

累积销量超过100万册

HTML and CSS Visual QuickStart Guide Eighth Edition

HTML5与CSS3基础教程

(第8版)

[美] Elizabeth Castro 著
Bruce Hyslop
望以文 译

- 全球最畅销Web入门书最新版
- 配套网站提供海量精彩示例
- 逐步指导快速创建响应式网站



人民邮电出版社
POSTS & TELECOM PRESS

数字版权声明

图灵社区的电子书没有采用专有客户端，您可以在任意设备上，用自己喜欢的浏览器和PDF阅读器进行阅读。

但您购买的电子书仅供您个人使用，未经授权，不得进行传播。

我们愿意相信读者具有这样的良知和觉悟，与我们共同保护知识产权。

如果购买者有侵权行为，我们可能对该用户实施包括但不限于关闭该帐号等维权措施，并可能追究法律责任。



Elizabeth Castro

享誉世界的计算机畅销书作家，电子出版先行者，2010年就出版了epub电子书制作教程*EPUB Straight to the Point*。Castro擅长使用详实的步骤和精美的实例教大家快速实现具体效果，她的HTML与CSS系列教程自出版以来广受读者欢迎，成为学习前端开发的首选参考图书。有关Castro的更多信息，可访问其个人主页ElizabethCastro.com。



Bruce Hyslop

自1997年开始从事网页开发，重点关注使用HTML、CSS和JavaScript进行网页开发和网站易用性的维护，并倡导最佳实践。Hyslop在加州大学洛杉矶分校进修部教授CSS课程，还著有*The HTML Pocket Guide*、*The Web Design Pocket Guide Boxed Set*。



望以文

毕业于中国人民大学，曾任百度前端工程师，现为网信金融产品经理。《HTML5与CSS3基础教程（第7版）》译者。热爱产品设计，维护微信公众号projoyo，不定期向读者推荐新鲜有趣的互联网产品。微博@weakow。



图灵程序设计丛书

HTML and CSS

Visual QuickStart Guide, Eighth Edition

HTML5与CSS3基础教程 (第8版)

[美] Elizabeth Castro Bruce Hyslop 著

望以文 译

人民邮电出版社

北 京

图书在版编目 (C I P) 数据

HTML5与CSS3基础教程：第8版 / (美) 卡斯特罗 (Castro, E.) , (美) 希斯洛普 (Hyslop, B.) 著 ; 望以文译. — 北京 : 人民邮电出版社, 2014. 5

(图灵程序设计丛书)

书名原文: HTML and CSS:visual quickstart guide
ISBN 978-7-115-35065-7

I. ①H… II. ①卡… ②希… ③望… III. ①超文本
标记语言—程序设计—教材②网页制作工具—教材 IV.
①TP312②TP393.092

中国版本图书馆CIP数据核字 (2014) 第058722号

内 容 提 要

本书自第1版至今,一直是讲解HTML和CSS入门知识的经典畅销书,全面系统地阐述HTML5和CSS3基础知识以及实际运用技术,通过大量实例深入浅出地分析了网页制作的方方面面。最新第8版不仅介绍了文本、图像、链接、列表、表格、表单等网页元素,还介绍了如何为网页设计布局、添加动态效果等,另外还涉及调试和发布。本书提供了一个强大的配套网站,上面列出了书中的完整代码示例以及更多优秀实例及进阶参考资料,以供读者参考学习。

通过学习本书,零起点读者即可创建网站,而中高级水平的开发人员也可以快速了解HTML5新元素、CSS3的奇幻效果、响应式Web设计以及各种最佳实践。

◆ 著 [美] Elizabeth Castro Bruce Hyslop
译 望以文
责任编辑 刘美英
责任印制 焦志伟

◆ 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: 27.25 彩插: 4
字数: 650千字 2014年5月第1版
印数: 1—5 000册 2014年5月北京第1次印刷

著作权合同登记号 图字: 01-2013-6737号

定价: 69.00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: 京崇工商广字第 0021 号

版 权 声 明

Authorized translation from the English language edition, entitled *HTML and CSS: Visual QuickStart Guide, Eighth Edition* by Elizabeth Castro, Bruce Hyslop, published by Pearson Education, Inc., publishing as Peachpit Press. Copyright © 2014 by Elizabeth Castro and Bruce Hyslop.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

Simplified Chinese-language edition copyright © 2014 by Posts & Telecom Press. All rights reserved.

本书中文简体字版由Pearson Education Inc.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。

译者序

HTML和CSS是万维网的根基。几年前，遵循Web标准之风盛行，网页结构与表现分离成为共识，网站开发人员普遍开始采用渐进增强的最佳实践，业内对HTML和CSS的认知度和重视度都得到了一次跨越式的提升。近年来，随着HTML5和CSS3技术的快速发展，HTML和CSS被赋予更强的功能性，它们的地位又到了前所未有的高度。

HTML和CSS的发展与变化，主要体现在两个方面。一是编写代码的方式有变化，例如，现在的智能手机普遍采用像素密度极高的Retina显示屏，如果还用过去的方式在网页中插入图像，这些图像在手机上就会显得不够锐利和清晰，因此我们有必要改变编写img标签的方式（参见本书5.8节）。二是新的技术取代了传统手段，例如，过去为了对标题等特殊文本使用不一样的字体，产生了图像替换文本的方法，现在，随着Web字体技术的成熟，人们可以使用更为简单、优雅的方式实现上述效果（参见本书第13章）。

过去的经验告诉我们，对于Web开发初学者来说，从一开始就学习最新技术和最佳实践，既可以节省时间，也有利于培养良好的编码习惯（因为避免了那些过时的编码方式和陈旧的技术手段的干扰）。

本书是HTML与CSS经典入门教程，目前已更新至第8版（第1版出版于18年前的万维网兴起之初）。跟过去的所有版本一样，本次新版也力图反映最新的编码实践与技术。相较于上一版，这一版几乎对全书的每个章节都有更新——删除了一些过时的做法，增加了不少新的提示与补充材料，大部分代码示例都进行了重写（相应的浏览器截图也都做了更新）。第8版不仅增加了一些新的知识板块（例如创建和使用为Retina显示屏准备的图像，通过使用可伸缩图像、弹性布局和媒体查询实现响应式网站等），还重新梳理了一些重点知识（例如对CSS层叠规则的讲解就重新整理了特殊性、顺序和重要性的关系）。

这套教程的经典之处是独特的分步讲解形式，贯穿全书的统一示例，以及通过提示和补充材料给出的丰富的扩展学习资源，因此对初学者而言，这套图书历来是学习效率的保证。新版不仅保留了这些经典的做法，还紧跟技术发展对内容做了全面更新，如果你是Web开发与设计的初学者，没有理由不将此书作为唯一的HTML与CSS入门书。

最后，借此机会，感谢图灵公司对我的信任与支持，感谢刘美英编辑，感谢她付出艰辛的努力，确保了本书按时出版。

前言

无论你是刚开始涉足网站制作，还是曾经做过网站、而今又想让自己的技能与时俱进，都会对这个领域振奋人心的进展感同身受。

在过去的几年里，为网页编写代码和添加样式的方式、浏览网页所用的浏览器以及使用浏览器的设备都发生了明显的变化。以前，我们只能通过台式机或笔记本浏览万维网，而如今我们可以通过很多设备访问万维网：手机、平板电脑，当然也包括笔记本、台式机，以及很多别的设备。

这是意料之中的事，因为万维网始终秉持消除边界的宗旨——无论是在城市还是在乡村，通过任何能访问万维网的设备，任何人都可以自由地分享和获取信息。总之，万维网的宗旨是普适性。万维网的覆盖范围不断扩大，过去，通信技术还未曾普及农村，而现在却已经实现了。

更伟大的是，万维网是属于每一个人的，任何人都可以自由地创建和发布网站。本书将指导读者创建和发布网站，是没有任何HTML和CSS知识的建站初学者的理想教程。书中内容结构清晰，浅显易懂，将一步一步地教会你如何创建网页。总之，本书是开发人员手头案边的必备指南。若想了解某个主题的更多信息，可以通过查找目录直接跳到相关页面。

HTML 和 CSS 简介

万维网成功的根基，是一种基于文本的标记语言，它简单易学，并且能被任何带有基本

Web浏览器的设备识读，它就是HTML。每个网页都或多或少会用一点儿HTML，否则也就不能称为网页了。

随着不断深入细致地学习，你会了解到，HTML用于定义内容的含义，而CSS用于定义内容和网页如何显示。HTML页面和CSS文件（**样式表**，stylesheet）都是文本文件，因此很容易编辑。后面的“如何学习本书”会向你展示一些HTML和CSS的代码片段。

从第1章起，我们开始学习基本的HTML页面；从第7章起学习用CSS定义页面样式。关于本书各章内容概述及主题汇总，请参见“本书涉及内容”。

HTML一词通常泛指这门语言本身，而HTML5则指代这一特定版本的HTML，比如讨论HTML早期版本所没有的HTML5新特性时。这样的规则也适用于CSS（泛指）和CSS3（特指CSS3）。

1. HTML与HTML5

了解一些有关HTML起源的基础知识对于理解HTML5很有帮助。

HTML诞生于20世纪90年代初，用于详细规定少量构建网页的元素。这些元素中的大多数都用于描述网页内容，如标题、段落、列表、指向其他网页的链接等。随着更多元素的引入以及对语法规则本身的调整，HTML这门语言的版本号也在更新。当前最新的版本便是HTML5。

HTML5是HTML早期版本的自然延续，它尽可能地满足了当前网站和未来网站的需求。它

从以前的版本中继承了大部分特性，这意味着，如果你在HTML5出现之前编写过HTML，那么你已经知道很多关于HTML5的知识了。这也意味着，HTML5的大部分内容都可以兼容新旧浏览器。向后兼容是HTML5的一项重要设计原则（参见www.w3.org/TR/html-design-principles/）。

HTML5还增加了大量新的功能。很多新功能都很简单，比如用于描述内容的辅助元素（如article、main、figure等）；还有一些用来协助创建强大Web应用程序的新功能则较为复杂。只有牢牢掌握了创建网页的技能，才能去学习HTML5更复杂的功能，这也是本书强调基础知识的原因。HTML5还引入了原生的音频和视频播放功能，书中也会讲到。

2. CSS与CSS3

CSS的第一个版本是在HTML诞生几年后才出现的，正式推出是1996年。同HTML5与其早期版本的关系一样，CSS3也是CSS早期版本的自然延续。

CSS3比CSS早期版本更为强大，它引入了大量的视觉效果，如阴影、圆角、渐变等。（关于本书涵盖这方面信息的详细情况，请参见“本书涉及内容”。）

Web 浏览器

无论是使用电脑（如图0.2.1所示）、手机，还是其他的设备，我们都是通过Web浏览器访问网站的。不过，你使用的浏览器可能与其他用户使用的浏览器不同。

Windows预装了微软公司的浏览器Internet Explorer，OS X则预装了苹果公司的浏览器Safari。不过，用户可以免费下载其他浏览器替换上面的两种浏览器，如Chrome（谷歌出品）、Firefox（Mozilla出品）、Opera（Opera出品）。以上讨论的都是用于桌面电脑的浏览器。

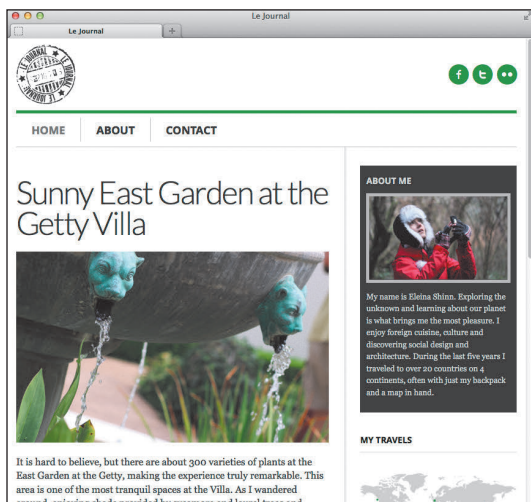


图0.2.1 Firefox的桌面版

至于移动设备，用户可以使用Safari的移动版（用于iPhone、iPad和iPod touch），各种Android默认浏览器——用于Android的Chrome、Firefox以及Opera Mini等。

本书会提到各种各样的浏览器。在大多数情况下，各个浏览器的最新版本对本书所讲的HTML和CSS特性的支持情况都是相似的。不过有时某个特性可能在一个或多个浏览器上不起作用（或表现不一致）。出现这些情况我会进行提示，并给出解决办法。这类情况主要发生在Internet Explorer 8上，这是目前开发人员还有必要关注的浏览器中最老的一个。（它的使用率仍在不断降低，因此2014年前后这种情况可能发生改变。）

20.2节会详细说明如何获取不同的浏览器，哪些浏览器对测试最重要，以及如何测试页面等问题。

浏览器的版本号

同HTML和CSS一样，浏览器也有版本号。版本号的数字越大，对应的浏览器就越新。

例如, Safari 7比Safari 6要新, 而Safari 6则比Safari 5要新。Internet Explorer 10比Internet Explorer 9新。不过Internet Explorer 10并不比Safari 7新。

出现这种情况是因为微软、苹果以及其他浏览器厂商并不会相互协调各自浏览器的版本号, 也不会协商新版浏览器的发布时间。Chrome和Firefox每隔六周就会更新一次, 而其他浏览器大约要间隔一年才更新一次, 因此Chrome和Firefox的版本号自然比其他浏览器的版本号要大得多。

很明显, 无论是哪家厂商发布浏览器, 新版本的浏览器对HTML和CSS特性的支持情况(当然也包括对其他特性的支持情况)都要好过先前的版本。

Web 标准与规范

你可能在想, 是谁第一个创造了HTML和CSS, 又是谁在持续地发展它们。由万维网和HTML的发明者Tim Berners-Lee主持的万维网联盟(W3C)便是负责带领Web标准发展的组织。

W3C发布的说明Web标准的文档称为规范(specification, 缩写为spec)。这些规范对HTML、CSS等语言的参数进行了定义。也就是说, 规范对语言规则进行了标准化。要了解W3C的活动, 请访问www.w3.org(如图0.3.1所示)。

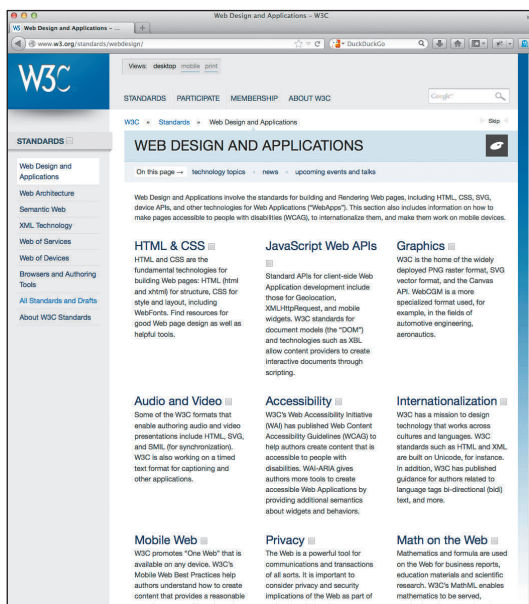


图0.3.1 W3C的网站是业内Web标准规范的主要信息来源

W3C与WHATWG

由于种种原因, 另一个组织, Web超文本应用技术工作组(WHATWG), 在负责开发HTML5规范的大量内容。W3C将WHATWG的工作纳入了其正在开发的规范的正式版本之中。可以通过www.whatwg.org访问WHATWG。

如果你想仔细查看这些规范(推荐你这样做), 可访问下面列出的这些规范最新版本的网址。

❑ HTML5 (W3C)

<http://www.w3.org/TR/html5/>

❑ HTML5.1 (W3C)

<http://www.w3.org/TR/html51/>

❑ 开发中的HTML标准 (WHATWG)

<http://www.whatwg.org/specs/web-apps/current-work/multipage/>

开发中的HTML标准包含了一些比HTML5.1规范更新的特性，这些特性尚处于开发状态（很多部分还在不断变动）。最终，这些内容会融入HTML5.1规范。

CSS的规范太多了，无法全部列在这里，不过可以通过这个网址找到CSS规范的列表：http://www.w3.org/standards/techs/css#w3c_all。

HTML4和HTML5的区别

如果你之前接触过HTML4，并想了解它与HTML5的区别，可以查看W3C为此创建的文档（<http://www.w3.org/TR/html5-diff>）。

本书会提到很多HTML4和HTML5的区别。这些内容对HTML的初学者来说并不重要，因为实际上现在大家都会使用HTML5。不过，如果你想进一步研究，这些内容也很有趣。

利用已经完成的标准，我们可以根据一套成形的规则创建网页，而浏览器也能根据这套规则显示这些页面。（整体上，浏览器对标准的支持是良好的。Internet Explorer的早期版本，尤其是Internet Explorer 8，则存在一些问题。）

经过几个阶段的发展，规范最终会被确定下来，成为**推荐标准**（Recommendation，www.w3.org/2005/10/Process-20051014/tr）。

尽管一部分HTML5和CSS3规范仍未最终敲定，但这并不意味着你不能使用这些规范。标准化的过程需要经历一些时间（准确地说需要数年）。浏览器会在规范成为推荐标准之前很久便实现其功能，这也是规范发展过程的反映。因此，浏览器已经包括了大量HTML5和CSS3的功能，尽管它们还没有成为推荐标准。

总之，本书涉及的功能都是规范中相对明确规定的，因此它们在推荐标准推出之前发生改变的可能性极小。很多HTML5和CSS3功能已经实际使用了一段时间了，开发者完全可以放心使用。

渐进增强：一种最佳实践

我在前言的开头处就讲到了万维网的普适性——万维网上的信息应该能被所有人访问。**渐进增强**（progressive enhancement）的理念能帮助你构建具有普适性的网站。这不是一门语

言，而是一种建站方法，它由Steve Champeon于2003年提出（http://en.wikipedia.org/wiki/Progressive_enhancement）。

这个想法很简单，但也很强大：开始用所有人都能访问的HTML内容和行为构建网站（如图0.4.1所示），再用CSS加入你的设计（如图0.4.2所示），最后用JavaScript（一种编程语言）添加额外的行为。这些组件都是分离的，但可以同时发挥作用。

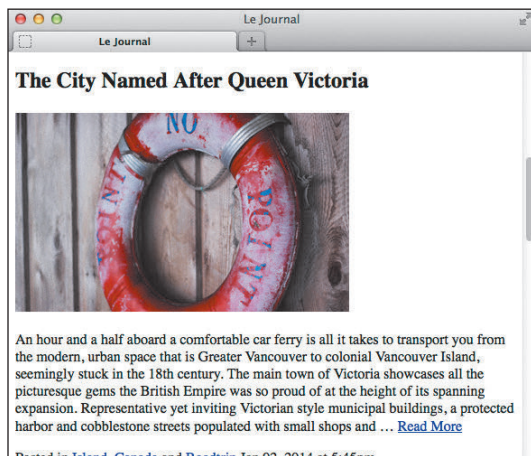


图0.4.1 一个基本的HTML页面，未对其应用任何自定义的CSS。通常，只有比较旧的浏览器才会以这种方式显示。这个页面可能并不是很好看，但信息都是可访问的，这一点非常重要



图0.4.2 同一个页面在支持CSS的浏览器中的显示效果。信息是相同的，只是呈现的方式不同（将页面向下滚动，右侧内容在图0.4.1中也是可见的）

更多例子

遵循渐进增强的原则，可以让网站的布局适应不同设备的屏幕尺寸和浏览器功能。这样，网页在移动设备、桌面电脑甚至更高级的设备上看起来都很不错。如果你对这个主题很感兴趣，可以先阅读第12章。

第14章介绍了如何让旧的浏览器显示简化的设计而现代浏览器显示包含CSS3效果的增强版本。

在本书的其他部分，你也会学到用于构建渐进增强网页的其他技术。

这样做的结果就是，那些只能访问基本页面的设备和浏览器得到的是简化的、默认的体验（如图0.4.1所示）。即便是20多年前万维网刚诞生时的浏览器也能显示这个页面，因此最早配备Web浏览器的手机也能显示它。此外，屏幕阅读器（可以为视障访问者读出网页内容的软件）也能轻松地处理这个页面。

同时，那些能够浏览更健壮的网站的可能设备和浏览器将看到增强的版本（如图0.4.2所示）。还有一些浏览器的处理能力介于二者之间，它们也能显示这个页面。不必要求网站对所有人来说体验都是一样的，关键是网站的内容是可访问的。

本质上，渐进增强背后的涵义是共赢。

目标读者

本书假定读者没有任何创建网站的知识。因此，从这一点上说，本书适合零起点初学者。你将从最基础的HTML和CSS知识学起。在这个过程中，你也会学到HTML5和CSS3的新特性，尤其是那些设计师和开发者在日常工作中经常用到的特性。

不过，即使你对HTML和CSS很熟悉，依然可以从本书获益，阅读本书可以快速了解HTML5的新元素、一些CSS3效果、响应式Web设计以及各种最佳实践。

1. 本书涉及内容

全书各章是按照如下方式进行组织的。

- ❑ 第1章～第6章及第15章～第18章讲解创建HTML页面的原则和我们需要使用的HTML元素，并清楚地说明什么时候该使用这些元素，以及如何使用。
- ❑ 第7章～第14章讲解CSS，从创建第一条样式规则开始，最终讲到运用CSS3增强视觉效果。
- ❑ 第19章演示如何向页面添加预先写好的JavaScript。
- ❑ 第20章讲解在将页面放到万维网上之前如何对其进行测试和调试。
- ❑ 第21章讲解如何保护自己的域名，以及如何把网站放到万维网上让其他所有人都能看到。

本书涵盖以下主题。

- ❑ 创建、保存、编辑HTML和CSS文件。
- ❑ 编写语义化HTML的含义及其重要性。
- ❑ 如何把页面的HTML内容、CSS表现和JavaScript行为分离（这是渐进增强的关键所在）。

- ❑ 通过一种有含义的方式，使用那些存在多年的HTML元素和新增的HTML5元素对内容进行结构化。
 - ❑ 从一个网页链接到另一个网页，或从页面的一部分链接到另一部分。
 - ❑ 在页面中添加图像，并针对万维网对其进行优化。这其中还包括了创建适应苹果的Retina显示屏及其他高分辨率显示屏的图像。
 - ❑ 使用ARIA（无障碍富因特网应用）地标角色和其他好的编码实践提升网站的可访问性。
 - ❑ 为文字添加样式（大小、颜色、粗体、斜体等），添加背景颜色和背景图像。
 - ❑ 实现多栏布局。
 - ❑ 构建响应式网页。响应式网页可以根据访问者的屏幕大小收缩或放大布局，从而按照开发者希望的方式适应不同的设备。这样做的结果是页面在手机、平板电脑、笔记本电脑、台式机以及其他可访问万维网的设备上看起来效果都很不错。
 - ❑ 使用@font-face为页面添加自定义Web字体，使用Font Squirrel和Google Fonts提供的字体服务。
 - ❑ 应用CSS3效果，如透明度、背景alpha透明、渐变、圆角、外阴影、内阴影、文字阴影以及多背景图像。
 - ❑ 使用CSS生成内容和sprite以减少页面所需图像文件的数量，让页面加载得更快。
 - ❑ 创建表单来获取访问者输入的内容，包括使用一些HTML5中新增的输入类型。
 - ❑ 使用HTML5的audio和video元素向页面插入媒体，对旧的浏览器使用备用的Flash播放器。
 - ❑ 还有很多主题。
- 这些主题都配有很多代码示例，说明如何基于

业内最佳实践来实现HTML5和CSS3的各个特性。

2. 本书未涉及内容

近年来，HTML和CSS领域的发展很迅猛，我们不得不舍弃一些主题。除了几处例外，我们坚持把那些极少用到、仍可能变化、缺乏广泛浏览器支持或者需要JavaScript知识的主题，以及一些高级主题排除在本书范围以外。

本书没有涉及下列主题。

- ❑ HTML5的details、summary、menu、command、output和keygen元素。W3C将其中的几个元素列入了2014年HTML5最终定稿时可能不包含的功能，应该尽量避免使用这里提到的元素。
- ❑ HTML5的canvas，这个元素允许我们使用JavaScript绘制图像（甚至创建游戏）。此外，还有可缩放矢量图形（SVG）。第17章对这两个主题进行了简要说明，还包含了一些涉及相关详细信息的网址。
- ❑ 需要JavaScript知识或者与HTML5新增的语义化元素没有直接关联的HTML5 API和其他高级特性。
- ❑ CSS3变换、动画和过渡效果。要了解更多信息，参见www.htmlcssvqs.com/resources/。
- ❑ CSS3中新的布局模型，如FlexBox、Grid等。这些技术致力于改变我们对页面进行布局的方法（前提是规范进一步发展，浏览器的支持程度更高一些）。参见Zoe Mickley Gillenwater的演示文档（www.slideshare.net/zomigi/css3-layout）和Peter Gasston的文章（www.netmagazine.com/features/pros-guide-css-layouts）。

如何使用本书

本书几乎所有章节都有实战代码示例（如图0.6.1和图0.6.2所示）。通常情况下，随后还会

展示这些代码对应的页面在浏览器中显示的屏幕截图（如图0.6.3所示）。

```
...
<body>
<header class="masthead" role="banner">
  ...
  <nav role="navigation">
    <ul class="nav-main">
      <li><a href="/" class="current-
        → page">Home</a></li>
      <li><a href="/about/">About</a>
        → </li>
      <li><a href="/contact/">Contact
        → </a></li>
    </ul>
  </nav>
  ...
</header>
...
</body>
</html>
```

图0.6.1 本书大部分章节都给出了一些HTML代码片段，且会突出显示跟相应章节内容对应的部分。省略号（...）表示出于简洁而省略的额外代码或内容。通常，可在其他代码示例中看到省略的部分

```
body {
  font-family: Georgia, "Times New Roman",
    → serif;
}

/* 站点导航 */
.nav-main {
  list-style: none;
  padding: .45em 0 .5em;
}

.nav-main li {
  border-left: 1px solid #c8c8c8;
}

.nav-main a {
  color: #292929;
  font-size: 1.125em;
  font-weight: bold;
}
```

图0.6.2 如果示例有CSS代码，会将它们单独显示出来，对应内容也会突出显示



图0.6.3 用于演示代码如何影响页面的一种或多种浏览器的屏幕截图

大多数屏幕截图用的都是当下最新版本的Firefox。不过，这并不是说与其他浏览器相比，我们更推荐Firefox。本书的代码示例在最新版本的Chrome、Internet Explorer、Opera和Safari中看起来都是相似的。

代码和屏幕截图下面都有对相应HTML元素和CSS属性的描述，以便于对示例背景进行说明，并加强你对它们的理解。

在很多情况下，你会发现只要理解说明和代码示例就可以使用HTML和CSS特性了。但如果你需要关于如何使用这些特性的明确指导，也可以阅读书中列出的详细步骤。

最后，大部分章节都有一些提示，为读者提供额外的用法信息、最佳实践、对本书相关部分的引用、对其他资源的链接等。

本书约定

本书遵循以下约定。

- ❑ 需要开发人员输入值的占位符或代码以斜体显示。大多数占位符出现在步骤讲解中。例如，输入padding: *x*，这里的*x*是要添加的内边距。
- ❑ 需要准确输入的代码（即HTML和CSS代码）使用等宽字体（如this font）。
- ❑ 代码图中的箭头（→）代表延续上一行的代码（这一行代码由于篇幅原因而进行了折行），如图0.6.2所示。箭头并非代码的一部分，不需要输入。实际使用中应该连续输入这一行代码，而无需像印刷页面中这样折行。

- ❑ 对于首次出现的词，在定义时使用黑体显示。
- ❑ IE是对Internet Explorer的简称。例如，IE9与Internet Explorer 9同义。
- ❑ 对HTML5和CSS3最新特性支持较好的浏览器统称为现代浏览器（modern browser）。通常，“Web浏览器”一节中提到的浏览器的最新版本都属于现代浏览器（IE8除外）。
- ❑ 浏览器的版本号后面出现加号（+）指的是该版本及其之后的所有版本。例如，IE8+指的是Internet Explorer 8及其后续版本。

