交互和复杂数据展示助力Web应用
- 在空间和时间维度上操作元素：变换、过渡与动画
- 混合模式、滤镜、蒙版等高级特效
- CSS开发实践中的质量控制与流程

Apress®
图灵社区：iTuring.cn
热线：(010)51095186转600
分类建议 计算机/Web开发/CSS
人民邮电出版社网址：www.ptpress.com.cn

ISBN 978-7-115-50690-0
ISBN 978-7-115-50690-0
定价：99.00元