本书配套网站

本书的配套网站上有本书的目录、本书涉及的（以及其他一些不方便放入书中的）完整代码

示例、引用资源的链接（以及一些额外资源的链接）、勘误表等。

下面是本书网站一些重要页面的URL。

- ❑ 主页：www.htmlcssvqs.com。
- ❑ 代码示例：www.htmlcssvqs.com/8ed/examples/。

读者可以直接在网页上查看这些代码，也可以把它们下载到自己的电脑里。读者可以获取书中涉及的所有HTML和CSS文件。

有时，我们会在代码里插入一些注释，解释对应代码的含义和用法。限于篇幅，我们对书中大量的代码示例进行了删减，但配套网站列出了完整的代码。

读者可随意使用这些代码，对它们进行修改使之满足自己项目的需要。

希望该网站对你有用。

致 谢

撰写本书过程中最有意思的就是与很多优秀的人士合作。所有的人都很用心、专业、和善且富有幽默感，让著书成为乐事。没有他们的努力，就不会有这本书。

将最诚挚的感谢献给以下各位。

Nancy Aldrich-Ruenzel和Nancy Davis，感谢他们对我的信任。

Cliff Colby，感谢他的支持，是他建立了我们的团队，让一切工作井井有条。

Robyn Thomas，感谢他让我们富有激情地工作，对内容提出建议，严格要求细节，并不断给予我们鼓励。

Scout Festa，感谢她简化语言的能力以及对本书的严厉监督，从而确保了全书的语言统一、文字优美流畅。

Aubrey Taylor，感谢他的众多建议和技术支持。这些内容对读者来说都非常有用。

David Van Ness，感谢他细致的页面排版工作，他的工作让全书看起来相当漂亮。

Valerie Haynes Perry，感谢他创建了一个高效的索引，可供读者反复查阅。

Peachpit公司的市场、销售等工作人员，感谢他们的幕后工作让本书得以面市。

Natalia Ammon，感谢她为第11章和第12章以及其他一些位置设计的漂亮页面。她的作品地址为www.nataliaammon.com。

Zach Szukala，感谢他将Natalia推荐给我。

Scott Boms、Ian Devlin、Seth Lemoine、Erik Vorhes和Brian Warren，感谢他们对本书上一版的贡献。

Victor Gavenda，感谢他帮助我们获取必要的软件。

Dan Cederholm、Ethan Dunham、Paul Irish、Mark James、Randy Jensen、Julie Lamba、Fontfabric、Google和Namecheap，感谢他们允许使用一些屏幕截图和设计资源（视具体情况而定）。

C.R. Freer，感谢她魔法般的摄影。

我的家人和朋友，感谢他们的鼓励，以及在写作间隙的陪伴。感谢他们在我几个月夜以继日的写作过程中不离不弃。

Robert Reinhardt，和前几版中的致谢一样，感谢他引领我接触写作，感谢他建议我蓄起美髯。

波士顿棕熊队^①，感谢他们在我写作间歇贡献了那么多激动人心的季后赛。

Web社区的人们，感谢他们分享技术知识，从而让他人获益（我在全书多处引用了他们的内容）。

读者朋友，感谢你们让我唤起自己当初学习HTML和CSS的记忆，让我得以按照最能让你们受益的方式讲解这些内容。感谢你们选择本书作为踏入万维网创作世界这一旅程的一部分。

^① 波士顿棕熊队是美国波士顿的一支冰球队。——译者注

最后，特别感谢Elizabeth Castro。她在20世纪90年代打造了这个品牌。多年来，本书一版再版，为无数读者提供了帮助。万维网赋予我太多

珍贵的东西，因此我尤其珍惜使用这一书名让读者学习HTML和CSS的机会。

Bruce

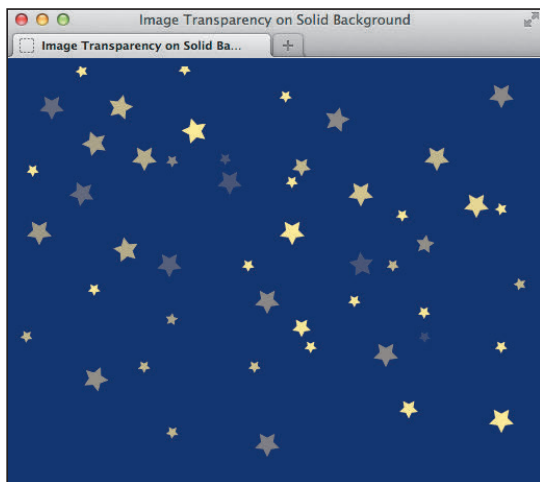


图 5.1.5 纯蓝色背景和从黑到红渐变色背景都不是星星 PNG 图的一部分，它们只是用 CSS 向页面的 body 添加的颜色。星星图具有 alpha 透明度，这样，无论背景是纯色、渐变色，还是另一张图像，就都能穿透，且看起来相当“干净”。JPEG 无法实现这一点，GIF 仅支持一种简单的透明度，效果并不太好（除非背景是纯色的）

图 5.1.2 标识和其他颜色较少的图像通常保存为 PNG-8 格式（也可以保存为 GIF 格式，不过相较之下 PNG 更好一些）



图 5.4.4 这里的修改反映的是左下方的区块。如果保存它，其大小就是 62.39K，尽管在图 5.4.2 中它只有 19.44K，但现在它的边缘更锐利了，这种质量上的提升是值得的。这里还修改了其他两个区块的设置。PNG-8（右上）的压缩让图像的像素化程度加深了，而且文件大小（70.56K）也比 JPEG 要大。PNG-24（右下）保持了很高的质量，但文件大小（224.3K）也大得多。显然，JPEG 是保存照片的最佳选择



图 5.4.5 这个图像（已被放大）有很多平铺的颜色及文字，它们需要保持锐利。可以看到，PNG-8 格式（左下）对图像的压缩是最好的，文件大小只有 5.5K。如果减少颜色的数量，其大小还将进一步减小。PNG-24 可以提供更多的颜色，使用这种格式的话，文件大小为 11.17K。使用最高质量的 JPEG 则为 19.76K。如果将 JPEG 的质量降为 50（这里没有显示），文件大小可比 PNG-8 更小一点，但其效果看上去却很可怕



图 7.1.4 标题从浏览器的默认的白色背景、黑色文字样式变为黄色背景、红色文字

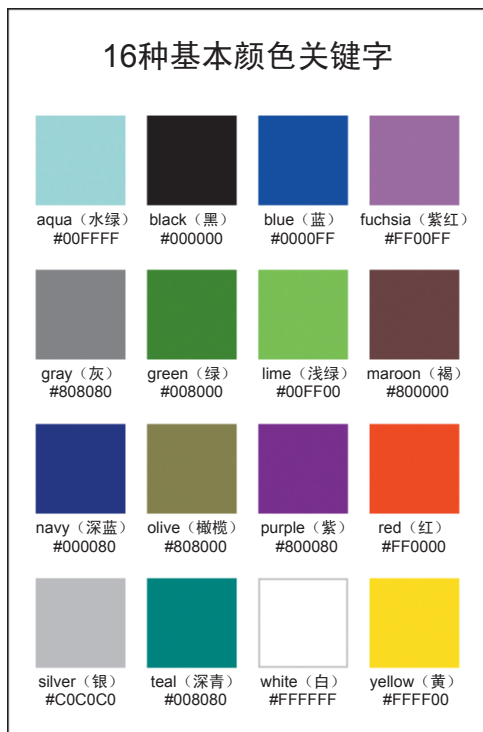


图 7.5.6 16 种基本颜色关键字，以及它们对应的十六进制数字值。CSS3 新增了 131 个关键字，不过通过十六进制、RGB 或 HSL 等格式可以定义多得多的颜色

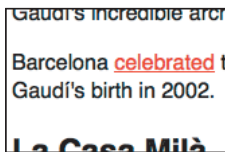


图 9.7.3 新的、未访问的链接显示为红色

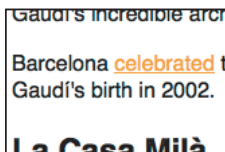


图 9.7.4 访问过的链接变为橙色

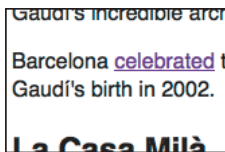


图 9.7.5 链接获得焦点（如通过 Tab 键）时显示为紫色

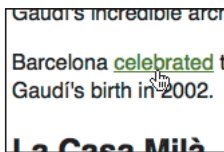


图 9.7.6 当访问者将鼠标指针停留在链接上时，它显示为绿色

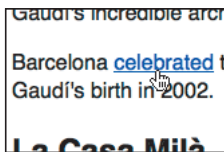


图 9.7.7 当访问者激活链接时，它变为蓝色

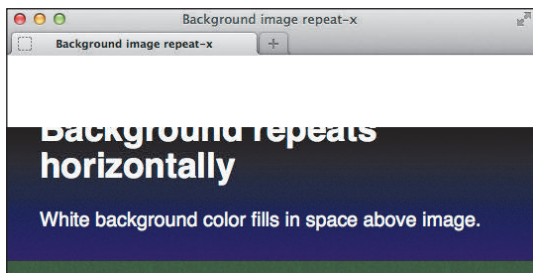


图 10.10.10 这张背景图像（上）由黑色向蓝色再向绿色渐变，很适合水平重复。不过它的高度不足以填充整个页面，在不设置背景色的情况下显得很难看。那一行文字其实并没有被切割，看不见它的上半部分是因为它的颜色与默认的背景色一样，都是白色

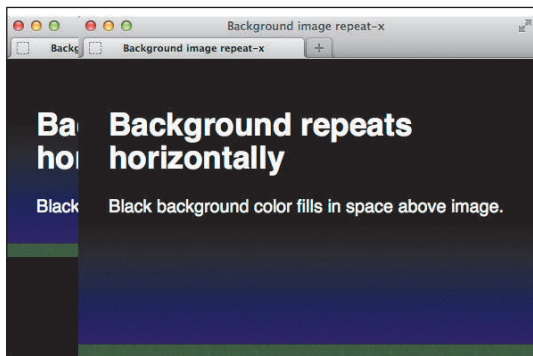


图 10.10.12 黑色（#000）背景色与图像顶端的黑色完美地融合在一起。现在，你可以看到那些文本了！这张图是由两个浏览器窗口截图合成的，这两个窗口的高度是一样的。如果没有为 html 元素添加图 10.10.11 中的规则，效果就如左图所示，背景图像的底部位于文字下面一点的位置（因为页面在那个位置结束了），因此黑色背景色会显示在它的下面

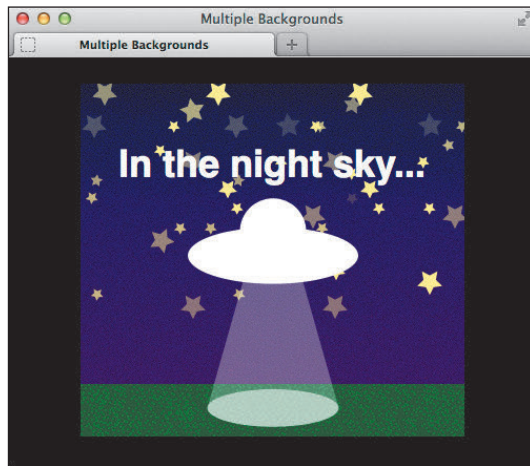


图 14.6.3 支持多重背景的浏览器显示示例的样子。其中，图像是分层次相互重叠在一起的，用逗号分隔的列表中的第一个图像位于顶部

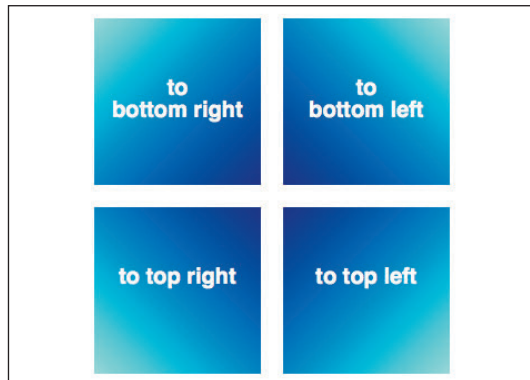


图 14.7.4 有两种指定渐变方向角度的方式。第一种方式就是这里所示的，使用关键字指定渐变方向需要旋转的角度。渐变会从跟指定角相对的角（对角线另一端的角）开始



图 14.7.5 使用关键字（参见图 14.7.4）只能创建沿对角线方向的渐变。第二种方式是指定渐变角度的度数，如 90deg（90 度）。数值代表的是圆周上的点的位置：0 代表最顶端的点，90 代表最左边的点，180 代表最底端的点，270 代表最右边的点。你列出的值决定的是渐变结束的点的位置。因此，0deg 等价于 to top，90deg 等价于 to right，以此类推

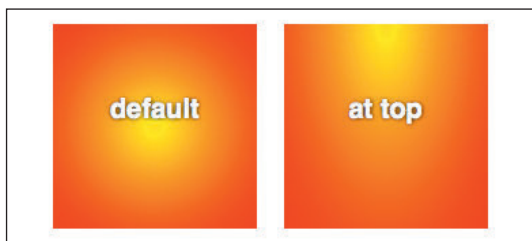


图 14.7.6 包含额外的可选参数的径向渐变。最简单的例子是默认的风格规则，它使用的参数与线性渐变是一样的。在这个例子中，渐变的原点是元素的中心。可以使用 `at top` 这样的关键字指定中心的位置。跟往常一样，可以在径向渐变声明之前为较旧的浏览器提供一个背景的定义

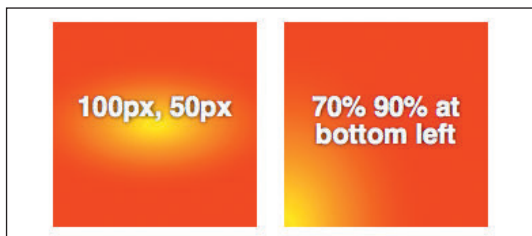


图 14.7.7 还可以控制渐变的尺寸。这里展示了几例子



图 14.7.8 这些例子结合了几个参数。它们都有三个颜色。位于 `at` 后面的值指定的是渐变的中心坐标。在第二个例子中，`30px 30px` 标记了渐变的尺寸。在第一个例子中，`closest-side` 决定了渐变的尺寸，它告诉浏览器从中心（指定为 `at 70px 60px`）向包含该渐变的区域的最近的一边伸展

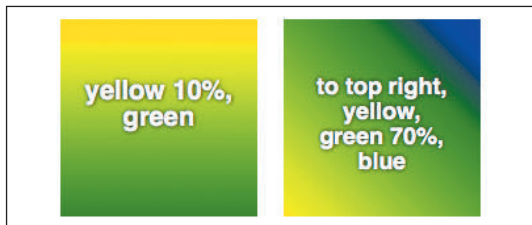


图 14.7.9 可以使用百分数指定一个以上的颜色停止位置

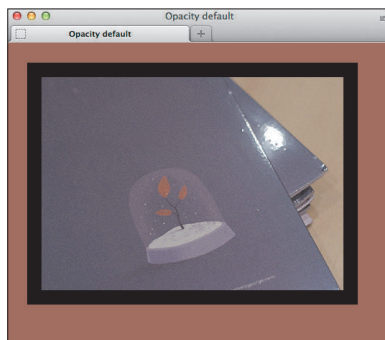


图 14.8.2 这是图 14.8.1 中 `div` 的默认样式。同其他元素一样，浏览器为其设置的默认样式为 `opacity: 1;`（不需要在样式表中指定）。该 `div` 有一个黑色的背景，由于该元素有内边距，因此背景会显示出来。`body` 的背景为褐色

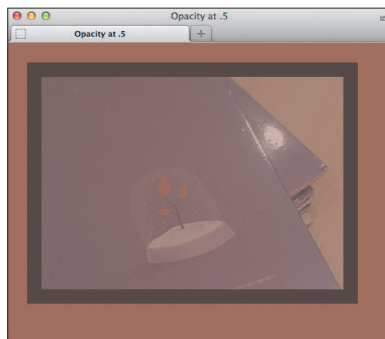


图 14.8.4 这是示例中 `div` 元素的 `opacity` 设为 0.5（即不透明度为 50%）时的样子。注意 `div` 元素的黑色背景现在呈现为深褐色，图像也变成半透明的，且带有一些褐色的色彩。如果将某元素的 `opacity` 设为小于 1 的值，包含它的周边元素就会透出来。在这个例子中，透出来的是 `body` 的褐色背景。如果将 `body` 的背景设为红色，效果会更明显（参见图 14.8.5）

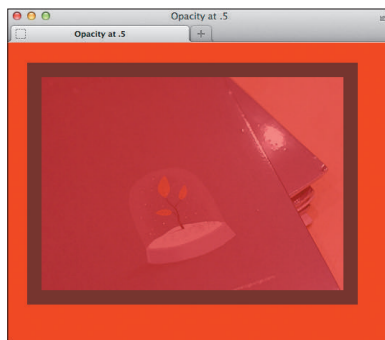


图 14.8.5 示例中 `div` 仍设为 `opacity: .5;`，但 `body` 改为 `background-color: red;`。通透的程度取决于不透明度。如果将不透明度设为 0.35，就会透出更多的红色；如果设为 0.8，则透出的红色就更少

目 录

第 1 章 网页的构造块	1	3.9 定义区块	45
1.1 HTML 思想	1	3.10 指定附注栏	46
1.2 基本的 HTML 页面	2	3.11 创建页脚	50
1.3 标签：元素、属性、值及其他	4	3.12 创建通用容器	53
1.4 网页的文本内容	7	3.13 使用 ARIA 改善可访问性	56
1.5 链接、图像和其他非文本内容	8	3.14 为元素指定类别或 ID 名称	59
1.6 文件名和文件夹名	9	3.15 为元素添加 title 属性	61
1.7 URL	10	3.16 添加注释	62
1.8 HTML：有含义的标记	13	第 4 章 文本	63
1.9 浏览器对网页的默认显示效果	16	4.1 添加段落	63
1.10 要点回顾	17	4.2 指定细则	64
第 2 章 处理网页文件	19	4.3 标记重要和强调的文本	65
2.1 规划网站	19	4.4 创建图	67
2.2 创建新的网页	20	4.5 指明引用或参考	69
2.3 保存网页	21	4.6 引述文本	70
2.4 指定默认页面或主页	24	4.7 指定时间	72
2.5 编辑网页	24	4.8 解释缩写词	75
2.6 组织文件	25	4.9 定义术语	76
2.7 在浏览器中查看网页	26	4.10 创建上标和下标	77
2.8 借鉴他人灵感	28	4.11 添加作者联系信息	79
第 3 章 基本 HTML 结构	30	4.12 标注编辑和不再准确的文本	80
3.1 开始编写网页	30	4.13 标记代码	83
3.2 创建页面标题	33	4.14 使用预格式化的文本	84
3.3 创建分级标题	34	4.15 突出显示文本	85
3.4 普通页面构成	36	4.16 创建换行	87
3.5 创建页眉	37	4.17 创建 span	88
3.6 标记导航	39	4.18 其他元素	89
3.7 标记页面的主要区域	41	第 5 章 图像	95
3.8 创建文章	42	5.1 关于 Web 图像	95
		5.2 获取图像	99

5.3 选择图像编辑器	100	9.7 按状态选择链接元素	163
5.4 保存图像	100	9.8 按属性选择元素	164
5.5 在页面中插入图像	103	9.9 指定元素组	168
5.6 提供替代文本	105	9.10 组合使用选择器	169
5.7 指定图像尺寸	106		
5.8 在浏览器中改变图像的尺寸	108	第 10 章 为文本添加样式	171
5.9 在图像编辑器中改变图像的尺寸	110	10.1 本章之前与本章之后	171
5.10 为网站添加图标	111	10.2 选择字体系列	173
第 6 章 链接	113	10.3 指定替代字体	174
6.1 创建指向另一个网页的链接	113	10.4 创建斜体	176
6.2 创建锚并链接到锚	118	10.5 应用粗体格式	177
6.3 创建其他类型的链接	120	10.6 设置字体大小	179
第 7 章 CSS 构造块	123	10.7 设置行高	184
7.1 构造样式规则	123	10.8 同时设置所有字体值	185
7.2 为样式规则添加注释	124	10.9 设置颜色	187
7.3 理解继承	126	10.10 设置背景	188
7.4 层叠: 当规则发生冲突时	129	10.11 控制间距	195
7.5 属性的值	132	10.12 添加缩进	195
第 8 章 操作样式表	139	10.13 对齐文本	196
8.1 创建外部样式表	139	10.14 修改文本的大小写	197
8.2 链接到外部样式表	140	10.15 使用小型大写字母	198
8.3 创建嵌入样式表	142	10.16 装饰文本	199
8.4 应用内联样式	143	10.17 设置空白属性	201
8.5 样式的层叠和顺序	145		
8.6 使用与媒体相关的样式表	147	第 11 章 用 CSS 进行布局	202
8.7 借鉴他人的灵感	148	11.1 开始布局的注意事项	203
第 9 章 定义选择器	150	11.2 构建页面	204
9.1 构造选择器	150	11.3 在旧版浏览器中为 HTML5 元素添加样式	206
9.2 按名称选择元素	152	11.4 对默认样式进行重置或标准 化	208
9.3 按类或 ID 选择元素	153	11.5 盒模型	209
9.4 按上下文选择元素	156	11.6 控制元素的显示类型和可见 性	211
9.5 选择第一个或最后一个子元 素	159	11.7 设置元素的高度和宽度	214
9.6 选择元素的第一个字母或者第 一行	161	11.8 在元素周围添加内边距	217
		11.9 设置边框	219
		11.10 设置元素周围的外边距	222
		11.11 使元素浮动	224
		11.12 控制元素浮动的位置	226

11.13 对元素进行相对定位.....	230	第 15 章 列表.....	301
11.14 对元素进行绝对定位.....	231	15.1 创建有序列表和无序列表	301
11.15 在栈中定位元素.....	232	15.2 选择标记	304
11.16 处理溢出.....	233	15.3 使用定制的标记	305
11.17 垂直对齐元素.....	234	15.4 选择列表的起始编号	308
11.18 修改鼠标指针.....	235	15.5 控制标记的位置	309
第 12 章 构建响应式网站.....	237	15.6 同时设置所有的列表样式属性	310
12.1 响应式 Web 设计：概述	237	15.7 设置嵌套列表的样式	311
12.2 创建可伸缩图像	239	15.8 创建描述列表	314
12.3 创建弹性布局网格	241	第 16 章 表单.....	318
12.4 理解和实现媒体查询	245	16.1 HTML5 对表单的改进.....	318
12.5 组合使用	251	16.2 创建表单	320
12.6 兼容旧版 IE.....	257	16.3 处理表单	323
第 13 章 使用 Web 字体.....	259	16.4 对表单元素进行组织	324
13.1 什么是 Web 字体	259	16.5 创建文本框	327
13.2 在哪里能找到 Web 字体	261	16.6 为表单组件添加说明标签	330
13.3 下载第一个 Web 字体	263	16.7 创建密码框	332
13.4 理解 @font-face 规则	265	16.8 创建电子邮件框、搜索框、 电话框和 URL 框	332
13.5 使用 Web 字体设置文本样 式	266	16.9 创建单选按钮	336
13.6 为 Web 字体应用斜体和粗 体	269	16.10 创建复选框	337
13.7 使用 Google Fonts 的 Web 字体	275	16.11 创建文本区域.....	338
第 14 章 使用 CSS3 进行增强.....	278	16.12 创建选择框	339
14.1 浏览器兼容性、渐进增强和 polyfill	278	16.13 让访问者上传文件	341
14.2 理解厂商前缀	280	16.14 创建隐藏字段	342
14.3 为元素创建圆角	281	16.15 创建提交按钮	343
14.4 为文本添加阴影	284	16.16 禁用表单元素	345
14.5 为其他元素添加阴影	285	16.17 根据状态为表单设置样式	346
14.6 应用多重背景	288	第 17 章 视频、音频和其他多媒体	349
14.7 使用渐变背景	290	17.1 第三方插件和步入原生	349
14.8 为元素设置不透明度	295	17.2 视频文件格式	350
14.9 生成内容的效果	297	17.3 在网页中添加单个视频	351
14.10 使用 sprite 拼合图像.....	299	17.4 为视频添加控件和自动播 放	352
		17.5 为视频指定循环播放和海报 图像	354

17.6 阻止视频预加载	355	第 19 章 添加 JavaScript	373
17.7 使用多种来源的视频和备用 文本	356	19.1 加载外部脚本	374
17.8 提供可访问性	358	19.2 添加嵌入脚本	378
17.9 音频文件格式	358	19.3 JavaScript 事件	378
17.10 在网页中添加带控件的单个 音频文件	359	第 20 章 测试和调试网页	380
17.11 自动播放、循环和预加载音 频	360	20.1 验证代码	380
17.12 提供带备用内容的多个视频 源	361	20.2 测试页面	382
17.13 添加具有备用 Flash 的视频 和音频	362	20.3 尝试一些调试技巧	385
17.14 高级多媒体	366	20.4 检查常见错误：一般问题	386
17.15 更多资源	366	20.5 检查常见错误：HTML	387
第 18 章 表格	368	20.6 检查常见错误：CSS	388
18.1 结构化表格	368	20.7 如果图像不显示	390
18.2 让单元格跨越多列或多行	371	第 21 章 发布网站	392
		21.1 获得域名	392
		21.2 为网站寻找主机	393
		21.3 将文件传送至服务器	394
		附录 A HTML 参考	398
		附录 B CSS 引用	408

第 1 章

网页的构造块



本章内容

- HTML 思想
- 基本的 HTML 页面
- 标记：元素、属性、值及其他
- 网页的文本内容
- 链接、图像和其他非文本内容
- 文件名和文件夹名
- URL
- HTML：有含义的标记
- 浏览器对网页的默认显示效果
- 要点回顾

尽管网页变得越来越复杂，但是其底层结构依然相当简单。正如前言中提到的，创建网页是离不开 HTML 的。你即将了解到，HTML 包含网页内容并说明这些内容的意义，Web 浏览器则会将 HTML 包着的内容呈现给用户。

一个网页主要包括以下三个部分。

- **文本内容（text content）**：在页面上让访问者了解页面内容的纯文字，比如关于业务、产品、家庭度假的内容，以及页面关注的其他任何内容。
- **对其他文件的引用（references to other files）**：我们使用这些引用来加载图像、音频、视频文件，以及样式表（控制页面的显示效果）和 JavaScript 文件（为页面添加行为）。这些引用还可以指

向其他的 HTML 页面和资源。

- **标记（markup）**：对文本内容进行描述并确保引用正确地工作。（HTML 一词中的字母 M 就代表标记。）

此外，在每个 HTML 页面的开头都有一些主要用于浏览器和搜索引擎（如 Bing、Duck Duck Go、Google、Yahoo 等）的信息。浏览器不会将这些信息呈现给访问者。

需要注意的是，网页的这些成分都仅由文本构成。这意味着网页可以保存为纯文本格式，可以在任何平台（无论是台式机、手机、平板电脑还是其他平台）上用任何浏览器查看。这个特性也确保了很容易创建 HTML 页面。

本章会带你创建一个基本的 HTML 页面，讲解 HTML 基础（包括上述三种主要成分），并讨论一些最佳实践。

注意 正如在前言中提到的，本书使用 HTML 泛指这门语言本身。如果需要突出 HTML 某一版本独有的特殊属性，则使用它们各自的名称。例如，“HTML5 引入了一些新的元素，并重新定义或删除了 HTML 4 和 XHTML 1.0 中的某些元素。”

1.1 HTML 思想

设想这样一种场景：你在厨房里，一只手上拿着便利贴，上面的每一页都写着一个

单词。有的贴纸上写的是“汤”，有的写着“燕麦片”、“盘子”、“酱汁”等。

你打开一个橱柜，为里面的每一样物品都贴上最能描述它的便利贴。一只黄色的装燕麦片的盒子贴上了写有“燕麦片”的贴纸。还有一个红色的盒子也是装燕麦片的，也为它贴上写有“燕麦片”的贴纸。其他的物品也采用类似的做法。

编写 HTML 与这个过程是很相似的。不同的是，编写 HTML 并非向食物和餐具上贴标签，而是为网页内容打上能够描述它们的标签。你无需自己创建标签名称，HTML 已经完成了这一工作，它有一套预先定义好的元素。`p` 元素用于段落，`abbr` 元素用于缩略词，`li` 元素用于列表项目。本书后面会进一步介绍这些元素，以及很多其他的元素。

注意，便利贴上的用词是“燕麦片”这样的，而不是“黄色的谷物盒子”、“红色的谷物盒子”。类似地，HTML 元素描述的是内容是什么，而非看起来是什么样。CSS（从第 7 章开始讲解）才控制内容的外观（如字体、颜色、阴影等）。因此，不管你最后让段落显示为绿色还是橙色，它们都是 `p` 元素，这才是 HTML 唯一关心的。

在学习本书和创建网页时，应该始终牢记这一思想。接下来即将讲到的基本网页就是这样做的。

1.2 基本的 HTML 页面

来看一个基本的 HTML 页面，大概了解一下本章及后面的内容。我们会对本节中的代码进行讲解。不过，即便你现在不能完全理解这些代码也不必担心，这只是让你初步了解 HTML，本书接下来的部分会对其进行详细说明。当然，随着你进一步阅读本书的后续内容，还会了解到更多的细节。此外，

每个人的学习方法都不同，或许有的读者认为先阅读下一节，再返回来读这一节会更有效。

每个网页都由图 1.2.1 中所示的结构开始构建。这个 HTML 相当于一张白纸（参见图 1.2.2），因为访问者看到的内容位于主体（`body`）部分（即 `<body>` 和 `</body>` 之间的部分），而这一部分现在是空着的。我们马上就会给它填上内容。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Your page title</title>
</head>
<body>

</body>
</html>
```

图 1.2.1 每个网页都包含 DOCTYPE、html、head 和 body 元素，它们是网页的基础。在这个页面中，可以定制的内容包括两项，一是赋予 `lang` 属性的语言代码，二是 `<title>` 和 `</title>` 之间的文字

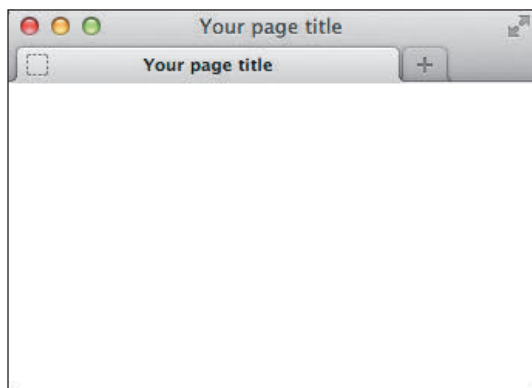


图 1.2.2 除非你是一个极简主义者，否则这不会是一个多么令人激动的页面

首先，简单说明一下（下一节会详细解释），HTML 使用 `<` 和 `>` 包围 HTML 标签。

开始标签（如 `<head>`）用于标记元素的开始，结束标签（如 `</head>`）用于标记元素的结束。有的元素没有结束标签，如 `meta`（参见图 1.2.1）。

1. 网页的顶部和头部

前面提到，页面内容位于主体部分，那么其他的代码有什么作用呢？实际上，`<body>` 开始标签以上的内容都是为浏览器和搜索引擎准备的。`<!DOCTYPE html>` 部分（称为 DOCTYPE）告诉浏览器这是一个 HTML5 页面。DOCTYPE 应该始终位于页面的第一行。

接下来是 `html` 元素，它包着页面的其余部分，即 `<html lang="en">` 和 `</html>` 结束标签（表示页面的结尾）之间的内容。在这里可以为网页指定其他语言，这一内容将在 3.1 节作详细说明。

再接下来是文档的头部，即 `<head>` 和

`</head>` 之间的区域。`<meta charset="utf-8" />` 会在 1.4 节进行讲解。主体前面的代码中，有一部分是用户可见的，即 `<title>` 和 `</title>` 之间的文本。这些文本会出现在浏览器标签页中，对于某些浏览器，这些文本还会出现在浏览器窗口的顶部，作为网页的标题（如图 1.2.2 所示）。此外，这些文本通常还是浏览器书签的默认名称，它们对搜索引擎来说也是非常重要的信息。

3.1 节和 3.2 节会对 DOCTYPE 和头部进行详细的讲解。

这就是网页的基础，并不太难，对吧？

2. 网页的主体：你的内容

接下来为页面添加一些内容（如图 1.2.3 所示）。图 1.2.4 显示了桌面浏览器呈现这段 HTML 的效果（如果添加 CSS，页面看起来会更漂亮）。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Blue Flax (Linum lewisii)</title>
</head>
<body>
<article>
  <h1>The Ephemeral Blue Flax</h1>

  <p>I am continually <em>amazed</em> at the beautiful, delicate <a href="http://en.wikipedia.
  → org/wiki/Linum_lewisii" rel="external" title="Learn more about Blue Flax">Blue Flax</a>
  → that somehow took hold in my garden. They are awash in color every morning, yet not a single
  → flower remains by the afternoon. They are the very definition of ephemeral.</p>
</article>
</body>
</html>
```

图 1.2.3 这个页面包含了本章开头提到的三种成分：文本内容、对其他文件的引用（图像的 `src` 值和链接的 `href` 值）及标记。我突出显示了主体部分的标记，读者可以很方便的区分标记和页面的文本内容。同时，请注意，行与行之间通过回车符分开了，这并不是必需的，不影响页面的呈现效果

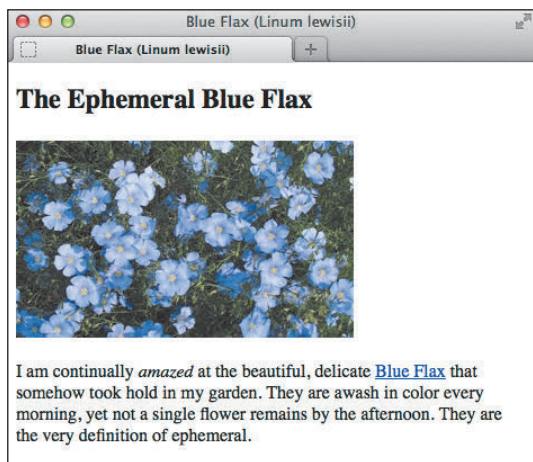


图 1.2.4 页面典型的默认呈现效果。这是页面在 Firefox 中显示的效果，在其他浏览器中的效果也是相似的。1.9 节解释了不同文本看起来不一样的原因

前面提到过（参见图 1.2.2），使用浏览器查看网页时，不会显示包围文本内容的标记。不过这些标记是非常有用的，我们就是使用它们来描述内容的。例如，`<p>` 标记用于表示段落的开始，具体的解释参见 1.8 节。

对 HTML 进行缩进

代码的缩进（参见图 1.2.3）与内容在浏览器中的显示效果（参见图 1.2.4）没有任何关系（第 4 章会讲到，`pre` 元素是一个例外）。不过，对嵌套在父元素中的代码进行缩进是一种惯例，这样我们在编写和阅读代码时就会更容易看出元素之间的层级关系。

HTML 提供了很多这样的元素。图 1.2.3 所示的例子演示了六种最为常见的元素：`a`、`article`、`em`、`h1`、`img` 和 `p`。每个元素都有各自的含义，例如，`h1` 是标题，`a` 是链接，`img` 是图像。

我们暂时不会对示例代码做过多的探讨。

在此之前，要进一步介绍 HTML 的基础知识，如通常意义上的元素、属性、文件名、URL 等。介绍完这些，再回过头来解释为什么要以这样的方式标记这些内容。我们还会介绍浏览器在默认情况下如何显示这些网页。

1.3 标签：元素、属性、值及其他

见识了一些 HTML 之后，我们来仔细看看标签的组成：元素（element）、属性（attribute）和值（value）。我们还将讨论父元素和子元素的概念。你在上文的基本 HTML 页面中已经见到了上面这些概念的例子，不过你可能并没有意识到。

1. 元素

我们在便利贴的例子中讲到，元素就像描述网页不同部分的小标签一样：这是一个标题，那是一个段落，而那一组链接是一个导航。我们在前面已经讨论过一些元素，有的元素有一个或多个属性，属性用来进一步描述元素。

大多数元素既包含文本，也包含其他元素（就像基本页面中的 `p` 元素）。如前面提到的，这些元素由开始标签、内容和结束标签组成。开始标签是放在一对尖括号中的元素的名称及可能包含的属性，结束标签是放在一对尖括号中的斜杠加元素的名称（参见图 1.3.1）。

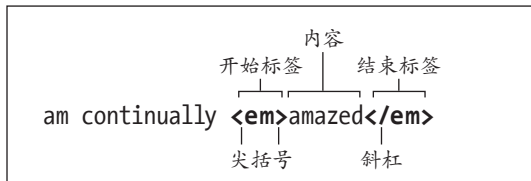


图 1.3.1 这是一个典型的 HTML 元素。开始标签和结束标签包着描述元素的文字。习惯上，标签采用小写字母

还有一些元素是空元素（empty element 或 void element），既不包含文本也不包含其他元素。它们看起来像是开始标签和结束标签的结合，由左尖括号开头，然后是元素的

名称和可能包含的属性，然后是一个可选的空格和一个可选的斜杠，最后是必需的右尖括号（参见图 1.3.2）。

```

```

可选的空格和斜杠

图 1.3.2 空元素（如这里显示的 `img` 元素）并不包含任何文本内容（`alt` 属性中的文字是元素的一部分，并非显示在网页中的内容）。空元素只有一个标签，同时作为元素的开始标签和结束标签使用。结尾处的空格和斜杠在 HTML5 中是可选的。不过，元素最后面的 `>` 是必需的

在 HTML5 中，空元素结尾处的空格和斜杠是可选的。XHTML 要求空元素结尾处必须有斜杠。以前用 XHTML 的人恐怕仍然倾向于在 HTML5 中继续使用斜杠，而其他人可能已经不用了。不管选择哪种方式，建议始终保持一致。

编写代码了，因此，除非无法抗拒，否则不推荐使用大写字母，这是一种过时的做法。

2. 属性和值

属性包含了元素的额外信息（参见图 1.3.3 和图 1.3.4）。在 HTML5 中，属性值两边的引号是可选的，但习惯上大家还是会写上，因此建议始终这样做。跟元素的名称一样，尽量使用小写字母编写属性的名称。

```
<label for="email">Email Address</label>
```

for 是 label 的一个属性
for 属性的值

图 1.3.3 这是一个 `label` 元素（关联文本标签与表单字段），它有一个简单的属性 - 值对。属性总是位于元素的开始标签内，属性的值通常放在一对括号中

```
<a href="http://en.wikipedia.org/wiki/Linum_lewisii" rel="external" title="Learn more about Blue Flax">Blue Flax</a>
```

href 是 a 的一个属性 href 的值 rel 也是 a 的一个属性 rel 的值
title 也是 a 的属性 title 的值

图 1.3.4 有的元素（比如图 1.3.2 中的 `a` 和 `img`）可以有多个属性，每个属性都有各自的值。属性的顺序并不重要。不同的属性 - 值对之间都用空格隔开

本书会对大多数属性可接受的值进行详细说明，不过不妨先看看都有哪些类型的值。

有的属性可以接受任何值，有的则有限制。最常见的还是那些仅接受预定义值（也

称为枚举值）的属性。也就是说，必须从一个标准列表中选一个值（参见图 1.3.5）。一定要用小写字母编写枚举值。

```
<link rel="stylesheet" media="screen" href="style.css" />
```

└─┬─┘
└─┬─┘
 预定义值 非预定义值

图 1.3.5 有的属性只接受特定的值。例如，只能将 link 元素里的 media 属性设为 all、screen、print 等值中的一个（参见第 8 章），不能像 href 属性和 title 属性那样可以输入任意值

有许多属性的值需要设置为数字，特别是那些描述大小和长度的属性。数字值无需包含单位，只需输入数字本身。图像（参见图 1.3.2）和视频的宽度和高度是有单位的，默认为像素。

有的属性（如 href 和 src）用于引用其他文件，它们只能包含 URL（统一资源定位符，是万维网上文件的唯一地址）形式的值。

本章后面会详细介绍 URL。

最后，还有一种特殊的属性称为布尔属性（Boolean attribute）。这种属性的值是可选的，因为只要这种属性出现就表示其值为真（图 1.3.6）。如果一定要包含一个值，就写上属性名本身（这样做的效果是一样的）。布尔属性也是预先定义好的，无法自创（你应该没那么叛逆吧）。

```
<input type="email" name="emailaddr" required />
```

└─┬─┘
 布尔属性

图 1.3.6 这段代码提供了一个让用户输入电子邮件地址的输入框（参见第 18 章）。布尔属性 required 代表用户必须填写该输入框。布尔属性不需要属性值，如果一定要加上属性值，则写作 required="required"

3. 父元素和子元素

如果一个元素包含另一个元素，它就被包含元素的父元素，被包含元素称为子元素。子元素中包含的任何元素都是外层的父元素的后代（参见图 1.3.7）。这种类似家谱的结构是 HTML 代码的关键特性，它有助于在元素上添加样式（从第 7 章开始讨论）和应用 JavaScript 行为（不在本书介绍范围）。

值得注意的是，当元素中包含其他元素时，每个元素都必须嵌套正确，也就是子元素必须完全地包含在父元素中。使用结束标签时，前面必须有跟它成对的开始标签。换句话说，先开始元素 1，再开始元素 2，就要先结束元素 2，再结束元素 1（参见图 1.3.8）。

```
<article>
  <h1>The Ephemeral Blue Flax</h1>
  
  <p>... continually <em>amazed</em> ... delicate <a ...>Blue Flax</a> ...</p>
</article>
```

图 1.3.7 在这段 HTML 代码中，article 元素是 h1、img 和 p 元素的父元素。反过来，h1、img 和 p 元素是 article 元素的子元素（也是后代）。p 元素是 em 和 a 元素的父元素。em 和 a 元素是 p 元素的子元素，也是 article 元素的后代（但不是子元素）。反过来，article 元素是它们的祖先

正确（代表标签对的线没有重叠）

```
<p>... continually <em>amazed</em> ...</p>
<p>... continually <em>amazed ...</p></em>
```

错误（代表标签对的线交叉了）

图 1.3.8 元素必须正确地嵌套。如果先开始 p，再开始 em，就必须先结束 em，再结束 p

1.4 网页的文本内容

元素中包含的文本可能是网页上最基本的成分。如果你用过文字处理软件，那么你一定输入过文本。不过，HTML 页面中的文本有一些重要的差异。

首先，浏览器呈现 HTML 时，会把文本中的多个空格或制表符压缩成单个空格，把回车符和换行符转换成单个空格，或者将它们一起忽略（参见图 1.4.1 和图 1.4.2）。

其次，HTML 过去只能使用 ASCII 字符。ASCII 只包括英语字母、数字和少数几个常用的符号。开发人员必须用特殊的字符引用来创建重音字符（在很多西欧语言中很常见）和许多日常符号，如 ´（表示 é）、©（表示 ©）等。完整列表见 www.elizabethcastro.com/html/extras/entities.html。

```
...
<body>
<p>I am continually <em>amazed</em> at
→ the beautiful, delicate Blue Flax
→ that somehow took hold in my garden.

They are awash in color every
→ morning, yet not a single flower
→ remains by the afternoon.

They are the very definition of
→ ephemeral.</p>
<p><small>&copy; Blue Flax Society.</small>
→ </p>
</body>
</html>
```

图 1.4.1 页面的文本内容（粗体部分）几乎就是标签以外的所有东西。在这个例子中，注意每个句子都至少包含一个回车符，有的词之间隔了好几个空格（在此只是为了说明 HTML 如何处理回车符和空格）。另外，第二段还包含一个表示版权符号的特殊字符引用（©）

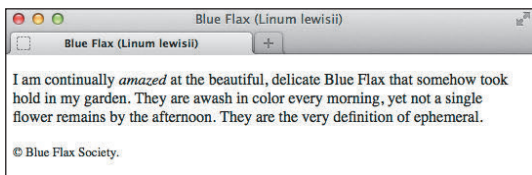


图 1.4.2 注意，使用浏览器查看这份文档时，多余的回车和空格都被忽略了，字符引用也替换成了对应的符号 (©)

Unicode 极大缓解了特殊字符问题。用 UTF-8 对页面进行编码（参见图 1.4.3），并用同样的编码保存 HTML 文件（参见 2.3 节）已成为一种标准做法。推荐你也这样做。在图 1.4.3 中将 charset 值指定为 UTF-8 和 utf-8 的结果是一样的。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title>Blue Flax (Linum lewisii)</title>
</head>
<body>
  ...
</body>
</html>
```

图 1.4.3 直接在 head 开始标签后面指明文档的字符编码。charset 属性用来设置编码类型（通常为 UTF-8）

有时还是会使用字符引用，如版权符号（因为 © 既好记又好输入），参见图 1.4.1。

1.5 链接、图像和其他非文本内容

显然，万维网充满生机的部分原因是页面之间的链接，以及图像、视频、音乐等。外部文件（如图像）实际上并没有放在

HTML 文件中，而是单独保存的，页面只是简单地引用了这些文件（参见图 1.5.1）。

```
...
<body>
<article>
  <h1>The Ephemeral Blue Flax</h1>

  <p>I am continually <em>amazed</em> at
  → the beautiful, delicate <a href=
  → "http://en.wikipedia.org/wiki/Linum_
  → lewisii" rel="external" title=
  → "Learn more about the Blue Flax">
  → Blue Flax</a> that somehow took
  → hold in my garden. They are awash
  → in color every morning, yet not a
  → single flower remains by the
  → afternoon. They are the very
  → definition of ephemeral.</p>
</article>
</body>
</html>
```

图 1.5.1 在我们的基本 HTML 文档中，有一个对图像文件 blueflax.jpg 的引用（位于 img 标签的 src 属性），浏览器在加载页面其他部分的同时，会请求、加载和显示这个图像。该页还包括一个指向关于 Blue Flax 的维基百科页面的链接（位于 a 标签的 href 属性）

浏览器可以轻而易举地处理链接和图像（参见图 1.5.2）。然而，它们无法处理其他任何文件类型。例如，对有的浏览器来说，要查看 PDF 就需要在系统中预先装好 Adobe Reader，要查看电子表格就需要预先装好 OpenOffice 这样的软件。

过去，HTML 没有内置的方法播放视频和音频文件。结果，各个厂商都开发出了相应的软件，用户可以下载并安装这些软件，从而弥补缺失的功能。这样的软件称为插件（plugin）。

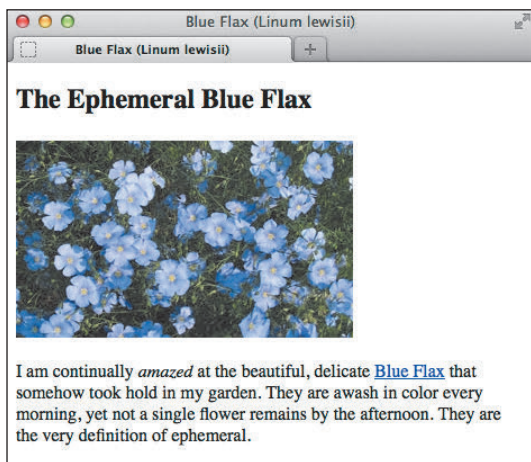


图 1.5.2 从网页引用图像和其他非文本内容，浏览器会将这些内容与文本一起显示。你在前面已经看到过了，默认情况下，链接文本的颜色与其他文本的颜色是不一样的，而且还带有下划线

在这些插件中，使用最为广泛的当数 Flash。多年以来，Flash 驱动了无数万维网上的视频。不过，这个插件也有一些问题，其中最为突出的就是它会耗费较多的计算资源。

幸好，HTML5 已经为缓解这一问题迈出了一大步。HTML5 提供了 audio 和 video 元素，这样，无需使用插件就可以播放音频和视频。不过，现代浏览器也提供了内置的媒体播放器，你仍然可以使用 Flash 播放器作为旧浏览器的备用工具。HTML5 的音频和视频还不完美，但至少开始向着无插件的目标前进了，而且还在继续前进。

本书将在第 5 章讨论图像，在第 6 章讨论链接，在第 17 章讨论 HTML5 的音频和视频。

1.6 文件名和文件夹名

同其他文本文件一样，网页也有文件名。对网页文件命名时要记住几点，这些要点有助于对文件进行组织，使访问者更容易找到并访问你的页面，确保他们的浏览器能正确地处理页面，以及增强搜索引擎优化（SEO），参见图 1.6.1 和图 1.6.2。（注意，本书会替换使用“文件夹”和“目录”，它们表示的含义相同。）

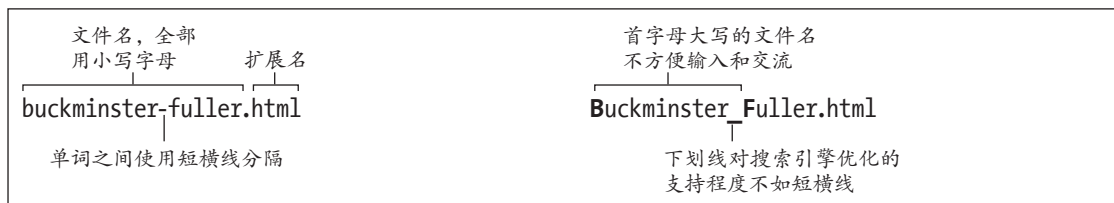


图 1.6.1 记住，文件名全部使用小写字母，用短横线分隔单词，用 .html 作为扩展名。混合使用大小写字母会增加访问者输入正确地址以及找到页面的难度

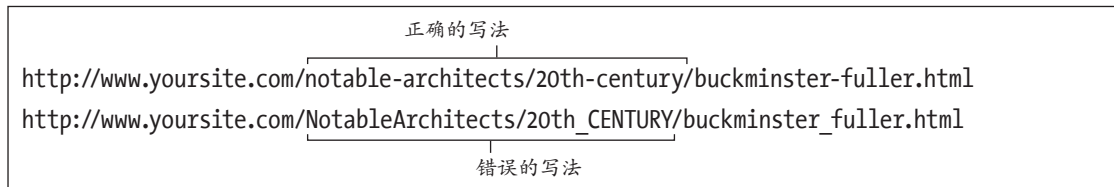


图 1.6.2 文件夹的名称也应全部用小写字母。关键是保持一致。如果使用小写字母，访问者和创建者就不必在大写字母和小写字母之间转换浪费时间了

1. 文件名采用小写字母

网页的文件名决定了访问者在访问你的页面时需要输入的文本，因此，文件名和文件夹名只用小写字母可以最大程度地避免访问者的输入错误。这样做对创建页面之间的链接也有很大的帮助。如果所有的文件名都用小写字母，你就少了一件需要操心的事情。

2. 使用正确的扩展名

浏览器主要通过查看文件的扩展名得知需要读取的文本文档是一个网页。尽管也可以使用 .htm 作为网页的扩展名，但通常使用的是 .html。如果页面使用其他的扩展名（如 .txt），浏览器会将其当做文本处理，相当于将你的代码直接呈现给访问者。

3. 用短横线分隔单词

不要在文件名和文件夹名中使用空格分隔单词。应该使用短横线，例如 company-history.html 和 my-favorite-movies.html。有的网站使用下划线（_），我们不推荐这种做法，因为短横线是搜索引擎更容易接受的方式。

提示 SEO 可以让网页在搜索引擎的搜索结果中排名靠前。

提示 Mac OS 和 Windows 有时不会显示文档的扩展名。如需查看扩展名，请更改文件夹选项。

1.7 URL

URL（Uniform Resource Locator，统一资源定位符）是地址的别名。它包含关于文件存储位置和浏览器应如何处理它的信息。互联网上的每个文件都有唯一的 URL。

URL 的第一个部分称为模式（scheme）。

模式告诉浏览器如何处理需要打开的文件。最常见的模式是 HTTP（Hypertext Transfer Protocol，超文本传输协议）。根据你上网的经验，你应该可以猜到，HTTP 是用于访问网页的（参见图 1.7.1）。HTTPS 是从 HTTP 衍生来的，用于电子商务网站等安全网页。它的格式同 HTTP 一样，只是用 https 替代了 http。

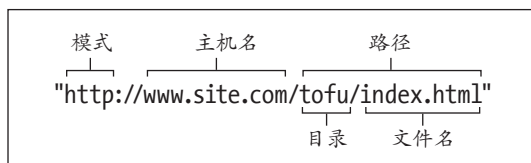


图 1.7.1 基本的 URL 包含模式、主机名称和路径。路径包含一个或多个目录（文件夹），最后是一个文件名

URL 的第二个部分是文件所在的主机的名称，紧接着是路径，路径包含到达这个文件的文件夹以及文件自身的名称（这些都是可选的）。如果 A 中的路径是 tofu/soft/index.html，就表示 index.html 位于 soft 目录，而 soft 目录则位于 tofu 目录，就像你在自己的电脑上组织文件和文件夹那样。

有时，URL 路径不以文件名结尾，而以一个目录结尾（可以包含一个结尾的斜杠，也可以不包含），如图 1.7.2 所示。在这种情况下，URL 指的是路径中最后一个目录中的默认文件，通常为 index.html。（通常情况下，所有 Web 服务器的配置都将 index.html 作为默认文件名，因此你不必修改任何服务器配置。）



图 1.7.2 以一个斜杠而非文件名结尾的 URL 指向最后一个目录（在这个例子中是 tofu 目录）中的默认文件。最常见的默认文件名是 index.html。因此，这个 URL 与上一个例子中的 URL 指向的是同一个页面

其他常见的模式有用于下载文件的 ftp（File Transfer Protocol，文件传输协议），如图 1.7.3 所示；用于发送电子邮件的 mailto，如图 1.7.4 所示（参见第 6 章）。



图 1.7.3 当用户点击这个 URL 时，浏览器会开始按 FTP 模式传输文件 proposal.pdf（有时需要先输入用户名和密码）



图 1.7.4 用于电子邮件地址的 URL 包括 mailto 模式，紧接着是一个冒号（没有斜杠），然后是电子邮箱地址本身

模式后面通常紧跟一个冒号和两个斜杠。mailto 是个例外，它后面只有一个冒号。

在上述模式中，最常用到的是 http（也包括 https），其次是 mailto，再次是 ftp。其他模式则只有在特殊情况下才会用到，你很少会碰到它们。

1. 绝对 URL

URL 可以是绝对的，也可以是相对的。绝对 URL（absolute URL）包含了指向目录或文件的完整信息，包括模式、主机名和路径（图 1.7.1 和图 1.7.2）。绝对 URL 就像是完整的通信地址，包括国家、州、城市、邮政编码、街道和姓名。无论邮件来自哪里，邮局都能找到收件人。就 URL 来说，这意味着绝对 URL 本身与被引用文件的实际位置无关，无论是在哪个主机上的网页中，某一文件的绝对 URL 都是完全一样的。

引用别人 Web 服务器上的文件时，应该总是使用绝对 URL。当你通过电子邮件跟你的朋友分享新闻或 YouTube 视频的 URL 时，如果只给他们 URL 的一部分，他们就无法看到相应的内容。这也就是为什么我们的基本页面（参见图 1.7.5）中指向维基百科页面的 href 值是完整的 URL 而不是简单的 Linum lewisii。

```
<p>I am continually <em>amazed</em> at
→ the beautiful, delicate <a href="http://
→ en.wikipedia.org/wiki/Linum_lewisii"
→ rel="external" title="Learn more about the
→ Blue Flax">Blue Flax</a> that somehow took
→ hold in my garden . . .</p>
```

图 1.7.5 由于我们的基本页面并不在 en.wikipedia.org，因此在链接到关于 Blue Flax（拉丁文中为 Linum lewisii）的页面时，需要使用绝对 URL。第 6 章对链接作了详尽的介绍，包括 rel 属性（如果链接是指向其他网站的，推荐使用这一属性）

对于 FTP 站点以及几乎所有不使用 HTTP 协议的 URL，都应该使用绝对 URL。

2. 相对 URL

当我告诉你我邻居家的位置时，我一般会说完整地址，而是说：“她家在右边第三个门。”这就是相对地址，它指出的位置是以信息提供者的位置为参照的。如果在别的城市按照同样的信息找我的邻居，你永远也找不到。（实际情况是，她出门很长一段时间了，因此恐怕你还是找不到她。）

同样，相对 URL 以包含 URL 本身的文件的位置为参照点，描述目标文件的位置。因此，相对 URL 可以表达像“指向本页面同一目录下的 xyz 页面”这样的意思。

◎ 引用同一目录下的文件

如果目标文件与当前页面（也就是包含 URL 的页面）在同一个目录中，那么这个文

件的相对 URL 就只有文件名和扩展名（参见图 1.7.6）。例如，链接的 HTML 可能是 `<a href="history.html">Take me to history.html!</a>`。

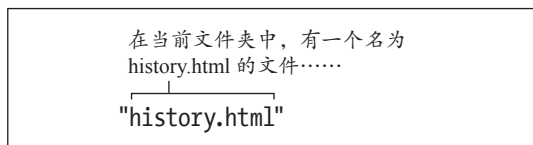


图 1.7.6 相对 URL 指向同一目录下的文件（参见图 1.7.10）。URL 中只需要文件名和扩展名，不必在前面加上 `http://www.site.com/about/`（这两个文件所在的主机名和目录）

◎ 引用子目录下的文件

如果目标文件在当前目录的子目录中，那么这个文件的相对 URL 就是子目录名，接着是一个斜杠，然后是文件名和扩展名（参见图 1.7.7）。例如，`<a href="info/data.html">Data supports my hypothesis</a>`。

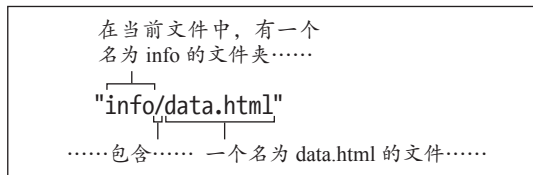


图 1.7.7 要引用当前文件夹的子文件夹中的文件，在这个例子中是 `data.html`（参见图 1.7.10），应在文件名之前加上子文件夹名称和一个斜杠

◎ 引用上层目录的文件

如果要引用文件层次结构中更上层目录中的文件，那么应该使用两个句点和一个斜杠（参见图 1.7.8），例如，`<a href="../products.html">our products</a>`。每个 `../` 都表示“到当前文件的上一层”，因此，`../..` 会向上走两级，`../../..` 会向上走三级。

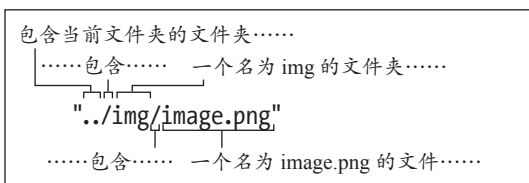


图 1.7.8 从图 1.7.10 中可以看到，这个文件位于与网站根目录下的当前文件夹（`about`）同属一层的文件夹（`img`）中。在这种情况下，使用两个句点和一个斜杠上升一级，然后指出子目录（`img`），再跟一个斜杠，最后是文件名（实践中，可以为 `image.png` 选择一个更具描述性的文件名，这里的名称仅作为示例）

◎ 根相对 URL

不过，如果文件位于 Web 服务器上，应该避免使用像 `../img/family/vacation.jpg` 这样显得较为笨拙的文件路径，不应先跳到网站的根目录再顺着向下找到目标文件（参见图 1.7.9）。可以在最开始使用一个斜杠，这样本例中的根相对 URL 就是 `/img/family/vacation.jpg`（假定 `img` 文件夹位于网站的根文件夹，这也是惯常的用法）。需要强调的是，这种做法只能用于 Web 服务器，例如在网站托管服务供应商的 Web 服务器上，或者在本地计算机运行的 Web 服务器上（参见提示）。

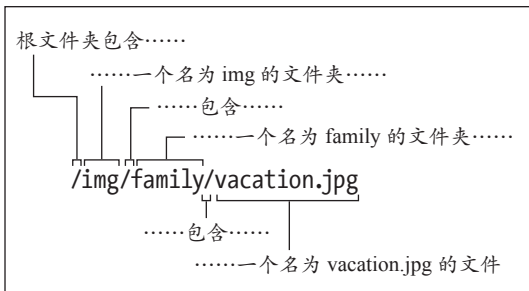


图 1.7.9 相同的根相对 URL 可用于所有的页面，不管页面在网站目录结构中的具体位置。例如，在这个例子中，主页和六层目录中某个页面到 `vacation.jpg` 的定位是同样简单的（这并不代表我建议使用这么深的层级结构）

如果不是在服务器本地开发网站，通常应使用相对 URL（当然，除非是指向其他服务器上的文件）。这样确保了将页面从本地系统传到服务器变得容易。只要每个文件的相对位置保持不变，就不必修改任何路径，链接依然有效。

提示 Apache 是在计算机上运行开发服务器最常见的选择。可以在网上搜索“set up local dev environment”（创建本地开发环境）。

提示 第 21 章会讨论如何找到一个 Web 主机。

3. 绝对 URL 和相对 URL 的比较

为了巩固上述知识，表 1.7.1 和图 1.7.10 共同演示了绝对 URL 和相对 URL 的区别。图 1.7.10 显示了两个网站的文件和目录结构。表 1.7.1 描述了从 `you-are-here.html` 访问各个不同文件的方式，这些文件有的位于同一个网站（`www.site.com`），有的位于另一个网站（`www.remote.com`）。尽管表格的前三行也可以使用绝对 URL，但访问同一网站的文件时最好使用相对 URL。

表 1.7.1 绝对 URL 与相对 URL

文 件 名	绝对 URL（可以在任何地方使用）	相对 URL（只能在 <code>you-are-here.html</code> 中使用）
<code>history.html</code>	<code>http://www.site.com/about/history.html</code>	<code>history.html</code>
<code>data.html</code>	<code>http://www.site.com/about/info/data.html</code>	<code>info/data.html</code>
<code>image.png</code>	<code>http://www.site.com/img/image.png</code>	<code>../img/image.png</code>
<code>news.html</code>	<code>http://www.remote.com/press/news.html</code>	（无，使用绝对 URL）
<code>join.html</code>	<code>http://www.remote.com/sign-up/join.html</code>	（无，使用绝对 URL）

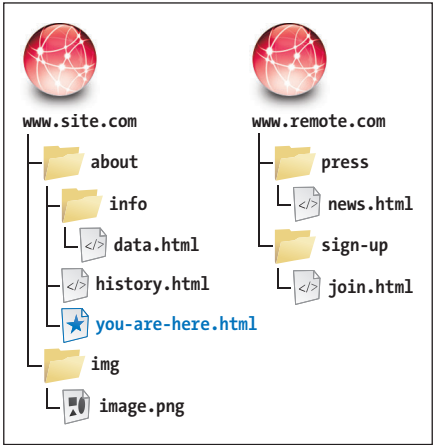


图 1.7.10 包含 URL 的文档（在这个例子中是 `you-are-here.html`）是相对 URL 的参照点。换句话说，相对 URL 是相对于这个文件在服务器上的位置。绝对 URL 与其所在的位置无关，因为它们总是包含资源的完整 URL

1.8 HTML：有含义的标记

本章一开始便将在网页中标记内容比作为橱柜物品打标签，因此你已经了解到了 HTML 的功能和相关的基础知识。

这里强调的是，HTML 描述的是网页内容的含义，即语义（*semantics*）。在 Web 社区中，语义化 HTML（*semantic HTML*）指的是那些使用最恰当的 HTML 元素进行标记的内容，在标记的过程中并不关心内容显示。我想你应该同意这种做法，这比胡乱选择元素要有意得多。

更好的是，这样做也非常容易，就像下面对我们的基本网页所做的解释那样。在此之后，“为什么语义很重要”进一步强调了编写语义化 HTML 是构建有效网站的基石。

1. 基本 HTML 页面的语义

本章前面提到，我们会回过头来讲解基本页面，接下来就来做这个工作。给你一分钟时间整理一下思路。

好了，接下来要深入地了解标记内容背后的思维过程。在此需要了解一些最为常用的 HTML 元素（参见图 1.8.1），所有这些元素都会在后续章节进行详细阐述。正如你看到的，创建拥有良好语义的 HTML 并不难。一旦熟悉这些元素，这一过程就会变得很简单了。

```
...
<body>
<article>
  <h1>The Ephemeral Blue Flax</h1>

  <p>I am continually <em>amazed</em> at
    → the beautiful, delicate <a href=
    → "http://en.wikipedia.org/wiki/Linum_
    → lewisii" rel="external" title="Learn
    → more about Blue Flax">Blue Flax</a>
    → that somehow took hold in my garden.
    → They are awash in color every morning,
    → yet not a single flower remains
    → by the afternoon. They are the very
    → definition of ephemeral.</p>
</article>
</body>
</html>
```

图 1.8.1 基本页面的 body 元素包含了 article、h1、img、p、em 和 a 元素。所有的内容都包含在 article 元素中

所有的内容都包含在 article 元素中，如图 1.8.1 所示。简言之，article 定义了一段独立的内容，其他地方可以重用这段内容。在我们的基本网页中使用 article 元素是个不错的选择，但并非每个网页都要这样编写。第 3 章会详细阐释 article 元素。

接下来是标题（参见图 1.8.2）。HTML

提供了六个标题级别，即 h1 ~ h6。其中，h1 是最重要的一级，h2 是 h1 的子标题，h3 是 h2 的子标题，以此类推，就像你使用文字处理软件编辑具有多级标题的文档一样。

```
<h1>The Ephemeral Blue Flax</h1>
```

图 1.8.2 标题是描述页面概貌的重要元素。标题确保了页面对屏幕阅读器用户来说更具可访问性，而搜索引擎用它们确定页面的重点

每个 HTML 都应该有一个 h1（或者多个 h1，这取决于页面的内容）。我们的页面只有一个标题，显然应该使用 h1。第 3 章会详细介绍 h1 ~ h6。

接下来，是一个图像（参见图 1.8.3）。img 元素是呈现图像的主要方式，对此，无需讨论用哪个元素合适。如果图像没有加载成功，或者页面是通过仅显示文本的浏览器查看的（这种情况已经很少见了），就会显示 alt 属性中的文字。屏幕阅读器可以对 alt 文本进行朗读（参见后面的“无障碍访问”）。第 5 章会详细介绍图像相关的内容。

```

```

图 1.8.3 用 img 在页面中插入图像很容易

段落使用 p 元素进行标记，如图 1.8.4 所示。正如印刷材料中的段落所显示的，一个段落可以包含多个句子。如果页面需要再加一个段落，只需要在第一个 p 元素之后再加一个 p 元素就可以了。

段落中嵌套了两个元素（em 和 a），分别定义其短语内容的含义（图 1.8.4）。这是 HTML5 提供的大量用于提升段落文本语义的短语内容（phrasing content）元素的两个例子。我们会在第 4 章中讨论短语内容元素和 p 元素。

```
<p>I am continually <em>amazed</em> at
→ the beautiful, delicate <a href="http://
→ en.wikipedia.org/wiki/Linum_lewisii"
→ rel="external" title="Learn more about
→ Blue Flax">Blue Flax</a> that somehow
→ took hold in my garden. They are awash in
→ color every morning, yet not a single
→ flower remains by the afternoon. They are
→ the very definition of ephemeral.</p>
```

图 1.8.4 p 元素可以包含定义段落内短语语义的其他元素，em 和 a 是两个例子

em 元素表示“强调”，就像你在说话时会重读某些词语。在我们的示例页面中，它强调了对花的惊叹之情（图 1.8.4）。记住，

```
<a href="http://en.wikipedia.org/wiki/Linum_lewisii" rel="external" title="Learn more about Blue Flax">
→ Blue Flax</a>
```

图 1.8.5 a 元素定义了一个指向维基百科中关于 Blue Flax 的链接。可选的 rel 属性指出链接指向的是另一个网站，这也增加了语义（没有这个属性链接也会正常工作）。可选的 title 属性提供了有关所指页面的信息，它也增强了 a 元素的语义。当用户用鼠标滑过该链接，就会显示 title 属性的内容

相当简单，对吧？当你对 HTML 元素了解更多之后，为内容选择正确的元素通常是相当简单的工作。偶尔，你会遇到一些内容，有一种以上的合理标记方式，这也没问题。有时候不能直接辨别对与错，当然大多数时候可以。

无论如何，浏览器都会将内容显示出来。浏览器还没有聪明到能够判断一段内容更适合使用 p 元素还是其他元素。

最后需要指出，HTML5 并没有试图为每一种能想到的内容类型提供对应的元素，因为这样会使这门语言变得笨重。相反，HTML5 采取了一种务实的态度，其所定义的元素覆盖了绝大多数情况。

HTML 之美，一部分在于人们能很容易掌握它的基础，构建一些页面，并在此基础上不断取得进步。因此，尽管 HTML 元素大约有 100 种，但不要被这个数字吓到，你经

HTML 描述的是内容的含义，因此 em 代表的是语义上的强调，而非视觉上的，不过通常会用斜体表示 em 文本（可以使用 CSS 改变这一样式）。

最后，基本页面通过 a 元素定义了一个链接。a 表示“锚”（anchor）。链接可谓是所有 HTML 元素中最强大的元素，因为它才得以形成万维网。万维网的定义就是，将一个页面与另一个页面或资源连接起来，或者将页面的一部分与另一部分连接起来。在这个例子中，文字“Blue Flax”是一个指向维基百科某页面的链接（参见图 1.8.5）。

常用到的只有少量核心元素，而其余的则较少使用。

你已经了解了一些常见元素，因此你已经有了一个良好的开始。

2. 为什么语义很重要

这里列出了使用语义化 HTML 最重要的几个原因。我们没有给出全部原因，我们已经在前面提到一些内容了。

- ❑ 提升可访问性和互操作性（内容对于借助辅助技术的残障访问者是可访问的，同时对于台式机、手机、平板电脑及其他设备上的浏览器都是可访问的）。
- ❑ 提升搜索引擎优化（SEO）的效果。
- ❑ 使维护代码和添加样式变得容易。
- ❑ （通常）使代码更少，页面加载更快。

● 无障碍访问

你可能对无障碍访问并不熟悉。它指

的是让内容对所有用户可用,不论其能力如何(参见 www.w3.org/standards/webdesign/accessibility)。万维网的发明者 Tim Berners-Lee 曾说过一句著名的话:“万维网的力量在于其普适性。让包括残障人士在内的每个人都能访问万维网,是极为重要的一点。”

任何带有浏览器的设备都可以显示 HTML,因为它只是文本。然而,用户获取内容的方式可能不同。例如,视力正常的人可以直接查看内容,而视力受损的用户则需要放大页面,调大字号,或者使用屏幕阅读器(可以将内容朗读出来的软件,是辅助技术的一个例子)。

有时,屏幕阅读器会将内容周围的 HTML 元素的类型读出来,让用户了解上下文。例如,对于列表,在读出列表各个条目之前,屏幕阅读器会首先告诉用户这里有一个列表。类似地,对于链接,屏幕阅读器会告诉用户这里有一个链接,方便其决定是否点击这个链接。

屏幕阅读器用户能够以多种方式浏览网页,例如通过键盘按键从一个标题跳到一个标题。这样,他们可以先了解一个页面的关键主题有哪些,再去听他们感兴趣的内容,而不必把整个页面从头到尾听下来。

由此可见,对残障人士来说,好的语义产生了多么大的差别。

● 搜索引擎优化 (SEO)

SEO 的效果也会得到改善,也就是说网页在搜索引擎中的排名会靠前,因为搜索引擎对用特殊方式标记的内容会赋予更高的权重。例如,标题告诉搜索引擎爬虫页面的主要主题,帮助浏览器对页面目录进行索引(index)。

● 更容易维护代码和添加样式

随着不断深入阅读本书,你会了解为什

么好的语义能使代码更有效、更易于维护和添加样式。你将了解到,借助 CSS,我们可以轻松统一特定元素的样式,例如,让所有的段落显示为深灰色,并使用 Georgia 字体。如果你将某些段落标记为 p 元素,某些标记为其他元素,那么就要在 CSS 中同时为这两种元素设定样式,这就添加了不必要的复杂性。

这样做也会让维护 HTML 变得困难。好的语义可以让网页变得统一和“干净”。这样做还会使文件尺寸变小,从而浏览器加载网页的速度变得更快。

1.9 浏览器对网页的默认显示效果

既然控制页面样式的是 CSS 而非 HTML (参见图 1.9.1),为什么在浏览器中查看我们的基本页面时,某些文字的字号会比其他文字的字号大呢,为什么有的文字是粗体或斜体呢(参见图 1.9.2)?

原因就是每个 Web 浏览器都有一个内置 CSS 文件(一张样式表),它决定了每个 HTML 元素的默认样式。你自己创建的 CSS 可以覆盖这些样式。对于不同的浏览器,默认样式会稍有差异,但总体上相当一致。重要的是,HTML 所定义的内容的底层结构和含义是一致的。

块级元素、行内元素以及 HTML5

容易看出,有些 HTML 元素(如 article、h1 和 p)从新的一行开始显示,就像纸质书中的各个段落一样,而另外一些元素(如 a 和 em)则与其他内容显示在同一行,如图 1.9.2 所示。需要强调的是,这是浏览器的默认样式,而不是 HTML 元素自身的样式,也不是由代码中两个元素之间的空行引起的(参见图 1.9.1)。

```

...
<body>
<article>
  <h1>The Ephemeral Blue Flax</h1>

  <p>I am continually <em>amazed</em> at
  → the beautiful, delicate <a href=
  → "http://en.wikipedia.org/wiki/Linum_
  → lewisii" rel="external" title="Learn
  → more about Blue Flax">Blue Flax</a>
  → that somehow took hold in my garden.
  → They are awash in color every morning,
  → yet not a single flower remains
  → by the afternoon. They are the very
  → definition of ephemeral.</p>

  <p><small>&copy; Blue Flax Society.
  → </small></p>
</article>
</body>
</html>

```

图 1.9.1 在基本页面的结尾又加了一段，这样在浏览器中查看网页时就能看到每个段落都占据单独的一行（图 1.9.2）。顺便说一下，`small` 元素表示的含义是法律声明等条文细则。默认情况下，它比其他的文字显示得小一些，但是显示小字号并不是使用这个元素的理由

这里需要多解释一下。在 HTML5 之前，大多数元素都可以划入块级（block-level，从新行开始显示）或行内（inline，与其他内容在同一行显示）两种类别。HTML5 废弃了这些术语，因为这些术语把元素与表现关联起来，而 HTML 并不负责表现。（通常，旧的行内元素在 HTML5 中都被归类为短语内容。）

尽管如此，浏览器不需要也不应该为这些元素改变默认显示规则。毕竟，你不想看到两个段落（`p` 元素）连到一起，或者强调的字句（`em` 元素包含的 `amazed`）将句子打断，单独成行。

因此，通常标题、段落和 `article` 这样的元素从新行开始显示，而短语内容（如 `em`、`a`

和 `small`）则与外围内容在同一行显示。

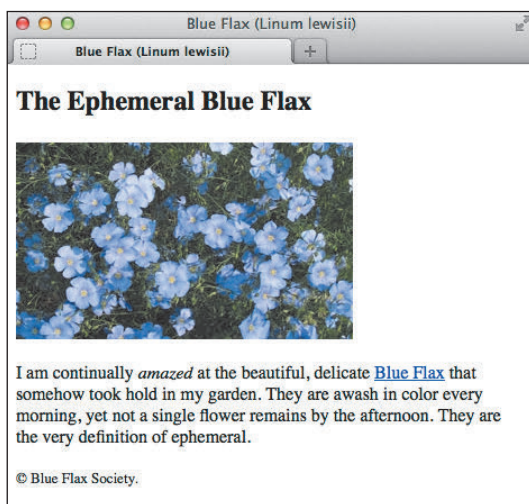


图 1.9.2 浏览器的默认样式表将标题（`h1 ~ h6`）与普通文本区别开来，对 `em` 文本加上斜体样式，对链接加上颜色和下划线。此外，有的元素从单独的一行开始（如 `h1` 和 `p`），而其他一些元素显示在外围内容的里面（如 `a`、`em` 和 `small`）。用你自己的样式表覆盖这些显示规则很简单

尽管 HTML5 不再使用块级、行内这些术语，但这样划分有助于理解它们的含义。由于在 HTML5 之前就成为 HTML 的常用词，因而这些术语在教程中很常见。本书也会偶尔使用这些术语，以说明元素在默认情况下是另起一行还是与其他内容共处一行。

从第 7 章开始将详细讲解 CSS，不过在只需要知道：样式表与 HTML 一样是纯文本，因此可以用编辑 HTML 的文本编辑器创建样式表。下一章就会讲解如何使用编辑器操作 HTML 文件。

1.10 要点回顾

HTML 基础以及某些关键最佳实践为构建有效网站打下了基础。下面我们来回顾一下本章讲述的要点。

- ❑ 一个网页主要由三种成分构成：文本内容、对其他文件的引用和标记。
- ❑ HTML 标记由元素、属性和值构成。
- ❑ 通常全部使用小写字母编写 HTML（DOCTYPE 是一个例外），用引号包住属性值。
- ❑ 为文件和文件夹命名时全部采用小写字母，词与词之间用短横线分隔，而不要用空格或者下划线。
- ❑ 始终用下面的 DOCTYPE 声明开始 HTML 文档，让浏览器知道这是一个 HTML5 页面。

<!DOCTYPE html>

- ❑ 页面内容都在 body 元素中。主要为浏览器和搜索引擎准备的指令位于 body 元素之前，在 head 元素中。
- ❑ 用语义化 HTML 标记内容，不关心它在浏览器中显示的样式。
- ❑ 语义化 HTML 提升了网站的可访问性，让网站更有效率，并使网站维护和添加样式变得更容易。
- ❑ CSS 控制 HTML 内容的表现。
- ❑ 每个浏览器自带的样式表规定 HTML 的默认表现样式。开发人员可以使用自己写的 CSS 覆盖这些规则。

本章内容

- ❑ 规划网站
- ❑ 创建新的网页
- ❑ 保存网页
- ❑ 指定默认页面或主页
- ❑ 编辑网页
- ❑ 对文件进行组织
- ❑ 在浏览器中查看网页
- ❑ 借鉴他人灵感

开始编写 HTML 元素和属性代码之前，有必要了解如何创建使用这些代码的文件。本章将介绍如何创建、编辑和保存网页文件，还会涉及一些设计和组织方面的注意事项。

如果你已经知道如何创建文件，没有耐心读完本章，可以直接跳到讲解 HTML 代码的第 3 章。

2.1 规划网站

我们可以一上来就直接编写网页，但最好还是先对网站进行思考和规划。这样，你就不会迷失方向，而且也会减少将来的重组工作。比起简单地知道如何编写代码，了解如何创建有效的网站要更重要一些。下面介绍的内容并不全面，不过确已涵盖了不少需要考虑的方面。

规划网站的方法

- ❑ 确定为什么要创建这个网站，需要展示什么内容。
- ❑ 考虑网站的访问者。应该如何调整内容使之吸引这些访问者。
- ❑ 需要多少个页面，你希望网站是怎样的结构（参见图 2.1.1），你是希望访问者以某种特定的次序浏览网站，还是希望访问者可以自由地探索。

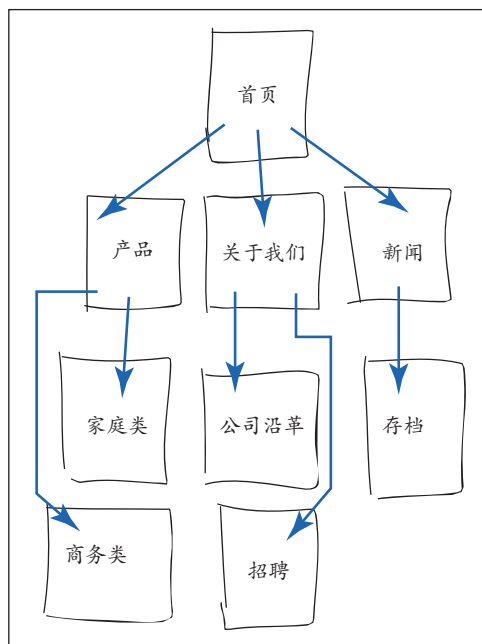


图 2.1.1 画出网站结构草图，考虑它可能包含的内容，这有助于创建者决定它需要何种结构

- ❑ 在纸上画出网站结构的草图，确定你在每个页面希望呈现的内容。与其他的工作不同，这项工作是能指导网站设计的。
- ❑ 为页面、图像和其他外部文件设计一个简单且一致的命名规则(参见1.6节)。

提示 如果你对万维网还不太熟悉，先上网逛逛，了解可能的网站形式。可以从竞争对手的网站开始入手。

提示 很多网站提供内容战略、用户体验(UX)、设计、开发等与建站相关的资源，其中最著名的两个是 A List Apart (www.alistapart.com) 和 Smashing Magazine (www.smashingmagazine.com)。

提示 Erin Kissane 的文章“A Checklist for Content Work”(www.alistapart.com/articles/a-checklist-for-content-work/) 讲解了如何制作网站内容。这部分内容也被收录到了她关于内容战略的一本书中。

提示 如果你还没有成为设计师，只是一位设计新手，正在寻找网站设计方面的指导，Jason Beaird 的《完美网页的视觉设计法则》(*The Principles of Beautiful Web Design*, SitePoint, 2010) 和 Mark Boulton 的《Web 设计实战》(*A practical Guide to Designing for the Web*, Five Simple Steps, 2009) 或许会引起你的兴趣。Boulton 的书最开始只有纸质书，现在又在网上发布了免费的版本(<http://designingfortheweb.co.uk/book/>)。这本书不仅讲解了设计理论，还介绍了 Web 设计者的工作方法。

提示 虽然不是必需的，但是将网站的文件夹结构与网站在纸上的组织结构对应起来是很常见的做法(图 2.1.1)，参见 2.6 节。

2.2 创建新的网页

创建网站并不需要特殊的工具。你可以使用任何文本编辑器，甚至是 Windows 自带的记事本 Notepad，如图 2.2.1 和图 2.2.3 所示，或者 OS X 上的免费软件 TextWrangler (www.barebones.com/products/textwrangler)，如图 2.2.1 和图 2.2.2 所示。(Mac 也自带一款编辑器，名为 TextEdit，但它在 OS X 某些版本中存在漏洞，使之无法正确处理 HTML 文件。)

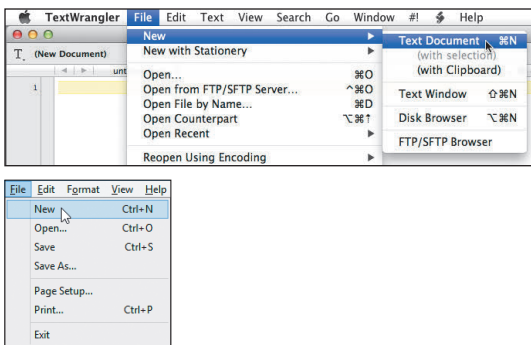


图 2.2.1 打开文本编辑器。在编辑器中出现的空白文档中输入 HTML，或者先选择 File → New。实际的菜单选项可能略有差异。如果使用 TextWrangler (Mac)，就是 File → New → Text Document (文件→新建→文本文档)，如上半部分所示。下半部分显示的是 Notepad (Windows) 的选项

创建新网页的步骤

- (1) 打开文本编辑器。
- (2) 选择 File → New (文件→新建) 创建一个新的空白文档，如图 2.2.1 所示。
- (3) 按照本书其余部分的解释(从第 3 章开始)创建 HTML 内容。
- (4) 一定要按照 2.3 节中的说明保存文件。

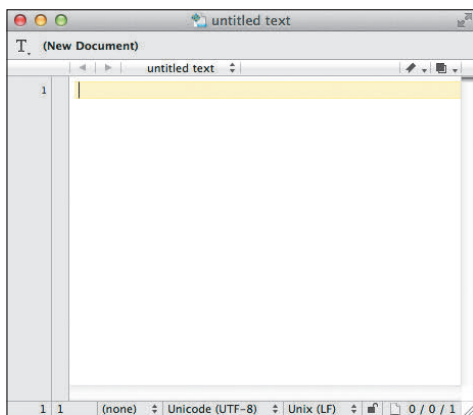


图 2.2.2 在 Mac 上, 可以使用 TextWrangler 编写网页的 HTML 代码。下文提示部分列出了一些功能更强的编写代码的 Mac 编辑器

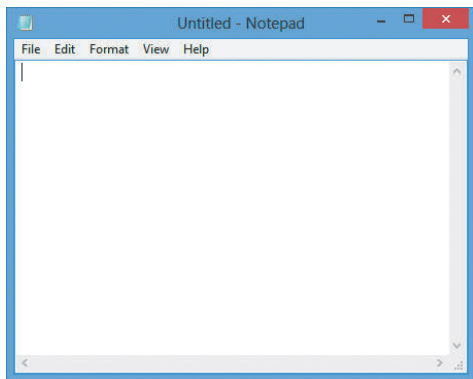


图 2.2.3 这是记事本, Windows 用户用以创建 HTML 页面的最基本的程序。也可以使用其他的编辑器 (参见提示)

提示 在 OS X 和 Windows 上都有很多专门编写 HTML (和 CSS) 代码的文本编辑器。它们的代码提示和代码补全功能可以让你可以更准确、更快速地编写代码; 它们会将代码突出显示, 方便区分 HTML 元素和其中的文本内容; 它们还有其他有用的功能。记事本没有这些功能。有不少免费的 HTML 编辑器, 不过一些收费的编辑器也值得购买, 通常还可以在购买之前试用。

提示 OS X 的流行编辑器包括 BBEdit (www.barebones.com/products/bbedit/)、Coda (www.panic.com/coda/)、Espresso (<http://macrabbbit.com/espresso/>)、Sublime Text (www.sublimetext.com) 以及 TextMate (<http://macromates.com>)。通常可以认为 TextWrangler 是 BBEdit Lite 的简化版。Sublime Text 也有 Windows 版本, Windows 下的编辑器还有很多, 如 Notepad++ (<http://notepad-plus-plus.org>), 等等。在网上搜索“HTML 编辑器”可以找到更多的编辑器。

提示 使用上面列出的某种编辑器创建新网页的方法也是类似的。要编辑已经存在的页面, 只需在文本编辑器中选择 File → Open (文件 → 打开) 并打开该文件 (参见 2.5 节)。

提示 请不要使用 Microsoft Word 这样的文字处理软件编写 HTML 页面。它们会在文件里添加一些无用的或无效的代码。

2.3 保存网页

用文本编辑器创建的网页需要在多种平台和多种设备上的多种浏览器查看。为了让所有这些程序都能访问网页, 网页需要保存为通用的“纯文本”格式, 不包含文字处理软件可能应用的任何专用格式化信息。

为了让浏览器将 HTML 文档作为网页识别并知道解释其中包含的标记, 网页文件应在文件名中使用 .html 或 .htm 作为扩展名。这样做也可以将网页文件与不是网页的普通文本文件区分开来。上述两种扩展名都可以, 但通常还是使用 .html, 因此推荐读者使用 .html 作为文件扩展名。

有了 .html 扩展名, 网页的图标会显示为

系统默认浏览器的图标，而不是用来编写这个文件的编辑器的图标（如图 2.3.1 所示）。双击网页文件会在浏览器中打开它，而不是在文本编辑器中。这对于在浏览器中测试页面很方便，但却为编辑网页增加了一个额外的步骤（参见 2.5 节）。

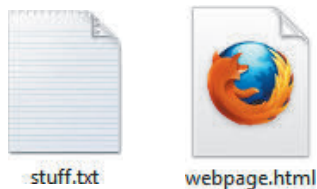


图 2.3.1 文本文件的扩展名是 .txt，在 Windows 上使用原生的文本文件图标进行标识（左图）。双击图标会在记事本中显示它的内容（如果你将文本文件跟其他的文本编辑器关联起来，就会启动那个软件）。对于网页文件（右图），不管是用哪个文本编辑器创建的，只要是以 .html 或 .htm 为扩展名，都会使用默认浏览器图标进行标识（在这里是 Firefox）。双击这个图标会在默认浏览器而不是文本编辑器中显示它

总之，保存文件时，需要将文件保存为纯文本格式，并使用 .html 或 .htm 作为扩展名。

保存网页的步骤

(1) 创建网页之后，在文本编辑器中选择 File → Save As（文件→另存为），如图 2.3.2 所示。

(2) 在随后弹出的对话框中，选择纯文本或文本文档（或别的叫法）作为文件格式。

(3) 为文档添加 .html 或 .htm 的扩展名（这一点非常重要）。

(4) 选择要保存网页的文件夹。

(5) 点击 Save（保存），如图 2.3.3 和图 2.3.4 所示。

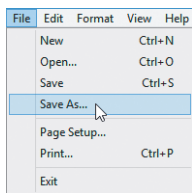


图 2.3.2 在文本编辑器中选择 File → Save As，这里显示的是记事本。TextWrangler 也有这个选项，不过位于一个更长的菜单中

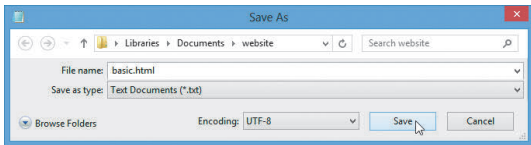


图 2.3.3 在记事本中，为文件名添加 .html 或 .htm 的扩展名，在 Save as type（保存类型）下拉菜单中选择文本文档，确保 Encoding（编码）选择的是 UTF-8（参见最后一项提示），点击 Save。其他文本编辑器中的选项可能并不相同（不过是相似的）

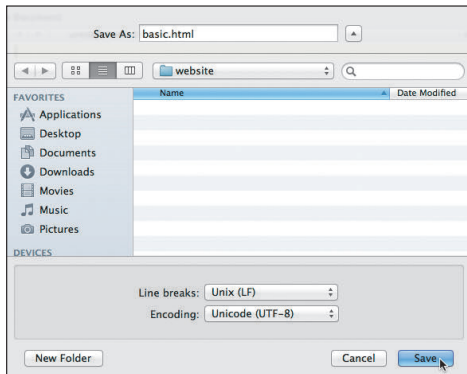


图 2.3.4 在 TextWrangler 中，为文件起名，选择保存位置。TextWrangler 默认以 UTF-8 进行编码（一般情况下，这就是应该选用的编码），不过你也可以从 Encoding 下拉菜单中选择其他编码。点击 Save 保存文件

提示 使用 .html 还是 .htm 并无区别，但推荐使用 .html，因为它更常见。无论你用哪种扩展名，请保持一致，因为使用相同的扩展名有助于后续记住 URL。

提示 即使已经指定了 .html 或 .htm 的扩展名，Windows 上的某些文本编辑器也会在文件名末尾加上默认扩展名。（注意大多数专门编辑 HTML 页面的编辑器并不存在这个问题。）这样文件名就变成了 webpage.html.txt，这样的文件无法在浏览器中查看。Windows 通常会隐藏扩展名，也导致这个问题变得隐蔽，它尤其容易困扰新手。有两个解决办法：一个是在首次保存文件时将文件名包围在双引号中，这样能防止添加额外的扩展名；另一个是让 Windows 显示文件扩展名（参见图 2.3.5 和图 2.3.6），从而可以看见程序自动添加的扩展名并将其删除。

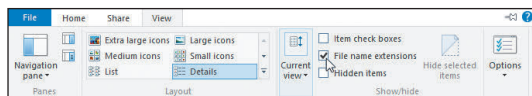


图 2.3.5 不同的 Windows 版本中菜单会有所不同（参见图 2.3.6）。在 Windows 8 中，可以从 Windows 资源管理器中选择 View（查看）选项卡找到这个菜单。如果想在桌面上和文件夹里看到文件的扩展名（如 .html），就要选取 File name extensions（让复选框打上勾）

提示 选择将文件保存为纯文本格式时，文件会以系统默认字符编码保存。如果需要创建其他编码的网页，需要使用可以选择编码类型的文本编辑器。通常，UTF-8 是最好的选择。如果编辑器中可选择文件编码类型包括 UTF-8, no BOM 或类似的选项，请选择该项。否则，就选择 UTF-8（参见图 2.3.7）。有时，编辑器的 UTF-8 模式并不包含 BOM，但编码类型选择菜单中并未显式指出这一点。（如果你对 BOM 的含义极感兴趣，可以参见 http://en.wikipedia.org/wiki/Byte_order_mark。做好看不懂的准备吧！）

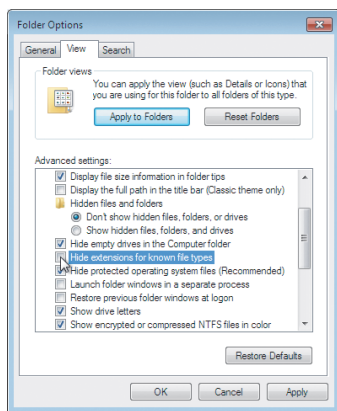


图 2.3.6 在 Windows 的早期版本中，这个菜单是上面这个样子的（根据 Windows 版本的不同，它们稍有差异）。这里是 Windows 7 中的菜单。在 Windows 资源管理器中，选择 Organize → Folder and search options（组织→文件夹和搜索选项），或者 Tools → Folder Options（工具→文件夹选项），具体取决于用户的 Windows 版本，显示图中的对话框。点击 View 选项卡，向下滚动，直到看到 Hide extensions for known file types（隐藏已知文件类型的扩展名）。如果想在桌面上和文件夹里看到文件的扩展名（如 .html），要确保这个选项是未选中的

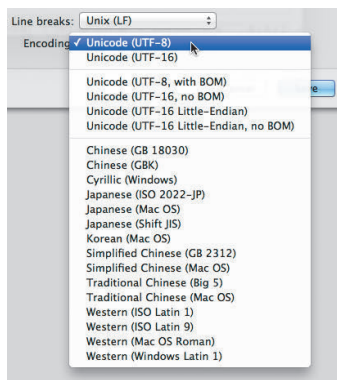


图 2.3.7 许多文本编辑器允许为文件选择编码，从而可以在同一个文档中保存不同语言中的符号和字符。大多数情况下，推荐选择 UTF-8。如果有 UTF-8, no BOM 的话就选这个选项，否则，直接选择 UTF-8。有的编辑器（如图中的 TextWrangler）会默认选择该项

2.4 指定默认页面或主页

大多数 Web 服务器都会根据文件名识别每个文件夹中的默认页面。大多数情况下，系统会将 `index.html` 识别为默认页面，参见图 2.4.1。如果没有 `index.html`，就会继续寻找 `index.htm`、`default.htm` 等文件名。如果访问者输入带目录的 URL，但没有指定文件名，那么就会打开默认页面，参见图 2.4.2。（目录就是文件夹，就像你自己计算机里的文件夹一样。）

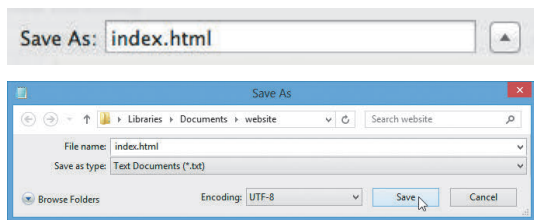


图 2.4.1 将文件保存为 `index.html`，从而指定这个文件是该目录下的默认页面（上图是 TextWrangler，下图是记事本）



图 2.4.2 当访问者输入目录的路径而没有指定文件名时，就会打开使用默认名称的页面。在这个例子中，输入的 URL 是 `http://htmlcssvqs.com/gaudi/`。如果输入 `http://htmlcssvqs.com/gaudi/index.html`，会显示相同的页面

在网站最顶层目录（通常称为根目录）中创建的默认页面是网站的主页。当访问者只输入域名而没有指定路径信息（如 `www.`

`yourdomain.com`）时，就会显示这个页面。如果输入 `www.yourdomain.com/index.html`，结果是一样的（这里假定主页的文件名是 `index.html`）。

类似地，可以为网站的任何一个目录（甚至每一个目录）创建默认页面。例如，网站中 `/products/` 或 `/about-us/` 目录的着陆页（即首页）也应该是 `index.html`，只不过它们放在各自的目录里。网站的访问者通常通过主页或每页都有的主导航访问这些页面。

指定网站主页或含目录的着陆页的方法

在目标文件夹中，将文件保存为 `index.html`。（当你将文件上传到网站的服务器以后，如果 `index.html` 无法起到默认页面的作用，请向 Web 主机提供商咨询。关于将网页上传到服务器，参见第 21 章。）

当默认页面不存在时

如果目录中没有默认页面，有的服务器就会将目录文件列表显示出来（你可能并不希望向访问者暴露这些内容）。为了防止这种情况的发生，应该在网站每一个包含 HTML 页面的目录下创建一个默认页面。另一种办法是修改服务器的配置，将文件列表隐藏起来（如果它们已被隐藏，也可以将它们显示出来）。对于包含图像、媒体文件、样式表以及 JavaScript 等资源的目录，建议隐藏文件列表。如果不知道如何修改服务器配置，请向 Web 主机提供商咨询。

2.5 编辑网页

因为网页在大多数情况下是通过浏览器查看的，所以在桌面上双击网页文件，会启动默认浏览器并显示它们。如果想编辑网页

文件，需要在文本编辑器中手动打开它。

编辑网页的步骤

- (1) 打开文本编辑器。
- (2) 选择 File → Open。
- (3) 找到包含目标文件的目录。
- (4) 如果没有看到目标文件，选择 All Files（全部文件）选项（或类似的叫法），如图 2.5.1 和图 2.5.2 所示。在不同的程序或平台，这个选项的叫法和位置都有细微的差异，如图 2.5.3 所示。
- (5) 点击 Open，就可以开始编辑文件了。

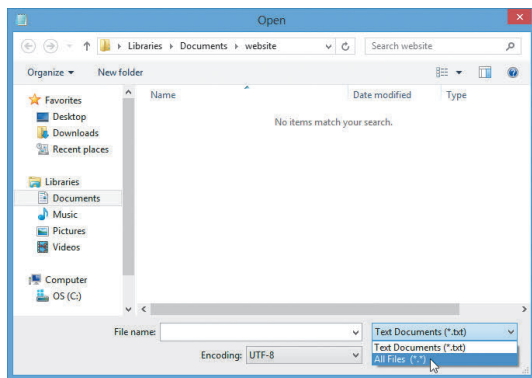


图 2.5.1 在 Windows 的一些文本编辑器（如记事本）中，不会自动看到 HTML 文件。选择 All Files 可以查看所有文件

提示 可以从文件所在的文件夹直接打开文件（如果文件在桌面上，也可以直接从桌面上打开）。

提示 通常，要保存对已有文档进行的修改，只需要选择 File → Save 即可，不必担心 2.3 节中提到的格式问题。使用快捷键保存文件更快。通常，编辑器中为保存设置的快捷键是 Command-S（对于 OS X）或 Ctrl-S（对于 Windows）。

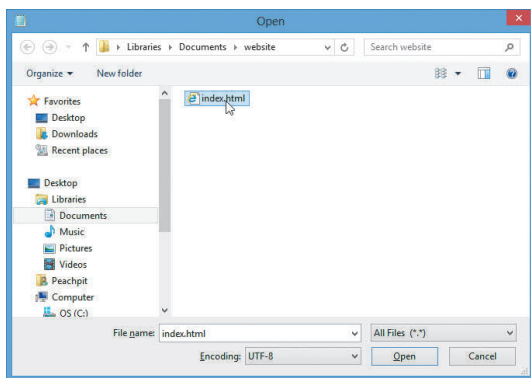


图 2.5.2 显示所有文件以后选中要编辑的 HTML 文件，点击 Open

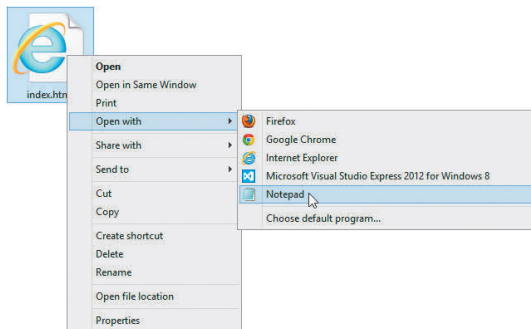


图 2.5.3 在 Windows 中，还可以右击文档的图标，在弹出菜单中选择 Edit 或 Open。在 Mac 上，右击图标或文件名，在弹出菜单中选择 Open With，再选择要使用的文本编辑器。在两种系统中，如果编辑器之前处于关闭状态，现在都会启动

2.6 组织文件

在文件数量变得很大之前，最好考虑好将它们放在什么地方。通常（但非必须）为网站的主要区块创建单独的文件夹，将相关的 HTML 页面放在一起。

组织文件的步骤

- (1) 创建一个主文件夹或目录存放网站上所有可用的资料。在 Mac 上，在 Finder 中选

择 File → New Folder（文件→新文件夹），如图 2.6.1 所示。在 Windows 中，右击桌面（或你选择的文件夹），选择 New → Folder（新建→文件夹），如图 2.6.2 所示。然后为文件夹命名。

(2) 根据网站的组织结构创建子文件夹，如图 2.6.1 和图 2.6.3 所示。例如，可以考虑为网站的每个部分创建单独的文件夹，并根据需要在其中创建单独的子文件夹。

(3) 为网站的图像、样式表（CSS 文件）和 JavaScript 文件创建一个或多个文件夹，每个文件夹还可以有各自的子文件夹。有很多种组织方式，具体做法完全取决于你自己。一种做法是像图 2.6.1 和图 2.6.3 中显示的那样进行组织，还有一种方法是将 CSS、JavaScript 文件夹同图像文件夹（及其他文件夹）一样放在根目录，也可以将这些文件夹一起放在根目录下的 assets（资源）文件夹里。

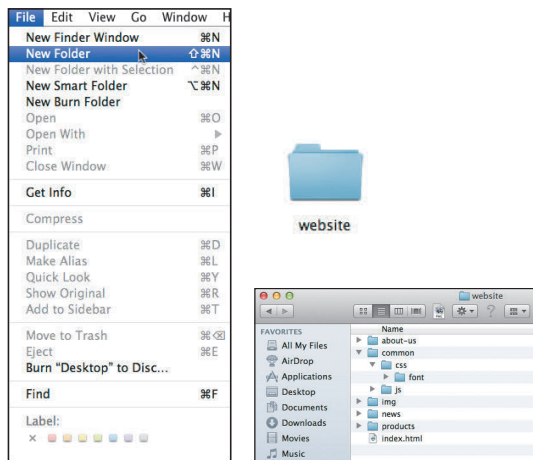


图 2.6.1 在 Mac 上，选择 New Folder，再为文件夹命名。为网站每个不同的部分创建独立的文件夹

提示 第 (2) 步和第 (3) 步是可选的，但推荐这样做。

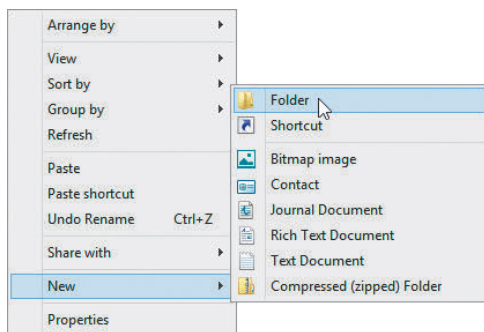


图 2.6.2 在 Windows 里，在桌面或资源管理器中右击，再选择 New → Folder

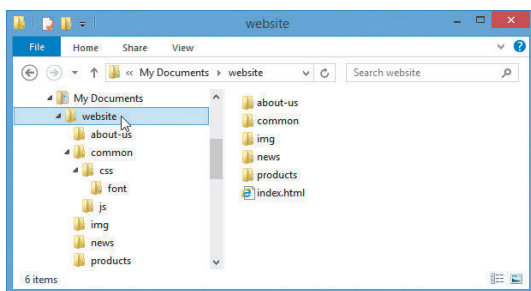


图 2.6.3 根据需要，可以将文件夹分解成多个子文件夹

提示 使用短的、描述性的名称作为文件和文件夹的名称，使用短横线（而不用空格）分隔不同的单词。全部使用小写字母可确保 URL 更容易输入，页面更容易被访问到。关于创建文件名的细节，参见 1.6 节。

2.7 在浏览器中查看网页

创建网页之后，你肯定希望看看它在浏览器中显示的样子。实际上，你并不知道访问者使用什么浏览器（不同的浏览器呈现页面的方式会有差异），因此建议使用多个浏览器查看页面（参见 20.2 节）。

在浏览器中查看网页的步骤

(1) 打开浏览器。

(2) 选择 File → Open File (在不同的浏览器中, Open File 项的表述可能不同, 但不管是什么, 不能选择 Open Location), 参见图 2.7.1。也可以使用快捷键 Command-O (对于 OS X) 或 Ctrl-O (对于 Windows)。使用快捷键会直接跳到第 (3) 步, 但 Internet Explorer 除外——IE 会先弹出一个如图 2.7.1 所示的对话框。

(3) 在弹出的对话框中, 找到目标文件所在的文件夹, 选中该文件, 点击 Open (参见图 2.7.2)。页面会显示在浏览器中 (参见图 2.7.3), 显示效果与发布到服务器的网页相同 (关于发布网页, 参见第 21 章)。对于不同的浏览器, 这些步骤可能有些差异。

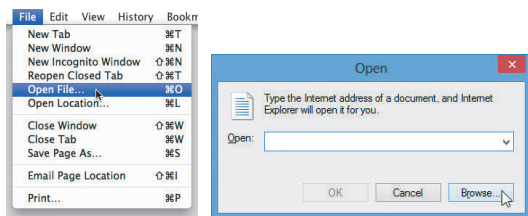


图 2.7.1 在 OS X 上, 从浏览器中选择 File → Open File (上图为 Chrome)。在 Internet Explorer 中使用 Ctrl-O, 在 Open (打开) 对话框中点击 Browse (浏览) 才能跳到下一步 (下图)

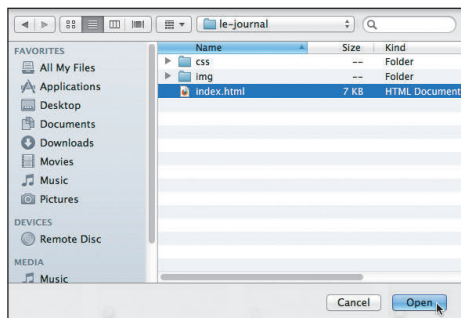


图 2.7.2 选择要打开的文件并点击 Open 按钮 (这里显示的是 OS X 上的对话框, 不过 Windows 上的是相似的)

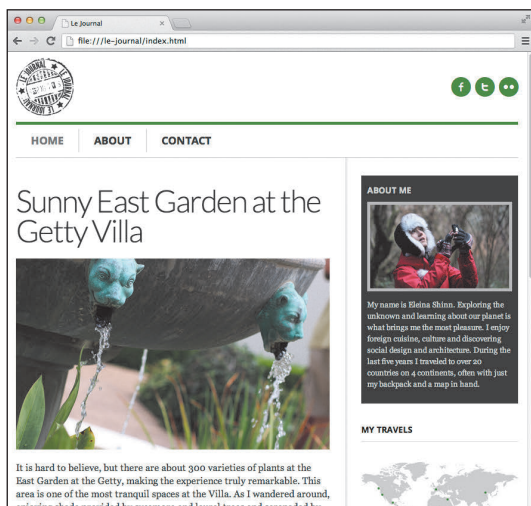


图 2.7.3 页面出现在浏览器中。仔细检查一下, 看看它是否符合预期

提示 通常还可以双击网页文件名或图标来查看该网页。如果你已经打开了浏览器, 还可以将网页图标拖到浏览器窗口, 这样也能打开。一旦掌握这种方法, 它往往是在浏览器中查看网页的最简单方法。

提示 一些现代浏览器没有 File → Open 这样的菜单选项来打开网页, 试试第 (2) 步中提到的快捷键, 或者使用上一条提示中提到的拖曳方法。

提示 如果网页没有出现在选取文件的对话框中 (图 2.7.2), 那么检查一下文件是否以纯文本格式保存, 且以 .html 或 .htm 为扩展名 (参见 2.3 节)。

提示 在浏览器中查看网页之前, 不需要在文本编辑器中关闭文档, 但一定要保存文件。在浏览器中打开网页之后, 如果又在文本编辑器中对页面作了修改, 那么需要再次保存

文件，并在浏览器中刷新页面。执行刷新操作最好使用浏览器的快捷键：Command-R（OS X）或 Ctrl-F5（Windows）。使用这些快捷键刷新网页可以避免浏览器使用缓存的网页（对于 Chrome，参见补充材料）。

提示 只有在 Web 服务器上发布网站（参见第 21 章）之后，访问者才能看到网站。

禁用 Chrome 缓存

Chrome 的缓存机制设计得有些特别。即便使用快捷键 Command-R（对于 OS X）或 Ctrl-F5（对于 Windows）刷新网页，有时还是会显示缓存的页面，而不是修改保存过的最新文件。查看网页的时候出现这种情况会产生误导，也可能让人烦躁。不过，幸好可以采取一些方法禁用 Chrome 缓存，这样就能总是显示最新的网页，方法如下。

(1) 在 Chrome 中按下 Command-Option-I（对于 OS X）或 Ctrl-Shift-I（对于 Windows），打开 Chrome 的 Developer Tools（开发者工具）。

(2) 点击右下角的齿轮图标。

(3) 在 Settings（设置）→ General（通用）中，选择 Disable Cache（禁用缓存）。

(4) 关闭 Settings 面板，但不关闭 Developer Tools。只要 Developer Tools 是打开状态的，就会禁用缓存（除非你取消选择 Disable Cache）。

顺便提一下，要让 Developer Tools 脱离 Chrome 窗口，显示为独立的窗口，可以点击 Developer Tools 左下角的图标。

2.8 借鉴他人灵感

学习其他网页开发和设计人员如何创建页面是提高 HTML 代码水平最容易的方法之一。幸好，我们很容易就可以查看和学习他人的 HTML 代码。不过，文本内容、图像、音频、视频、样式表及其他外部文件可能受版权保护。通常的做法是借鉴其他人的页面为自己的 HTML 寻找灵感，再创建自己的内容。

1. 使用 View Source 查看其他设计者的 HTML 代码

(1) 在浏览器中打开网页。

(2) 选择 View Source（查看源代码，或浏览器中的其他类似选项），如图 2.8.1 和图 2.8.2 所示。这时会显示 HTML 代码，如图 2.8.3 所示。

(3) 如果需要，可以保存该文件以作进一步研究。

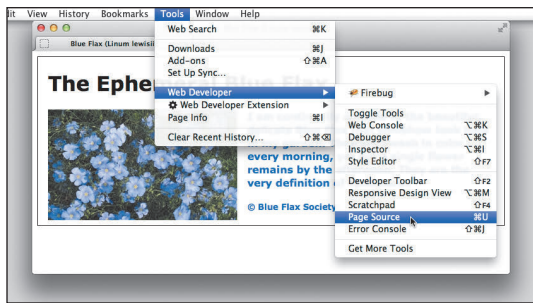


图 2.8.1 所有的桌面浏览器都有让你查看页面 HTML 源代码的菜单选项。选项的名称可能是 View Source，也可能是 Page Source（图中的 Firefox），或者其他类似的名称。在 Chrome 中为 Tools → View Page Source（工具 → 查看页面源代码）

2. 通过开发者工具查看其他设计者的 HTML 代码

查看网页源代码的另一种方法是使用浏览器的开发者工具。不同浏览器提供的开发者工具并不一样，但有些功能是一样的。

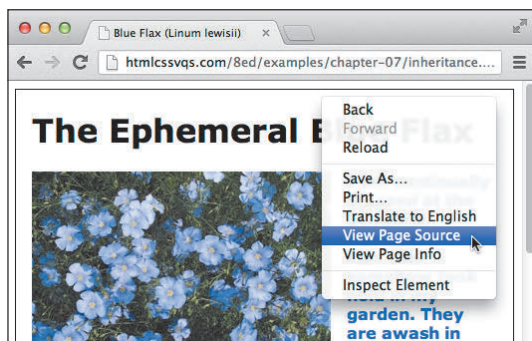


图 2.8.2 大多数浏览器还可以在页面上右击，然后在弹出的菜单中选择 View Source（或者别的类似叫法）。图上显示的是 Chrome 中的菜单。这通常是查看源代码最方便的方法，因为很难在主菜单或子菜单上找到该选项



图 2.8.3 现代浏览器在其自身的选项卡或窗口中显示源代码（如图所示），而旧的浏览器可能在特定的文本编辑器中显示。页面内容的颜色与 HTML 元素、属性和属性值的颜色是区分开来的。这称为语法高亮（syntax highlighting）。左侧显示的行号并不是 HTML 代码的一部分，也并非所有的浏览器都会在 View Source 模式中显示行号

这些工具提供了一种交互式的查看源代码的视图。可以审查页面特定部分的 HTML 和 CSS，在浏览器中编辑代码，并立即查看修改后的效果。开发者工具适用任何网站（不仅仅是自己的）。使用开发者工具作的修改都是临时的，这些工具并未真正修改页面的 HTML 和 CSS。开发者工具是学习 HTML 和

CSS 不可多得资料，通过它可以查看某个特定的效果是如何实现的，可以随意改动代码而不必担心破坏任何东西。

关于开发者工具（既有现代浏览器的，也有旧浏览器的）的信息，参见 20.1 节中的“浏览器开发者工具”。

提示 不存在关于谁能在万维网上发布网站的规则。这就是万维网的伟大之处——它是一种进入门槛很低的开放媒体。你可以是一名新手，一位专家，或者介于二者之间的角色。查看其他网站源代码时应该记住这一点，如果有些代码看起来并不怎么样，不要仅仅因为该网站放在网上就认为其制作者知道的比你多。有大量网站可以作为编码最佳实践的范例，也有大量网站做得并非那么理想。保持批判的态度，如果对某项技术的合理性有所怀疑，请查看本书或其他资源。

提示 要保存代码，可以将代码从 View Source 窗口中复制到文本编辑器中，再保存文件。在 OS X 上，使用 Command-A 选择所有代码，再使用 Command-C 复制代码。再切换到编辑器中，按下 Command-V 粘贴代码。在 Windows 上，过程是相似的，只是用 Ctrl 键代替 Command 键。

提示 要同时保存源代码及网页资源（如图像），在大多数浏览器中可以选择 File → Save As（或 File → Save Page As，文件 → 页面另存为）。不过，保存页面时，浏览器可能改写代码的某些部分，因此代码与使用上一条提示保存的代码并不完全一样。

提示 关于查看网页的 CSS，参见 8.7 节。

本章内容

- 开始编写网页
- 创建页面标题
- 创建分级标题
- 普通页面构成
- 创建页眉
- 标记导航
- 标记网页主要区域
- 创建文章
- 定义区块
- 指定附注栏
- 创建页脚
- 创建通用容器
- 使用 ARIA 改进可访问性
- 为元素指定类或 ID
- 为元素添加 title 属性
- 添加注释

本章讨论的是构建文档基础和结构所需的 HTML 元素，即网页内容主要的语义化容器。

你将学到以下内容：

- 开始编写网页；
- HTML5 的文档大纲；
- h1~h6、header、nav、main、article、section、aside、footer 及 div 元素（其中大多数是 HTML5 新增的元素）；
- 使用 ARIA 的 role 属性提升页面的可

访问性；

- 为元素应用类或 ID；
- 为元素应用 title 属性；
- 为代码添加注释。

创建清晰、一致的结构不仅可以为页面建立良好的语义化基础，也可以大大降低在文档中应用层叠样式表（CSS）的难度（从第 7 章开始讲解 CSS）。

3.1 开始编写网页

每个 HTML 文档都应该包含以下基本成分（如图 3.1.1 所示）。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <title></title>
</head>
<body>

</body>
</html>
```

图 3.1.1 这是每个 HTML 页面的基础。缩进并不重要，但结构很重要。在这个例子中，默认语言（由 lang 属性指定）被设为代表英语的 en。字符编码被设为 UTF-8

□ DOCTYPE

- html 元素（包含 lang 属性，该属性不是必需的，但推荐加上）
- head 元素
- 说明字符编码的 meta 元素
- title 元素
- body 元素

这份 HTML 等同于一张空白的纸，因为 body 里面没有任何内容，如图 3.1.2 所示。

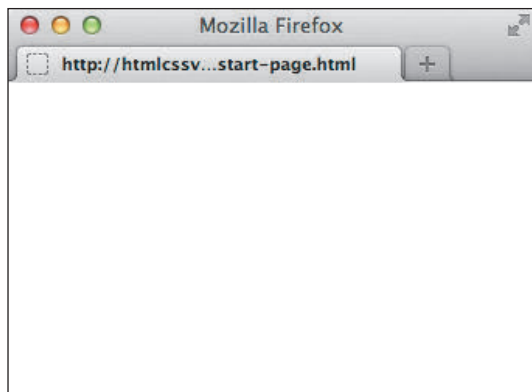


图 3.1.2 最少 HTML 基础代码在 Firefox 中显示的样子。如你所见，什么内容也没有！不过，很快就可以开始添加内容了

在添加任何内容或其他信息之前，需要先建立起页面的基础。

1. 编写 HTML5 页面开头的步骤

(1) 输入 `<!DOCTYPE html>`，声明页面为 HTML5 文档。（关于 HTML 早期版本的相关信息，参见本节末尾的“改进后的 HTML5 DOCTYPE”。）

(2) 输入 `<html lang="language-code">`，开始文档的实际 HTML 部分。其中，*language-code* 是页面内容默认语言的代码。例如，`<html lang="es">` 表示西班牙语，`<html lang="fr">` 表示法语。还可以写得更详细些，如 `<html lang="en-US">` 表示美国英语，而 `<html lang="en-GB">` 则表示英国英语。

(3) 输入 `<head>`，开始网页文档的头部。

(4) 输入 `<meta charset="utf-8"/>`（或 `<meta charset="UTF-8"/>`），将文档的字符编码声明为 UTF-8。空格和斜杠是可选的，因此 `<meta charset="utf-8">` 跟其他表达式形式都是有效的。（UTF-8 以外的其他字符编码也是有效的，不过 UTF-8 的适用面最广，很少有需要用其他编码的情况。）

(5) 输入 `<title></title>`。这里将包含页面的标题。3.2 节中将讲解如何添加标题文字。

(6) 输入 `</head>`，结束页面文档的头部。

(7) 输入 `<body>`，开始页面的主体。这里是放置页面内容的地方。

(8) 为页面内容预留一些空行。后续章节将讲解如何创建内容。

(9) 输入 `</body>`，结束主体。

(10) 输入 `</html>`，结束页面。

步骤似乎有点多，不过，由于所有的页面都是这样开始编写的，可以使用一个单独的 HTML 页面作为创建每个页面的模板，以节省输入的时间。实际上，很多代码编辑器都可以为新页面指定初始的代码，这样就更简单了。如果你在编辑器中没有找到 Settings（设置）或 Preferences（偏好设置）菜单，可以在 Help（帮助）中搜索。

2. 网页的两个部分：head 和 body

快速回顾一下第 1 章学到的知识，HTML 页面分为两个部分：head 和 body，如图 3.1.1 所示。DOCTYPE 出现在每个页面的开头，就像一本书的序言。

在文档 head 部分，通常要指明页面标题，提供为搜索引擎准备的关于页面本身的信息，加载样式表，以及加载 JavaScript 文件（不过，出于性能考虑，多数时候在页面底部 `</body>` 标签结束前加载 JavaScript 是更好的选择）。随着进一步阅读本书，你会看到一些关于这

些内容的例子。除了 title，其他 head 里的内容对页面访问者来说都是不可见的。

body 元素包住页面的内容，包括文本、图像、表单、音频、视频以及其他交互式内容，也就是访问者看见的东西。本书有好几章都讲述 HTML 的内容相关元素，本章会提前接触到其中的一些。

提示 使用 HTML5 DOCTYPE 可以确保浏览器以可靠的模式呈现页面，同时告诉 HTML 验证器根据 HTML5 允许的元素和语法检查代码。第 20 章会讨论 HTML 验证器。

提示 HTML5 的 DOCTYPE 不区分大小写。有的人可能输入 `<!doctype html>`，不过，使用 `<!DOCTYPE html>` 是更常规的做法，参见图 3.1.1。

提示 搜索引擎可能会根据 lang 属性指定的语言区分搜索结果，从而仅显示与搜索词语言相同的页面。屏幕阅读器也可能通过指定的语言调整其发音。

提示 要查找语言代码，可以使用 Richard Ishida 的语言标签查询工具（<http://rishida.net/utills/subtags/>）。

提示 要确保将你的代码编辑器配置为以 UTF-8 保存文件，与代码中通过 `<meta charset="utf-8" />` 语句指定的编码相同，参见图 3.1.1。（如果指定了另一种编码，就按那种编码保存文件。）并非所有的编辑器都默认以 UTF-8 保存文件，但大多数编辑器可以在菜单或面板中选择编码（参见 2.3 节）。如果页面没有设置为 UTF-8，有的字母（如带重音符的 i、带波字符（~）的 n）就会变成一些奇怪的字符。

提示 嵌套在 head 元素里的代码不一定要缩进，参见图 3.1.1。不过，缩进可以让你一眼看出 head 从哪里开始，包含什么内容，以及在哪里结束。在有些页面中，head 变得很长并不少见。

改进后的 HTML5 DOCTYPE

有了 HTML5 以后，编写代码的开头部分变得容易多了。特别是跟从前的 DOCTYPE 相比，HTML5 的 DOCTYPE 短得让人惊讶。

在 HTML 4 和 XHTML 1.0 时代，有好几种可供选择的 DOCTYPE，每一种都会指明 HTML 的版本，以及使用的是过渡型还是严格型模式。由于它们太难记，开发人员不得不每次都从某个地方把这些代码复制进来。

例如，下面是 XHTML 严格型文档的 DOCTYPE。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
→ "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

看起来令人费解。

幸好，所有的浏览器，无论新旧，都理解 HTML5 的 DOCTYPE，因此你可以坚持在所有页面中使用它，忘掉过去的那些 DOCTYPE。（唯一可能用到过去那些 DOCTYPE 的情况是继承一个旧站，而网站的拥有者不允许将 DOCTYPE 修改为 HTML5 的版本。）

3.2 创建页面标题

3.1 节 HTML 基础代码中 `<title></title>` 仅为占位符，现在开始讨论 `title` 元素。

每个 HTML 页面都必须有一个 `title` 元素。每个页面的标题都应该是简短的、描述性的，而且是唯一的，如图 3.2.1 所示。在大多数浏览器中，页面标题出现在窗口的标题栏（Chrome 是个例外）和浏览器的标签页中，如图 3.2.2 所示。页面标题还会出现在访问者浏览历史列表和书签里，如图 3.2.3 所示。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Antoni Gaudí - Introduction
</head>
<body>
</body>
</html>
```

图 3.2.1 `title` 元素必须位于 `head` 部分，将它放置在指定字符编码的 `meta` 元素后面

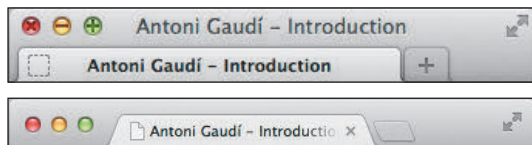


图 3.2.2 在大多数浏览器（如 Firefox）中，页面标题既显示在标题栏，也显示在标签页上。不过，Chrome（下图）只在标签页上显示页面标题

或许更为重要的是，页面标题会被 Google、Bing、DuckDuckGo、Yahoo! 等搜索引擎采用，从而能够大致了解页面内容，并将页面标题作为搜索结果中的链接显示，如图 3.2.4 所示。

总之，要让每个页面的 `title` 是唯一的，从而提升搜索引擎结果排名，并让访问者获得更好的体验。

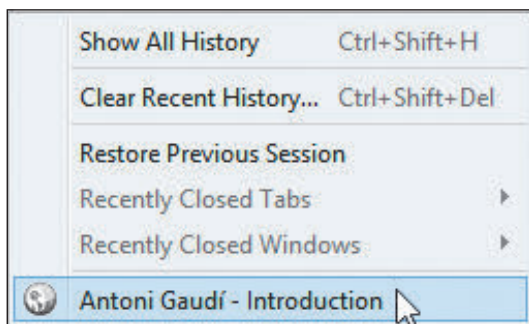


图 3.2.3 页面标题也出现在访问者的 History 面板（如图）、收藏夹列表以及书签列表中



图 3.2.4 或者最重要的是，页面标题通常是 Google 等搜索引擎的搜索结果中链接的文字，它也是判断搜索结果中页面相关度的重要因素。此处为 Google 中显示的页面标题和部分主体内容

创建页面标题的步骤

- (1) 将光标放在文档 `head` 中的 `<title>` 和 `</title>` 之间。
- (2) 输入网页的标题。

提示 `title` 元素是必需的。

提示 `title` 中不能包含任何格式、HTML、图像或指向其他页面的链接。

提示 创建新页面时，有的代码编辑器会预先为页面标题填上默认文字，除非已经按照 3.1 节中介绍的方法使用了特定的开头代码。要注意到这些默认文字，确保用自己的标题替换它们。

深入探讨页面标题

很多开发人员不太重视 title 元素，甚至那些相当用心、很有经验的开发人员也是这样。他们只是简单地输入网站名称，并将其复制到全站每一个网页。或者更糟糕，让 title 文字仍然保存为代码编辑器默认添加的文字。如果流量是网站追求的指标之一，这样做对建站者和潜在的访问者都会产生巨大的损害。

不同搜索引擎确定网页排名和内容索引规则的算法是不一样的。不过，title 通常都扮演着重要的角色。搜索引擎会将 title 作为判断页面主要内容的指标，并将页面内容按照与之相关的文字进行索引。有效的 title 应包含几个与页面内容密切相关的关键词。

作为一种最佳实践，选择能简要概括文档内容的文字作为 title 文字。这些文字既要为屏幕阅读器用户友好，又要有利于搜索引擎排名。

其次，将网站名称放入 title（这不是必需的）。常见的做法是将网站名称放在 title 的开头，不过将页面特有的文字放在开头更好。

建议将 title 的核心内容放在前 60 个字符（含空格）中，因为搜索引擎通常将超过此数目（作为基准）的字符截断。不同浏览器显示在标题栏中的字符数上限不尽相同。浏览器标签页会将标题截得更短，因为它占的空间较少。

3.3 创建分级标题

HTML 提供了六级标题用于创建页面信息的层级关系。使用 h1、h2、h3、h4、h5 或 h6 元素对各级标题进行标记，其中 h1 是最高

级别的标题，h2 是 h1 的子标题，h3 是 h2 的子标题，以此类推。为简洁起见，本书使用 h1 ~ h6 表示这些元素，不再逐一列出。

为了解 h1 ~ h6 标题，可以将它们比作学期论文、销售报告、新闻报道、产品手册等非 HTML 文档里的标题。当你撰写这些文章时，会根据需要为内容的每个主要部分指定一个标题和任意数量的子标题（以及子子标题，等等）。总之，这些标题代表了文档的大纲。网页的分级标题也是这样的（参见图 3.3.1 和图 3.3.2）。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8" />
  <title>Antoni Gaudí - Introduction
  → </title>
</head>
<body>
  <h1>Barcelona's Architect</h1>
  <h2 lang="es">La Sagrada Família</h2>
  <h2>Park Güell</h2>
</body>
</html>
```

图 3.3.1 用标题建立文档结构，就像大纲一样。这里，标记为 h2 的 La Sagrada Família 和 Park Güell 是标记为 h1 的顶级标题 Barcelona's Architect 的子标题。如果 Park Güell 是 h3，它就成了 La Sagrada Família 的子标题（也是 h1 的子子标题）。如果继续编写页面其余部分的代码，相关的内容（段落、图像、视频等）就要紧跟在对应的标题后面

1. 分级标题的重要性

对任何页面来说，分级标题都可以说是最重要的 HTML 元素。由于标题通常传达的是页面的主题，因此，对搜索引擎而言，如果标题与搜索词匹配，这些标题就会被赋予很高的权重，尤其是等级最高的 h1（这并不是说页面中的 h1 越多越好，搜索引擎还是足够聪明的）。

```

...
<body>
<h1>Product User Guide</h1>
  <h2>Setting it up</h2>

  <h2>Basic Features</h2>
    <h3>Video Playback</h3>
    <h4>Basic Controls</h4>
    <h4>Jumping to Markers</h4>

    <h3>Recording Video</h3>
    <h4>Manual Recording</h4>
    <h4>Scheduling a Recording</h4>

  <h2>Advanced Features</h2>
    <h3>Sharing Video</h3>
    <h3>Compressing Video</h3>
</body>
</html>

```

图 3.3.2 在这个例子中，产品指南有三个主要的部分，每个部分都有不同层级的子标题。标题之间的空格和缩进只是为了让层级关系更清楚一些，它们不会影响最终的显示效果。在实践中我通常不会添加这样的空格和缩进，如果你想添加可以这样做

对人来说，好的分级标题也很重要。视力良好的用户需要通过分级标题确定页面内容。屏幕阅读器用户也是这样，只不过是用手和耳朵。他们通常通过键盘按键在标题间移动，这样可以快速了解页面内容并去查看最感兴趣的内容，而不用把整个页面从头到尾听完。当他们找到感兴趣的标题之后，可以选择阅读该标题下的内容。可见 h1 ~ h6 对可用性和无障碍访问的作用是最大的。

总之，好的标题层级结构对你和你的访客来说都很重要。

2. 使用标题对网页进行组织的步骤

(1) 在 HTML 的 body 部分，输入 `<hn>`，其中 n 是 1 ~ 6 的数字，此数字取决于要创建的标题的级别。h1 是最重要的标题，而 h6 则是最不重要的标题。

(2) 输入标题的内容。

(3) 输入 `</hn>`，其中 n 是与第 (1) 步中相同的数字。

提示 在默认情况下，浏览器会从 h1 到 h6 逐级减小标题的字号，如图 3.3.3 所示。（在有的浏览器里，嵌套在特定元素中的 h1 和 h2 在默认情况下看起来是一样的。）不过别忘了要依据内容所处的层次关系选择标题级数，而不是根据你希望文字应该显示的大小。可以按照你希望的样子为标题添加样式，包括字体、字号、颜色等。用 CSS 实现这些样式的详细说明参见第 10 章。



图 3.3.3 在默认情况下，所有的标题都以粗体显示，h1 的字号比 h2 的大，而 h2 的又比 h3 的大，以此类推。每个标题之间的间隔也是由浏览器默认的 CSS 制造的，它们并不代表 HTML 文档中有空行

提示 创建分级标题时，要避免跳过某些级别，如从 h3 直接跳到 h5。不过，允许从低级别跳到高级别的标题。例如，在图 3.3.2 中，`<h4>Scheduling a Recording</h4>` 后面紧跟着 `<h2>Advanced Features</h2>` 是没有问题的，因为包含 Scheduling a Recording 的 Basic Features（也是 h2）在这里结束了，而 Advanced Features 的内容开始了。

提示 不要使用 h1 ~ h6 标记副标题、标语以及无法成为独立标题的子标题。例如，假设有一篇新闻报道，它的主标题后面紧跟着一个副标题，这时，这个副标题就应该使用段落（参见图 3.3.4）或其他非标题元素。

提示 先前，HTML5 包含了一个名为 hgroup 的元素，用于将连续的标题组合在一起，W3C 将这个元素从 HTML5 规范中移除了。

```
...
<h1>Giraffe Escapes from Zoo</h1>
<p class="subhead">Animals Worldwide
→ Rejoice</p>

<p>... story content ...</p>
...
```

图 3.3.4 这是标记文章副标题的一种方法。可以添加一个 class，从而能够应用相应的 CSS。该 class 可以命名为 subhead 等名称

提示 另外注意一下，图 3.3.1 中对一个 h2 使用了 lang 属性，用以表明这里的文字为页面默认语言（英语，因为 `<html lang="en">`）之外的另一种语言（西班牙语，由 es 表示）。

提示 在本书写作之际，曾有人提议在 HTML 中引入 subhead 元素，用于对子标题、副标题、标语、署名等内容进行标记。目前还无从知道这个提议是否会被采纳。

HTML5 的文档大纲

在本书写作之际，HTML5 对如何处理位于 article、aside、nav 和 section 元素中的 h1 ~ h6 有一套算法。该算法通常称为 HTML5 文档大纲（HTML5 document outline）。不过，目前还没有浏览器实现这套算法，而且也看不到支持的迹象。而且，屏幕阅读器中只有 JAWS（一款运行于 Windows 下的屏幕阅读器）支持，而它的实现还存在问题。

鉴于此，W3C 已经将文档大纲列入可能从 2014 年最终定稿的规范中移除的特性。即便文档大纲最终留在规范中，或者浏览器将其实现了，我们还是可以按照这里介绍的方法对分级标题进行标记。这种方法不仅适用于当前环境，还足以应对未来（因为文档大纲不会对页面造成破坏）。

由此看来，你大可不必理会文档大纲。这里提到文档大纲只是为了让你在其他地方看到时不必犹豫是否需要了解其机理。

3.4 普通页面构成

你肯定已经访问过了大量像图 3.4.1 中显示的那种网站。抛开内容不谈，我们可以看

到该页面有四个主要组件：带导航的页头、显示在主体内容区域的文章、显示次要信息的附注栏以及页脚，如图 3.4.2 所示。

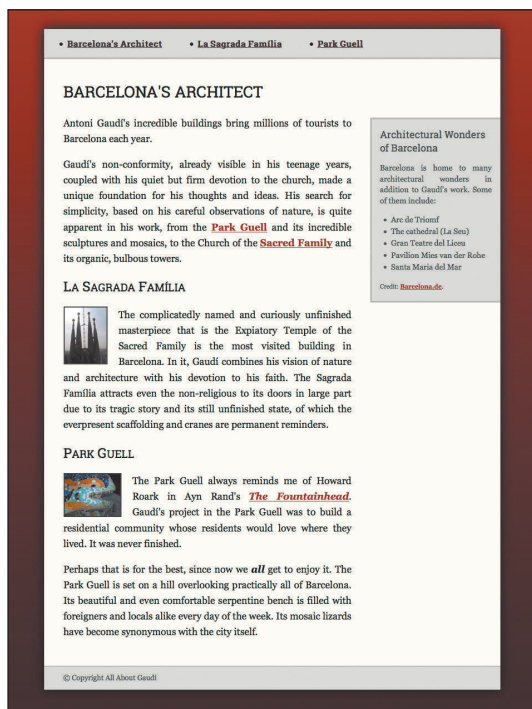


图 3.4.1 一个普通的布局，顶部是主导航，左侧是主要内容，右侧是附注栏，底部是页脚。要让页面成为这个样子，需要添加 CSS



图 3.4.2 页面的常规信息类型。这只是一种安排方式，不过是很常见的一种

现在，在没有引入 CSS 的情况下，你还无法像这样为页面添加样式。我们将从第 7 章开始学习 CSS，在第 10 章开始学习如何对文字进行格式化、添加颜色等，在第 11 章学习多栏布局。

不过，应用到这些常规页面结构的语义都是非常相似的，与布局无关。本章剩余章节的大部分内容都会讲解这些结构。按照从页面顶端向下的顺序，将依次讲解用 header、nav、main、article、section、aside 和 footer 定义页面的结构，以及用来添加额外样式信息或实现其他目的的通用容器 div。除了 div 以外，这些元素都是 HTML5 推出后才有的。

在学习这些元素的过程中，不要关心它们在示例布局中的位置，而应该关注它们的语义。

接下来，我们还会先大概了解一下其他元素，如 ul（无序列表）和 a（链接），后续章节会详细讲解。

3.5 创建页眉

如果页面中有一块包含一组介绍性或导航性内容的区域，应该用 header 元素对其进行标记。

一个页面可以有任意数量的 header 元素，它们的含义可以根据其上下文而有所不同。例如，处于页面顶端或接近这个位置的 header 可能代表整个页面的页眉（有时称为页头），如图 3.5.1 所示。通常，页眉包括网站标志、主导航（图 3.5.2）和其他全站链接，甚至搜索框（图 3.5.3）。这无疑是 header 元素最常见的使用形式，不过不要误认为是唯一的形式。

```

...
<body>
<header role="banner">
  <nav>
    <ul>
      <li><a href="#gaudi">Barcelona's
        → Architect</a></li>
      <li lang="es"><a href="#sagrada-
        → familia">La Sagrada Familia</a>
        → </li>
      <li><a href="#park-guell">Park
        → Guell</a></li>
    </ul>
  </nav>
</header>
</body>
</html>

```

图 3.5.1 这个 header 代表整个页面的页眉。它包含一组代表整个页面主导航的链接（在 nav 元素中）。可选的 role="banner" 并不适用于所有的页眉。它显式地指出该页眉为页面级的页眉，因此可以提高可访问性。图 3.5.4 中也显示了一个这样的例子。要了解更多信息，参见 3.13 节（对于 nav 元素的 role 属性值，参见 3.6 节）



图 3.5.2 包含导航的页面级页眉

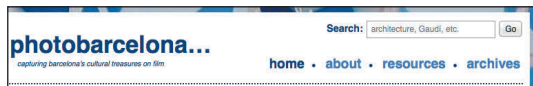


图 3.5.3 这是另一个网站的页眉（包含样式）。这种页面级页眉的形式在网上很常见。它包含网站名称（通常为一个标识）、指向网站主要板块的导航链接以及一个搜索框

header 也很适合对页面深处的一组介绍性或导航性内容进行标记。例如，一个区块的目录，参见图 3.5.4。

```

...
<body>
<header role="banner">
  ... [网站标识、导航等] ...
</header>

<main role="main">
<article>
  <header>
    <h1>Frequently Asked Questions</h1>
    <nav>
      <ul>
        <li><a href="#answer1">What is
          → your return policy?</a></li>
        <li><a href="#answer2">How do I
          → find a location?</a></li>
      </ul>
    </nav>
  </header>

  <!-- 第二个header的链接指向这里 -->
  <article id="answer1">
    <h2>What is your return policy?</h2>
    <p> ... [答案] ... </p>
  </article>

  <article id="answer2">
    <h2>How do I find a location?</h2>
    <p> ... [答案] ... </p>
  </article>
  ...
</article> <!-- 结束父元素article -->
</main>
</body>
</html>

```

图 3.5.4 这个页面有两个 header，一个是整个页面的，另一个是 Frequently Asked Questions 的父元素 article 的。注意第一个并不包含任何 h1 ~ h6 标题，而第二个则有。而且，只有第一个 header 中有 role="banner"，因为它是页面级的页眉

header 通常包含其自身的标题（h1 ~ h6），但这并不是强制性的。例如，图 3.5.3 中有标题，但图 3.5.1 中就没有。

创建页眉的步骤

(1) 将光标放置在需要创建页眉的元素里。

(2) 输入 `<header>`。

(3) 输入页眉的内容，包括各种类型的内容，它们分别由各自的 HTML 元素（本书余下部分将会讲到其中的大多数）进行标记。例如，`header` 可以包含 `h1 ~ h6` 标题、标识、导航、搜索框，等等。

(4) 输入 `</header>`。

提示 只在必要时使用 `header`。大多数情况下，如果使用 `h1 ~ h6` 能满足需求，就没有必要用 `header` 将它包起来。

提示 `header` 与 `h1 ~ h6` 元素中的标题（参见 3.3 节）是不能互换的。它们都有各自的语义目的。

提示 不能在 `header` 里嵌套 `footer` 元素或另一个 `header`，也不能在 `footer` 或 `address` 元素里嵌套 `header`。

提示 `header` 不一定要像示例那样包含一个 `nav` 元素（图 3.5.1 和图 3.5.4），不过在大多数情况下，如果 `header` 包含导航性链接，就可以用 `nav`。在图 3.5.4 所示的例子中，`nav` 包住 Frequently Asked Questions 链接列表是恰当的，因为它是页面内的主要导航组（将在 3.6 节中讨论）。

提示 要了解 `header` 如何取代 HTML5 之前版本中 `div` 元素的一个功能，参见 3.12 节。

3.6 标记导航

HTML 的早期版本没有元素明确表示主导航链接的区域，而 HTML5 则有这样一个元素，即 `nav`。`nav` 中的链接可以指向页面中的

内容，如图 3.6.1 所示，也可以指向其他页面或资源，或者两者兼而有之。无论是哪种情况，应该仅对文档中重要的链接群使用 `nav`。

```
...
<body>
<header role="banner">
  <nav role="navigation">
    <ul>
      <li><a href="#gaudi">Barcelona's
        → Architect</a></li>
      <li lang="es"><a href="#sagrada-
        → familia">La Sagrada Família</a>
        → </li>
      <li><a href="#park-guell">Park
        → Guell</a></li>
    </ul>
  </nav>
</header>
</body>
</html>
```

图 3.6.1 这些链接（`a` 元素）代表一组重要的导航，因此将它们放入一个 `nav` 元素。`role` 属性并不是必需的，不过它可以提高可访问性。更多有关 `nav` 应用 `role="navigation"` 的信息参见本节最后一个提示

如果你仔细看过上一节的代码，就已经见识过 `nav` 元素了。下面将那一段代码搬过来，并对 `nav` 进行了突出显示。`nav` 元素不会对其内容添加任何默认样式，如图 3.6.2 所示。



图 3.6.2 在默认情况下，我们的导航看起来相当普通。那些圆点并不是由 `nav` 元素产生的。除了开启一个新行以外，该元素没有任何默认样式。显示这些圆点是因为每个链接都包在一个 `li` 元素（列表项）里面。使用 CSS 可以隐藏这些圆点或显示其他的东西，还可以将这些链接水平显示，改变它们的颜色，让它们看起来像按钮，等等。本书将从第 7 章开始讲解 CSS

将一组链接指定为重要导航

(1) 输入 `<nav>`。

(2) 输入一组链接并将其标记为 `ul`（无序列表）结构，除非链接的顺序比较重要（例如面包屑导航）。如果链接顺序重要，就将其标记为 `ol`（有序列表）结构。（要了解链接和列表，分别参见第6章和第15章。）

(3) 输入 `</nav>`。

提示 有编写 HTML 或 XHTML 经历的开发人员或许已经习惯了使用 `ul` 或 `ol` 元素对链接进行结构化。在 HTML5 中，`nav` 并没有取代这种最佳实践。应该继续使用这些元素，只是在它们的外围简单地包上一个 `nav`（参见图 3.6.1）。

提示 尽管屏幕阅读器整体上还在追赶 HTML5 新出现的语义功能，但 `nav` 能帮助它们识别页面的主导航，并允许用户通过键盘直接跳至这些链接。这可以提高页面的可访问性，提升访问者的体验。

提示 HTML5 规范不推荐对辅助性的页脚链接（如“使用条款”、“隐私政策”等）使用 `nav`，这不难理解。不过，有时页脚会再次显示顶级全局导航，或者包含“商店位置”、“招聘信息”等重要链接。在大多数情况下，我们推荐将页脚中的此类链接放入 `nav` 中。

提示 HTML5 不允许将 `nav` 嵌套在 `address` 元素中。

提示 要了解如何在 `nav` 中使用 `role="navigation"`（参见图 3.6.1），请参见 3.13 节。

深入了解 nav

上文提到，要在页面中插入一组链接并非意味着一定要将它们包在 `nav` 元素里。

设想有一个如图 3.6.3 所示的新闻页面。这是一篇文章的 Arts & Entertainment 部分，是一则关于某画廊开业的短文。该页面包含四个链接列表，其中只有两个列表具有足够的重要性，可以包在 `nav` 中。（可以看到，这里省略了部分代码。）

位于 `aside` 中的次级导航允许用户跳转到 Arts & Entertainment 中的分类页面（如 Movies、Music），因此它构成了页面的一个主要导航区块。不过，Other Stories 这个 `aside` 则不是，`footer` 里的链接也不是。（关于 `aside` 元素，参见 3.10 节。）

那么，如何判断是否对一组链接使用 `nav` 呢？归根结底，这取决于内容的组织情况。至少，应该将网站全局导航（让用户可以跳至网站各个主要部分的导航）标记为 `nav`。这种 `nav` 通常（但并不总是）出现在页面级的 `header` 元素里面（参见 3.5 节）。

```

...
<body>
  <!-- ==== 开始页面级页眉 ==== -->
  <header role="banner">
    <!-- 站点标识可以放在这里 -->
    <!-- 全站导航 -->
    <nav role="navigation">
      <ul> ... </ul>
    </nav>
  </header>

  <!-- ==== 开始主要内容 ==== -->
  <main role="main">
    <h1>Arts & Entertainment: Museums
    → </h1>

    <article>
      <h2>Gallery Opening Features the
      → Inspired, Inspiring</h2>
      <p>... [故事内容] ... </p>
    </article>

    <aside>
      <h2>Other Stories</h2>
      <!-- 没有包含在nav里 -->
      <ul> ... [故事链接] ... </ul>
    </aside>
  </main>

  <!-- ==== 开始附注栏 ==== -->
  <aside>
    <!-- 次级导航 -->
    <nav role="navigation">
      <ul>
        <li><a href="/arts/movies/">
        → Movies</a></li>
        <li><a href="/arts/music/">
        → Music</a></li>
        ...
      </ul>
    </nav>
  </aside>

  <!-- ==== 开始页面级页脚 ==== -->
  <footer role="contentinfo">
    <!-- 辅助性链接并未包在nav中 -->
    <ul> ... </ul>
  </footer>
</body>
</html>

```

图 3.6.3 在这个页面中,只有两组链接放在 nav 里,另外两组则由于不是主要的导航而没有放在 nav 里

3.7 标记页面的主要区域

大多数网页都有一些不同的区块,如页眉、页脚、包含额外信息的附注栏、指向其他网站的链接,等等。不过,一个页面只有一个部分代表其主要内容。可以将这样的内容包在 main 元素中,该元素在一个页面仅使用一次(参见图 3.7.1 和图 3.7.2)。

```

...
<body>
  <header role="banner">
    <nav role="navigation">
      ... [包含多个链接的ul] ...
    </nav>
  </header>

  <main role="main">
    <article>
      <h1 id="gaudi">Barcelona's Architect
      → </h1>
      <p>Antoni Gaudí's incredible
      → buildings bring millions ...</p>

      ... [页面主要区域的其他内容] ...
    </article>
  </main>

  <aside role="complementary">
    <h1>Architectural Wonders of Barcelona
    → </h1>

    ... [附注栏的其他内容] ...
  </aside>

  <footer role="contentinfo">
    ... [版权] ...
  </footer>
</body>
</html>

```

图 3.7.1 这是本章反复使用的页面的完整代码的主体结构。你可能还不熟悉其中某些元素,不过随后就会讲到。main 元素包围着代表页面主题的内容。最好在 main 开始标签中加上 role="main" (参见最后一条提示)

提示 main 元素是 HTML5 新添加的元素。记住,在一个页面里仅使用一次。



图 3.7.2 同 p、header、footer 等元素一样，main 元素的内容显示在新的一行，除此之外不会影响页面的任何样式。（这里显示的布局并不是由 main 产生的，而是由 CSS 控制的。）这里为了方便你在应用了样式的完整页面中识别出 main，特意为其添加了蓝色的边框

提示 如果创建的是 Web 应用，则使用 main 包围其主要的功能。

提示 不能将 main 放置在 article、aside、footer、header 或 nav 元素中。

提示 如图 3.7.1 所示，role="main" 可以帮助屏幕阅读器定位页面的主要区域。要进一步了解这样的 ARIA 地标，参见 3.13 节。

3.8 创建文章

HTML5 的另一个新元素便是 article，如图 3.8.1 和图 3.8.2 所示。你已经见过一些用到

该元素的示例。现在，让我们深入了解一下它的作用。

```
...
<body>
<header role="banner">
  <nav role="navigation">
    ... [包含多个链接的ul] ...
  </nav>
</header>

<main role="main">
  <article>
    <h1 id="gaudi">Barcelona's Architect
    → </h1>

    <p>Antoni Gaudí's incredible buildings
    → bring millions of tourists to
    → Barcelona each year.</p>

    <p>Gaudí's non-conformity, already
    → visible in his teenage years,
    → coupled with his quiet but firm
    → devotion to the church, made a
    → unique foundation for his thoughts
    → and ideas. His search for simplicity
    → ...is quite apparent in his work,
    → from the <a href="#park-guell">Park
    → Guell</a> and its incredible
    → sculptures and mosaics, to...</p>

    <h2 id="sagrada-familia" lang="es">
    → La Sagrada Família</h2>

    <p> The
    → complicatedly named and curiously
    → unfinished masterpiece...</p>

    <h2 id="park-guell">Park Guell</h2>

    ... [图像和段落] ...
  </article>
</main>
</body>
</html>
```

图 3.8.1 为了精简，我对文章内容进行了缩写，略去了与上一节相同的 nav 代码。可以在本书网站查看完整的代码（www.htmlcssvqs.com/8ed/structure-final）。尽管在这个例子里只有段落和图像，但 article 可以包含各种类型的内容

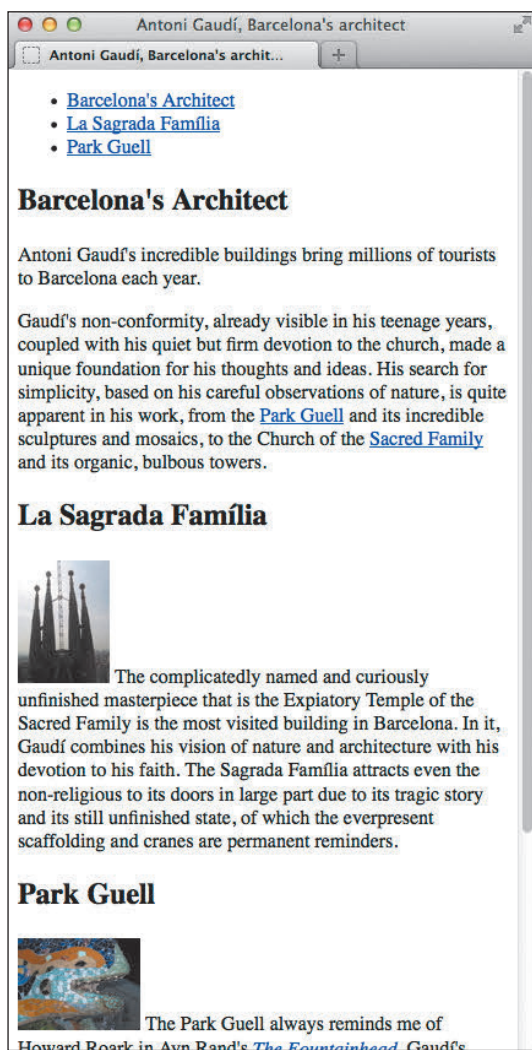


图 3.8.2 现在，页面有了 header、nav、main 和 article 元素，以及它们各自的内容。在不同的浏览器中，article 中标题的字号可能不同。可以应用 CSS 使它们在不同的浏览器中显示相同的大小（参见第 10 章）

根据其称，你大概会猜想 article 用于包含像报纸文章一样的内容。不过，article 并不局限于此。在 HTML5 中，“文章”（article）并非这个词本来的意思。

HTML5 对该元素的定义如下：

article 元素表示文档、页面、应用或网站中一个独立的容器，原则上是可独立分配或可再用的，就像聚合内容中的各部分。它可以是一篇论坛帖子、一篇杂志或报纸文章、一篇博客条目、一则用户提交的评论、一个交互式的小部件或小工具，或者任何其他独立的内容项。

其他 article 的例子包括电影或音乐评论、案例研究、产品描述，等等。你或许惊讶于它还可以是交互式的小部件或小工具，不过这些确实是独立的、可再分配的内容项。

创建文章的步骤

(1) 输入 <article>。

(2) 输入文章的内容。内容可以包含任意数量的元素。元素类型包括段落、列表、音频、视频、图像、图形等。

(3) 输入 </article>。

提示 可以将 article 嵌套在另一个 article 中，只要里面的 article 与外面的 article 是部分与整体的关系。下文提供了这样的例子。

提示 一个页面可以有多个 article 元素（也可以没有）。例如，博客的主页通常包括几篇最新的文章，其中每一篇都是其自身的 article。

提示 一个 article 可以包含一个或多个 section 元素（下一节会介绍该元素）。在 article 里包含独立的 h1 ~ h6 也是很好的做法，就像示例代码那样。

更多 article 示例

图 3.8.1 中的示例只是使用 `article` 的一种方式，下面看看其他的用法。

图 3.8.3 展示了对基本的新闻报道或报告进行标记的方法。注意 `footer` 和 `address` 元素的使用（对它们的讨论分别参见本章和第 4 章）。这里，`address` 只应用于其父元素 `article`（即这里显示的 `article`），而非整个页面或任何嵌套在那个 `article` 里面的 `article`（如图 3.8.4 中的读者评论）。

图 3.8.4 展示了嵌套在父元素 `article` 里面的 `article` 元素。该例中嵌套的 `article` 是用户提交的评论，就像你在博客或新闻网站上见到的评论部分。该例还显示了 `section` 元素（参见 3.9 节）和 `time` 元素（在第 4 章讨论）的用法。

这些只是使用 `article` 及有关元素的几个常见方式。

```
...
<article>
  <h1>The Diversity of Papua New Guinea
  → </h1>
  <p>Papua New Guinea is home to more than
  → 800 tribes and languages ...</p>

  ... [其余内容] ...

  <!-- 文章的页脚, 并非页面级的页脚 -->
  <footer>
    <p>Leandra Allen is a freelance
    → journalist who earned her degree
    → in anthropology from the University
    → of Copenhagen.</p>

    <address>
      You may reach her at <a href="mailto:
      → leandra@therunningwriter.com">
      → leandra@therunningwriter.com</a>.
    </address>
  </footer>
</article>
...
```

图 3.8.3 标记包含作者信息的文章的常见方式

```
...
<article>
  <h1>The Diversity of Papua New Guinea
  → </h1>
  ... [父元素article内容] ...

  <footer>
    ... [父元素article页脚] ...
  </footer>

  <section>
    <h2>Reader Comments</h2>
    <article>
      <footer>travelgal wrote on
      → <time datetime="2014-02-26">
      → February 26, 2014</time>:
      → </footer>
      <p>Great article! I've always
      → been curious about Papua New
      → Guinea.</p>
    </article>

    <article>
      ... [下一则读者评论] ...
    </article>
  </section>
</article>
...
```

图 3.8.4 每条读者评论都包含在一个 `article` 里，这些 `article` 元素则嵌套在主 `article` 里

3.9 定义区块

另一个 HTML5 的新元素是 `section`，参见图 3.9.1 至图 3.9.3。HTML5 对该元素的部分定义如下：

`section` 元素代表文档或应用的一个一般的区块。在这里，`section` 是具有相似主题的一组内容，通常包含一个标题。

`section` 的例子包含章节、标签式对话框中的各种标签页、论文中带编号的区块。比如网站的主页可以分成介绍、新闻条目、联系信息等区块。

尽管我们将 `section` 定义成“通用的”区块，但不要将它与 `div` 元素（参见 3.12 节）混淆。从语义上讲，`section` 标记的是页面中的特定区域，而 `div` 则不传达任何语义。

```
...
<body>
...
<main role="main">
  <h1>Latest World News</h1>
  <section>
    <h2>Breaking News</h2>
    <ul>... [标题列表] ...</ul>
  </section>

  <section>
    <h2>Business</h2>
    <ul>... [标题列表] ...</ul>
  </section>

  <section>
    <h2>Arts</h2>
    <ul>... [标题列表] ...</ul>
  </section>
</main>
...
</body>
</html>
```

图 3.9.1 几乎任何新闻网站都会对新闻进行分类。每个类别都可以标记为一个 `section`

```
...
<h1>Graduation Program</h1>
<section>
  <h2>Ceremony</h2>
  <ol>
    <li>Opening Procession</li>
    <li>Speech by Valedictorian</li>
    <li>Speech by Class President</li>
    ...
  </ol>
</section>

<section>
  <h2>Graduates (alphabetical)</h2>
  <ol>
    <li>Molly Carpenter</li>
    ...
  </ol>
</section>
...
```

图 3.9.2 这个例子是在 HTML5 规范中一个例子的基础上修改而成的，其中，`section` 元素用于标记一次毕业活动安排中的不同部分

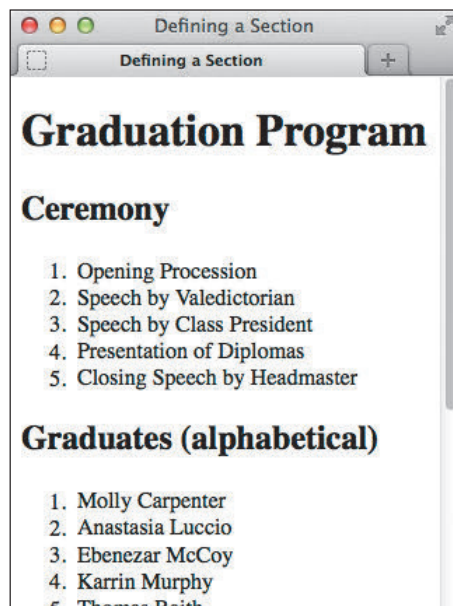


图 3.9.3 跟本章很多其他的元素一样，`section` 并不影响页面的显示。这里用数字编号展示各个条目是因为图 3.9.2 中使用了有序列表（`ol`）。关于列表，参见第 15 章

定义区块的步骤

(1) 输入 `<section>`。

(2) 输入区块的内容。内容可以包含任意数量的元素。元素类型包括段落、列表、音频、视频、图像、图形等。

(3) 输入 `</section>`。

提示 如果只是出于添加样式的原因要对内容添加一个容器，应使用 `div` 而不是 `section`。

提示 可以将 `section` 嵌套在 `article` 里，从而显式地标出报告、故事、手册等文章的不同部分或不同章节。例如，可以在本章所示 Antoni Gaudí 的例子中使用 `section` 元素，即使用一个 `section` 包围 La Sagrada Família 这个 h2 及相关的段落，用另一个 `section` 包围 Park Guell 这个 h2 及相关的段落。

体会 section 元素

我特意引用了 HTML5 对 `section` 的定义，以便你对该元素有个最为直观的理解。`section` 元素可能是 Web 社区讨论最激烈的主题，因为似乎很难解释清楚它的用法。

在考虑何时使用 `section` 的时候，记住定义中“具有相似主题的一组内容”这一条是很有帮助的。这也是 `section` 区别于 `div` 的另一个原因。`section` 和 `article` 的区别在于，`section` 在本质上组织性和结构性更强，而 `article` 代表的是自包含的容器。

正如在第 1 章中提到的，对内容进行标记时，并非总能分出对和错，只是大多数时候有正确的选择。而其他时候，就只能依靠个人对最适合描述内容的 HTML 元素的感觉了。

在考虑是否使用 `section` 的时候，一定要仔细思考，不过也不必每次都对是否用对感到担心。有时，些许主观并不会影响页面正常工作。而且，也不会有人半夜来敲你的门。

哦，我可能会，不过这只是因为外面很黑，很吓人。

3.10 指定附注栏

有时候，页面中有一部分内容与主体内容相关性没有那么强，但可以独立存在（参见图 3.10.1 和图 3.10.2），如何在语义上表示出来呢？

在 HTML5 之前一直无法显式地做到这一点。在 HTML5 中，我们有了 `aside` 元素。

使用 `aside` 的例子还包括重要引述、侧栏（图 3.10.3）、指向相关文章的一组链接（通常针对新闻网站）、广告、`nav` 元素组（如博客的友情链接），Twitter 源、相关产品列表（通常针对电子商务网站），等等。

尽管我们很容易将 `aside` 元素看做侧栏，

但该元素其实可以用在页面的很多地方，具体依上下文而定。如果 `aside` 嵌套在页面主要内容内（而不是作为侧栏位于主要内容之外），

则其中的内容应与其所在的内容密切相关，而不是仅与页面整体内容相关。“其他 `aside` 的例子”中的例 1 就是这样的。

```
...
<body>
<header role="banner">
  <nav role="navigation">
    ... [包含多个链接的ul] ...
  </nav>
</header>

<main role="main">
  <article>
    <h1 id="gaudi">Barcelona's Architect</h1>

    ... [文章的其他部分] ...
  </article>
</main>

<aside role="complementary">
  <h1>Architectural Wonders of Barcelona</h1>

  <p>Barcelona is home to many architectural wonders in addition to Gaudi's work. Some of them include:</p>
  <ul>
    <li lang="es">Arc de Triomf</li>
    <li>The cathedral <span lang="es">(La Seu)</span></li>
    <li lang="es">Gran Teatre del Liceu</li>
    <li lang="es">Pavilion Mies van der Rohe</li>
    <li lang="es">Santa Maria del Mar</li>
  </ul>

  <p><small>Credit: <a href="http://www.barcelona.de/en/barcelona-architecture-buildings.html"
  → rel="external"><cite>Barcelona.de</cite></a></small></p>
</aside>
</body>
</html>
```

图 3.10.1 这个 `aside` 是有关巴塞罗那建筑奇迹的信息，与页面主要关注的 Antoni Gaudi 内容相关性稍差，且可以在没有这个上下文的情况下独立存在。我可以将它嵌套在 `article` 里面，因为它们是相关的，但最终我将它放在了 `article` 后面，使用 CSS 让它看起来像侧栏（图 3.10.3）。`aside` 里面的 `role="complementary"` 是可选的，但它可以提高可访问性。要了解更多信息，参见最后一条提示

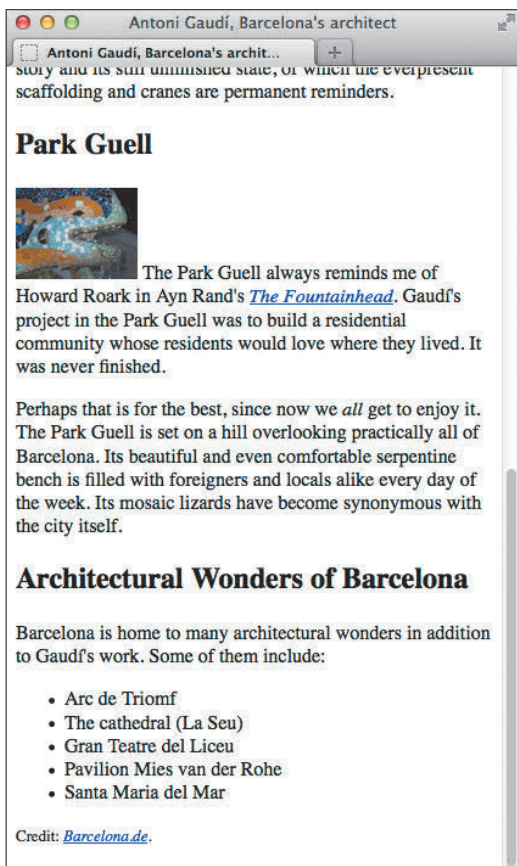


图 3.10.2 `aside` 出现在文章的下面，因为在 HTML 中 `aside` 就跟在 `article` 的后面，参见图 3.10.1。可以看到，浏览器在默认情况下并未对 `aside` 应用任何特殊样式（除了从新行开始）。不过，可以通过 CSS 控制其外观显示（图 3.10.3）

指定附注栏的步骤

(1) 输入 `<aside>`。

(2) 输入附注栏的内容。内容可以包含任意数量的元素。元素类型包括段落、列表、音频、视频、图像、图形等。

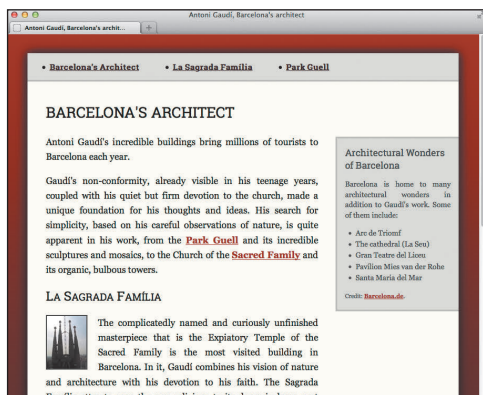


图 3.10.3 对已完成的页面应用 CSS 时，可以让 `aside`（以 `Architectural Wonders of Barcelona` 开头）显示在主要内容的旁边，而不是下边。这个例子已将 `aside` 做成了侧栏（我们将在第 11 章学习如何制作两栏的布局）

(3) 输入 `</aside>`。

提示 在 HTML 中，应该将附注栏内容放在 `main` 的内容之后（图 3.10.1）。出于 SEO 和可访问性的目的，最好将重要的内容放在前面。可以通过 CSS 改变它们在浏览器中的显示顺序。

提示 对于与内容有关的图像（如图表、图形或带有说明文字的插图），使用 `figure`（参见第 4 章）而非 `aside`。

提示 HTML5 不允许将 `aside` 嵌套在 `address` 元素内。

提示 要了解如何在 `aside` 中使用 `role="complementary"`，参见 3.13 节。

其他 aside 的例子

上面提到，aside 可以嵌套在主要内容里面，也可以位于主要内容外面。

例 1（嵌套在主要内容中）

```
...
<body>
<main role="main">
  <article>
    <h1>The Diversity of Papua New Guinea</h1>
    ... [ 文章内容 ] ...
    <aside>
      <h2>Papua New Guinea Quick Facts</h2>
      <ul>
        <li>The country has 38 of the 43 known birds of paradise</li>
        <li>Though quite tropical in some regions, others occasionally
        → experience snowfall.</li>
        ...
      </ul>
    </aside>
    ... [ 更多文章内容 ] ...
  </article>
</main>
</body>
</html>
```

如果这个 article 要包含一个重要引述，也应该包在 aside 里。也可以弄一个“相关故事”的 aside，包含一组指向关于该国家的其他文章的链接。另外，aside 还可以成为页面的另一个区块，而不是嵌套在 main 中的 article 里。

我们已看到了 aside 位于附注栏的例子（图 3.10.1 和图 3.10.3）。现在，考虑一套设计作品或一组案例研究，其中每个 HTML 页面展示一个项目，并在毗邻的一栏中（显示位置由 CSS 控制，而不是由例 2 中的代码控制）提供指向其他项目页面的链接（嵌套在 nav 中）。

例 2（aside 并未嵌套在主要内容中，且包含一个 nav）

```
...
<body>
<main role="main">
  <article>
    <h1>... [ 项目名称 ] ...</h1>
    <figure>... [ 项目图片 ] ...</figure>
    <p>... [ 项目报道 ] ...</p>
  </article>
</main>
```

```
<!-- 这个附注栏没有嵌套在主要内容中 -->
<aside>
  <h2>Other Projects</h2>
  <nav>
    <ul>
      <li><a href="habitat-for-humanity.html">Habitat for Humanity
        → brochure</a></li>
      <li><a href="royal-philharmonic.html">Royal Philharmonic Orchestra
        → website</a></li>
      ...
    </ul>
  </nav>
</aside>
</body>
</html>
```

这个 `aside` 位于 `article` 的外面，因为它与页面整体内容相关，但与 `article` 的内容相关性没那么强。

3.11 创建页脚

当你想到页脚的时候，你大概想的是页面底部的页脚（通常包括版权声明，可能还包括指向隐私政策页面的链接以及其他类似的内容）。HTML5 的 `footer` 元素可以用在这样的地方，但它同 `header` 一样，还可以用在其他的地方。

`footer` 元素代表嵌套它的最近的 `article`、`aside`、`blockquote`、`body`、`details`、`fieldset`、`figure`、`nav`、`section` 或 `td` 元素的页脚。只有当它最近的祖先是 `body` 时，它才是整个页面的页脚（参见图 3.11.1 和图 3.11.2）。

如果一个 `footer` 包着它所在区块（如一个 `article`）的所有内容，它代表的是像附录、

索引、版权页、许可协议这样的内容。

创建页脚的步骤

- (1) 将光标放在希望创建页脚的元素里。
- (2) 输入 `<footer>`。
- (3) 输入页脚的内容。
- (4) 输入 `</footer>`。

提示 页脚通常包含关于它所在区块的信息，如指向相关文档的链接、版权信息、作者及其他类似条目。参见“其他 `footer` 示例”中的前两个例子。

提示 页脚并不一定要位于所在元素的末尾，不过通常是这样的。

```
...
<body>
<header role="banner">
  <nav role="navigation">
    ... [包含链接的无序列表] ...
  </nav>
</header>

<main role="main">
  <article>
    <h1 id="gaudi">Barcelona's Architect
    → </h1>
    ... [文章的其他部分] ...

    <h2 id="sagrada-familia" lang="es">
    → La Sagrada Família</h2>
    ... [图像和段落] ...

    <h2 id="park-guell">Park Guell</h2>
    ... [其他图像和段落] ...
  </article>
</main>

<aside role="complementary">
  <h1>Architectural Wonders of Barcelona
  → </h1>
  ... [附注栏的其他部分] ...
</aside>

<footer>
  <p><small>&copy; Copyright All About
  → Gaudí</small></p>
</footer>
</body>
</html>
```

图 3.11.1 这个 footer 代表整个页面的页脚，因为它最近的祖先是 body 元素。现在，页面有了 header、nav、main、article、aside 和 footer 元素。并非每个页面都需要以上所有元素，但它们确实代表了 HTML 中的主要页面构成要素

提示 不能在 footer 里嵌套 header 或另一个 footer。同时，也不能将 footer 嵌套在 header 或 address 元素里。

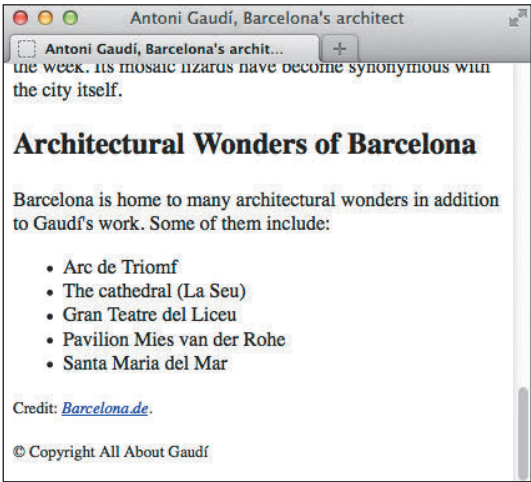


图 3.11.2 这个 footer 出现在页面的最底部，位于 aside 之后。footer 元素本身不会为文本添加任何默认样式。这里，版权信息的字号比普通文本的小，这是因为它嵌套在“用来对法律文本进行语义化”的 small 元素里（参见第 4 章）。像其他内容一样，可以通过 CSS 修改 footer 元素所含内容的字号

提示 要了解 footer 如何取代 HTML5 之前版本中 div 元素的一个功能，参见 3.12 节。

提示 要了解在哪种情况下可对页面级的 footer 使用 role="contentinfo"，参见 3.13 节。将其放入图 3.11.1 中的代码是合理的，因为它代表的是整个页面的页脚，不过我有意略去了，避免给读者留下所有 footer 元素都需要 role="contentinfo" 的印象。在“其他 footer 示例”中，我们在不同的示例中体现了这种区别，另外还演示了使用 role 属性的正确方法。

其他 footer 示例

我们已经学习了页面级 footer 的一个小例子（图 3.11.1 和图 3.11.2）。图 3.11.3 是另一个页面级 footer 的例子，不过其中的内容更多一些。

图 3.11.4 展示了位于页面区块 article 中的一个 footer，以及作为整个页面页脚的另一个 footer。（对 address 元素的解释，参见“更多 article 示例”。）

注意只有页面级页脚有 `role="contentinfo"`。参见 3.13 节了解此功能。

```
...
<body>
... [包含全局导航的页面级页眉] ...
... [页面内容] ...

<!-- 这是一个页面级的页脚，因为它最近的祖先元素
      是body -->
<footer role="contentinfo">
  <p><small>&copy; Copyright 2014 The
    → Corporation, Inc.</small></p>

  <ul>
    <li><a href="terms-of-use.html">
      → Terms of Use</a></li>
    <li><a href="privacy-policy.html">
      → Privacy Policy</a></li>
  </ul>
</footer>
</body>
</html>
```

图 3.11.3 页面级的 footer 通常包含版权信息以及不属于页面全局导航（位于 header 里）的一些链接

```
...
<body>
...
<article>
  <h1>... [文章标题] ...</h1>
  <p>... [文章内容] ...</p>

  <!-- 文章的页脚 -->
  <footer>
    <p>Leandra Allen is a freelance
      → journalist who earned her
      → degree in anthropology from the
      → University of Copenhagen.</p>
    <address>
      You may reach her at
      → <a href="mailto:leandra@
        → therunningwriter.com">
        → leandra@therunningwriter.com
        → </a>.
    </address>
  </footer>
</article>

<!-- 页面级页脚 -->
<footer role="contentinfo">
  ... [版权信息等] ...
</footer>
</body>
</html>
```

图 3.11.4 第一个 footer 包含在 article 内，因此是属于该 article 的页脚。第二个 footer 是页面级的。只能对页面级的 footer 使用 `role="contentinfo"`，且一个页面只能使用一次

3.12 创建通用容器

有时需要在一段内容外围包一个容器，从而可以为其应用 CSS 样式或 JavaScript 效果。如果没有这个容器，页面就会不一样（参见图 3.12.1）。在评估内容的时候，考虑使用 `article`、`section`、`aside`、`nav` 等元素，却发现它们从语义上讲都不合适。

这时，你真正需要的是一个通用容器，一个完全没有任何语义含义的容器。这个容器就是 `div`（来自 `division` 一词）元素，如图 3.12.2 所示。有了 `div`，就可以为其添加样式（参见图 3.12.3）或 JavaScript 效果了。一定要阅读补充材料“有关 `div` 的一些历史以及何时在 HTML5 中使用它”，了解什么时候可以在页面中使用 `div`。

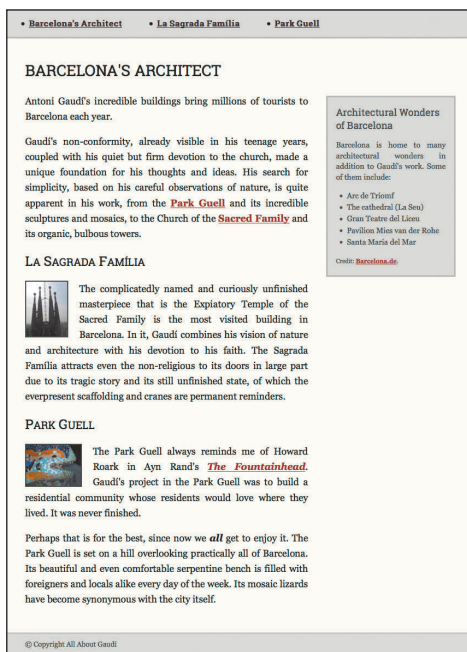


图 3.12.1 在不使用 `div` 元素的情况下，我们实现了这个设计。不过，为页面内容加上 `div`（参见图 3.12.2）以后，我们有了一个可以添加更多样式的通用容器（效果如图 3.12.3 所示）

```
...
<body>
<div>
  <header role="banner">
    <nav role="navigation">
      ... [包含多个链接的无序列表] ...
    </nav>
  </header>

  <main role="main">
    <article>
      <h1 id="gaudi">Barcelona's
      → Architect</h1>
      ... [文章的其余内容] ...

      <h2 id="sagrada-familia" lang="es">
      → La Sagrada Família</h2>
      ... [图像和段落] ...

      <h2 id="park-guell">Park Güell</h2>
      ... [另一个图像和段落] ...
    </article>
  </main>

  <aside role="complementary">
    <h1>Architectural Wonders of
    → Barcelona</h1>
    ... [aside的其余内容] ...
  </aside>

  <footer role="contentinfo">
    <p><small>&copy; Copyright All About
    → Gaudí</small></p>
  </footer>
</div>
</body>
</html>
```

图 3.12.2 现在有一个 `div` 包着所有的内容。页面的语义没有发生改变，但现在我们有了一个可以用 CSS 添加样式的通用容器（图 3.12.3）



图 3.12.3 `div` 元素自身没有任何默认样式，只是其包含的内容从新的一行开始（图 3.12.4）。不过，我们可以对 `div` 添加样式以实现自己的设计。这里，我对 `div` 添加了浅的背景色和阴影，这样我就可以将 `body` 元素的背景改为渐变的红色，使内容凸显出来。要查看实现这个页面的 HTML 和 CSS，可以访问 www.htmlcssvqs.com/8ed/structure-final

创建通用容器的步骤

- (1) 输入 `<div>`。
- (2) 创建容器的内容，内容可以包含任意数量的元素。
- (3) 在容器的结尾处输入 `</div>`。

提示 同 `header`、`footer`、`main`、`article`、`section`、`aside`、`nav`、`h1` ~ `h6`、`p` 和其他很多元素一样，默认情况下，`div` 从新的一行开始显示（图 3.12.4）。

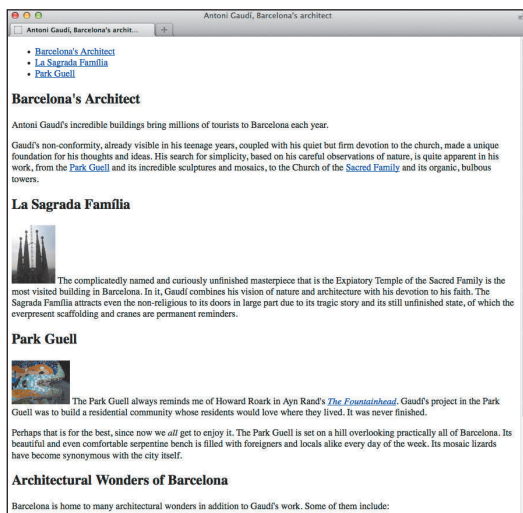


图 3.12.4 同一个页面在未对 `div`、标题、段落及其他元素添加 CSS 的样子。可以看到，`div` 并未产生任何独特的样式

提示 `div` 对使用 JavaScript 实现一些特定的交互行为或效果也是有帮助的。例如，在页面中展示一张照片或一个对话框，同时让背景页面覆盖一个半透明的层（这个层通常是一个 `div`）。

提示 尽管我始终强调 HTML 用于对内容的含义进行描述，但 `div` 并不是唯一没有语义价值的元素。`span` 是与 `div` 对应的一个元素：`div` 是块级内容的无语义容器，而 `span`（写作 `<span>` 这里是内容 `</span>`）则是短语内容的无语义容器，例如它可以放在段落元素 `p` 之内。参见第 4 章了解 `span`。

提示 要了解如何在 `div` 中使用地标角色，参见 3.13 节。

有关 div 的一些历史以及何时在 HTML5 中使用它

在本章讲到的结构性元素中，div 是除了 h1 ~ h6 以外唯一早于 HTML5 出现的元素。在 HTML5 之前，div 是包围大块内容（如页眉、页脚、主要内容、插图、附注栏等）从而可用 CSS 为之添加样式的不二选择。之前 div 没有任何语义含义，现在也一样。

这就是 HTML5 引入 header、footer、main、article、section、aside 和 nav 的原因。这些类型的构造块在网页中普遍存在，因此它们可以成为具有独立含义的元素。在 HTML5 中，div 并没有消失，只是使用它的场合变少了。

让我们看几个使用 div 的常见示例。

我们之前已经见过一个了：为添加样式，用一个容器将整个页面包起来（参见图 3.12.2 和图 3.12.3）。

如何用 div 实现两栏布局呢？我对 article 和 aside 元素分别添加一些 CSS，就能让它们各自成为一栏（我们将从第 7 章开始学习 CSS；关于用 CSS 进行布局，参见第 11 章）。

然而，大多数情况下，每一栏都有不止一个区块的内容。例如，主要内容区第一个 article 下面可能还有另一个 article（或 section、aside 等）。又如，也许你想在第二栏再放一个 aside 显示指向关于 Gaudí 的其他网站的链接，或许再加一个其他类型的元素。

你需要将期望出现在同一栏的内容包在一个 div 里（图 3.12.5），然后对这个 div 添加相应的样式。（你或许在想是否也可以用 section，但要知道该元素并不能作为添加样式的通用容器。）图 3.12.6 有助于你形象地理解代码和期望呈现的 CSS 布局之间的关系。记住这只是这段 HTML 可能形成的布局之一。要知道 CSS 是相当强大的。

因此，将希望在样式上组成一栏的内容用 div 包起来是一种非常常见的做法（当然，也可以用 div 包住两个以上的栏）。至于放在其中的内容，则可能差别极大，视页面内容安排而定。别忘了，作为内容区块的主要语义化容器，article、section、aside 和 nav 可以位于任何地方。此外，header 和 footer 也是这样。在图 3.12.5 和图 3.12.6 中，main 中只有 article 元素，附注栏只有 aside 元素，但不要对此作过分解读。

不过，可以肯定的是，div 应该作为最后一个备用容器，因为它没有任何语义价值。大多数时候，使用 header、footer、main（仅使用一次）、article、section、aside 甚至 nav 代替 div 会更合适。但是，如果语义上不合适，也不必为了刻意避免使用 div 而使用上述元素。有用得上 div 的地方，只是需要限制其使用。

说到这里，有一种情况，div 适合所有（或几乎所有，这由你决定）页面容器，而无需使用新的 HTML5 元素。更多信息参见 11.3 节。

```
...
<body>
<!-- 开始页面容器 -->
<div class="container">
  <header role="banner">
    ...
  </header>

  <!-- 应用CSS后的第一栏 -->
  <main role="main">
    <article>
      ...
    </article>

    <article>
      ...
    </article>

    ... [更多的区块] ...
  </main>
  <!-- 结束第一栏 -->

  <!-- 应用CSS后的第二栏 -->
  <div class="sidebar">
    <aside role="complementary">
      ...
    </aside>

    <aside role="complementary">
      ...
    </aside>

    ... [更多的区块] ...
  </div>
  <!-- 结束第二栏 -->

  <footer role="contentinfo">
    ...
  </footer>
</div>
<!-- 结束页面容器 -->
</body>
</html>
```

图 3.12.5 这个页面有一个包含整个页面的div，另外还新增了一个包着附注栏的div。带有class="sidebar"（使用任何类名都可以）的div包围着想要显示为第二栏的内容。然后可以在CSS中通过类定位到这样的div并对其应用样式。main则加上了显示为第一栏的样式

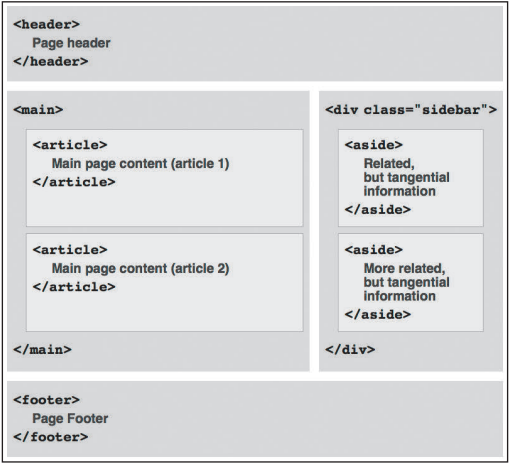


图 3.12.6 此图显示了图 3.12.5 中的代码如何在概念上对应到 CSS 布局对应的形式。这是很常见的一种安排，但只是针对这段 HTML 众多可能的 CSS 样式中的一种。一定要阅读 3.13 节，了解如何进一步增强页面的语义和可访问性

3.13 使用 ARIA 改善可访问性

WAI-ARIA（Web Accessibility Initiative's Accessible Rich Internet Applications，无障碍网页倡议 – 无障碍的富互联网应用，也简称 ARIA）是一种技术规范，自称“有桥梁作用的技术”。之所以这样说，是因为在 HTML 提供相应的语义功能之前，它会使用属性来填补一些语义上的空白。

你可能已经注意到，本章的大多数例子都使用了一个或多个 ARIA role 属性，而且多次提示你跳到这一节了解更多关于该属性的说明。现在，就让我们开始吧！噢，等等，在讲解 role 属性之前，先简单看一下无障碍访问的知识。

无障碍访问的意义是让所有的访问者都能获取网站的内容。你的网站的一部分访问者可能需要借助辅助设备（如屏幕阅读器）访问你的页面内容（参见补充材料“试用屏

幕阅读器”)。

让网站具备无障碍访问功能是成为严肃负责的 Web 公民的重要方面。而且，这样做对你也是有好处的——为什么不让访问者能够获取网站的内容呢？

幸运的是，在大多数情况下，让页面具有无障碍访问功能并不难。只要对内容使用最恰当的 HTML 进行标记，就能改进网站的可访问性。如果你一直在这样做，非常好。在这一节，我会讲到如何在 HTML 中添加一些简单的属性进一步提升网站的可访问性。

地标角色

前面提到，ARIA 填补了 HTML 的语义空白。例如，如果你想让屏幕阅读器用户知道如何跳至页面级的页脚，应该使用什么 HTML 标记呢？标记为 footer 还不够，要知道页面可以包含多个 footer。

对此，HTML 没有解决方案，而 ARIA 的地标角色 (landmark role) 则可以满足需求。地标角色可以帮助用户识别页面区域，从而让屏幕阅读器用户可以直接跳到这些区域。通常，对这些区域指定 role 属性就可以了。

如表 3.13.1 所示，地标角色和 HTML5 元素之间有一些重合（要知道 HTML5 也使用本章介绍的一些新元素填补了某些语义空白），但目前屏幕阅读器对 ARIA 的支持更多。因

此你可以继续按既有的方式创建 HTML（包括使用新元素），同时添加一些 ARIA 角色提升页面的可访问性。

试用屏幕阅读器

以下屏幕阅读器是最为常见的一些。除了 VoiceOver 以外，这些屏幕阅读器都仅适用于 Windows。

- ❑ JAWS: 免费的演示版位于 www.freedomscientific.com。
- ❑ NVDA: 免费，位于 www.nvda-project.org。
- ❑ VoiceOver: 免费，是 OS X 和 iOS 4+ 的一部分。在 OS X 上，可以通过 Command-F5 开启或关闭 VoiceOver。要了解如何在 iOS 设备上使用 VoiceOver，请访问 <https://support.apple.com/kb/HT3598>。
- ❑ Window-Eyes: 免费的演示版位于 www.gwmicro.com。

强烈建议你从这些软件中挑选一个进行试用。这样，你会对屏幕阅读器的用户体验有更深入的理解。而且你还将直观地了解到语义化的 HTML 如何影响使用屏幕阅读器的体验。在屏幕阅读器中测试网页最好成为建站标准流程的一部分。

表 3.13.1 一些可用的地标角色

地标角色	如何使用及何时使用
role="banner" (横幅) 面向全站的内容，通常包含网站标志、网站赞助者标志、全站搜索工具等。横幅通常显示在页面的顶端，而且通常横跨整个页面的宽度	将其添加到页面级的 header 元素，每个页面只用一次
role="navigation" (导航) 文档内不同部分或相关文档的导航性元素(通常为链接)的集合	与 nav 元素是对应关系。应将其添加到每个 nav 元素，或其他包含导航性链接的容器。这个角色可在每个页面上使用多次，但是同 nav 一样，不要过度使用该属性

(续)

地标角色	如何使用及何时使用
<code>role="main"</code> (主体) 文档的主要内容	与 <code>main</code> 元素是对应关系。最好将其添加到 <code>main</code> 元素，也可以添加到其他表示主体内容的元素 (可能是 <code>div</code>)。在每个页面仅使用一次
<code>role="complementary"</code> (补充性内容) 文档中作为主体内容补充的支撑部分。它对区分主体内容是有意义的	与 <code>aside</code> 元素是对应关系。应将其添加到 <code>aside</code> 或 <code>div</code> 元素 (前提 是该 <code>div</code> 仅包含补充性内容)。可以在一个页面里包含多个 <code>complementary</code> 角色，但不要过度使用
<code>role="contentinfo"</code> (内容信息) 包含关于文档的信息的大块可感知区域这类信息的例子包括版权声明和指向隐私权声明的链接等	将其添加至整个页面的页脚 (通常为 <code>footer</code> 元素)。每个页面仅使用一次

使用地标角色很简单。图 3.13.1 与 3.12 节中的例子几乎是一样的，只不过添加了一个 `nav` 元素，并对地标角色进行了突出显示。

表 3.13.1 引用了一些 ARIA 规范 (www.w3.org/TR/wai-aria/roles#landmark_roles) 中

对地标角色的定义及推荐用法 (另外还有三种地标角色，参见第一条提示)。对它们的描述看起来会很熟悉，因为它们与对应的 HTML 元素的用法是相似的 (参见图 3.13.1 和图 3.13.2)。

```
...
<body>
<!-- 开始页面容器 -->
<div class="container">
  <header role="banner">
    ...
    <nav role="navigation">
      ... [包含多个链接的无序列表] ...
    </nav>
  </header>

  <!-- 应用CSS后的第一栏 -->
  <main role="main">
    <article>
      ...
    </article>

    <article>
      ...
    </article>

    ... [其他区块] ...
  </main>
```

```
<!-- 结束第一栏 -->

<!-- 应用CSS后的第二栏 -->
<div class="sidebar">
  <aside role="complementary">
    ...
  </aside>

  <aside role="complementary">
    ...
  </aside>

  ... [其他区块] ...
</div>
<!-- 结束第二栏 -->

<footer role="contentinfo">
  ...
</footer>
</div>
<!-- 结束页面容器 -->
</body>
</html>
```

图 3.13.1 这个页面使用了表 3.13.1 中列出的五种地标角色 (还有一种表中未列出的)

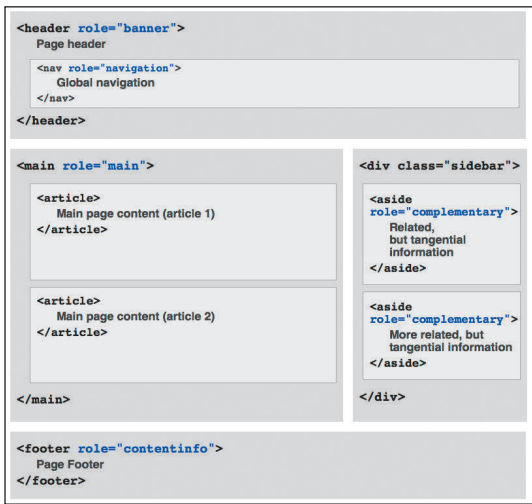


图 3.13.2 这张图与 3.12 节中的图很相似。这张图仅用于将常见的地标角色可视化，并不代表它们对布局有影响。例如，在 main 元素中，可以包含一个带有 role="navigation" 的 nav 作为局部导航。同时，带有 class="sidebar" 的 div 也可以添加 role="complementary"，从而代替 aside 元素，前提是整个附注栏的内容确实为补充性内容

需要澄清的是，即便不使用 ARIA 地标角色，页面看起来也没有任何差别，但是使用它们可以提升使用辅助设备的用户的体验。出于这个理由，推荐使用它们。在本书的例子中，我都在恰当的地方使用了它们。本书配套网站上的代码示例也是这样做的。

提示 表 3.13.1 没有包含另外三种地标角色。对表单元来说，form 角色是多余的；search 用于标记搜索表单；application 则属于高级用法。16.8 节有一个使用 role="search" 的例子。

提示 不要在页面上过多地使用地标角色。过多的地标角色会让屏幕阅读器用户感到累赘，从而降低地标的的作用，影响整体体验。

提示 可用性专家 Steve Faulkner 对地标角色有详细的讲解，见 blog.paciellogroup.com/2013/02/using-wai-aria-landmarks-2013/。（表 3.13.1 与他的表格之间的细小差别可以忽略。）他还引用了另一位专家 Léonie Watson 录制的演示屏幕阅读器用户访问页面行为的视频，推荐观看！

提示 WebAIM 会定期对屏幕阅读器用户做一些调查，以了解他们访问网站的偏好和遇到的问题。调查结果很值得一读，见 webaim.org/projects/screenreadersurvey4/。

提示 地标角色只是 ARIA 规范 (www.w3.org/TR/wai-aria/) 众多特性中的一个。如果对相关的实施指南感兴趣，可以访问 www.w3.org/WAI/PF/aria-practices/。

提示 可以在 CSS 选择器中使用 ARIA 角色属性，从而对使用它们的元素添加样式。具体细节参见第 11 章。

3.14 为元素指定类别或 ID 名称

尽管并非必需，但是可以给 HTML 元素分配唯一的标识符 (ID)，或指定其属于某个 (或某几个) 类别，也可以同时指定标识符和类别，如图 3.14.1 所示。

接着，就可以对具有给定 id 或 class 名称的元素添加样式了 (但一般不推荐出于添加样式的目的使用 id)；或者创建指向具有特定 id 的元素的链接 (图 3.14.1)；还可以使用 JavaScript 获取 id 和 class 属性，从而对元素添加特定的行为。

```

...
<body>
<div class="container">
  <header role="banner">
    <nav role="navigation">
      <ul>
        <li><a href="#gaudi">Barcelona's
          → Architect</a></li>
        <li><a href="#sagrada-familia"
          → lang="es">La Sagrada Família
          → </a></li>
        <li><a href="#park-guell">Park
          → Guell</a></li>
      </ul>
    </nav>
  </header>

  <main role="main">
    <article class="architect gaudi">
      <h1 id="gaudi">Barcelona's
        → Architect</h1>

      <p>Antoni Gaudí's incredible
        → buildings...</p>
      ...

      <h2 id="sagrada-familia" lang=
        → "es">La Sagrada Família</h2>
      ...

      <h2 id="park-guell">Park Guell
        → </h2>
      ...
    </article>
  </main>
</div>
</body>
</html>

```

图 3.14.1 nav 中的链接指向 h1 和 h2 中的 id。为一个或多个元素添加 class 属性，就可以对所有这类元素统一进行格式化。例如，可以将 architect 类应用到关于其他建筑师的内容，从而为它们应用一致的格式；gaudi 类可以为与他相关的内容提供另一种样式

1. 为元素添加唯一的 ID

在元素的开始标签中输入 `id="name"`，其中 *name* 是唯一标识该元素的名称（图 3.14.1）。

name 几乎可以是任何字符，只要不以数字开头且不包含空格。

2. 为元素指定类别的方法

在元素的开始标签中输入 `class="name"`，其中 *name* 是类别的名称（图 3.14.1）。如果要指定多个类别，用空格将不同的类别名称分开即可，如 `class="name anothername"`。（可以指定两个以上的类别名称。）

提示 关于通过 id 或 class 为元素添加样式，参见 9.3 节。不过，需要在这里说明的是，推荐使用类为元素添加样式。

提示 关于创建指向标有 id 的元素的链接，参见 6.2 节。如何对具有 id 或 class 的元素使用 JavaScript 已经超出了本书的范围，不过我们在第 19 章会看几个示例。

提示 HTML 文档中的每个 id 都必须是唯一的。换句话说，一个页面里不能出现两个具有相同 id 的元素，并且每个元素都只能有一个 id。相同的 id 可以出现在不同的页面里，同一 id 也不一定每次都赋给同一元素，尽管这是惯常的做法。

提示 相反，一个 class 名称可以分配给页面中任意数量的元素，并且一个元素可以有一个以上的 class。

提示 class 和 id 属性可以应用于任何 HTML 元素。元素可以同时拥有 id 和任意数量的 class。

提示 在 class 和 id 名称中，通常使用短横线分隔多个单词，例如 `class="footer-page"`。

提示 不管你打算如何使用 id 和 class，都应该为它们选择有意义的名称。例如，如果你使用 class 是出于格式化目的，应避免使用描述表现样式的名称，如 class="red"，因为你可能在下周决定将配色方案改为蓝色。尽管在 CSS 中对分配给某一类元素的颜色进行修改是相当容易的，但这样做会导致你的 HTML 拥有一个名为红色却实际呈现为另一种颜色的 class。同时，改变 HTML 中所有的 class 通常是一项繁琐的工作。这点在你开始学习 CSS 之后尤为明显。

提示 可以使用 class 属性实现微格式（microformat）。更多信息参见 <http://microformats.org>。

3.15 为元素添加 title 属性

可以使用 title 属性（不要与 title 元素混淆）为网站上任何部分加上提示标签（参见图 3.15.1 至图 3.15.3）。不过，它们并不只是提示标签，加上它们之后屏幕阅读器可以为用户朗读 title 文本，因此使用 title 可以提升无障碍访问功能。

```
...
<ul title="Table of Contents">
  <li><a href="#gaudi" title="Learn about
    → Antoni Gaudí">Barcelona's Architect
    → </a></li>
  <li><a href="#sagrada-familia" lang="es">
    → La Sagrada Família</a></li>
  <li><a href="#park-guell">Park Guell</a>
    → </li>
</ul>
...
</body>
</html>
```

图 3.15.1 可以为任何元素添加 title，不过用的最多的是链接



图 3.15.2 当访问者将鼠标指向加了说明标签的元素时，就会显示 title。如果指向 Barcelona's Architect 链接……



图 3.15.3 ……会看到 Learn about Antoni Gaudí，因为该链接有 title 属性

在网页中为元素添加标签

在要添加 title 的 HTML 元素中，输入 title="label"，其中 label 是访问者将鼠标移到这个元素上时希望出现在提示框里的文本，或者希望由屏幕阅读器朗读的文本。

提示 旧版本的 Internet Explorer（IE7 及之前的版本）还会将 img 元素（参见第 5 章）的 alt 属性作为提示框的文本。不过，如果 img 元素同时包括 title 和 alt 属性，则提示框会采用 title 属性的内容，而不是 alt 属性的内容。

3.16 添加注释

可以在 HTML 文档中添加注释, 标明区块开始和结束的位置, 提醒自己 (或未来的代码编辑者) 某段代码的意图, 或者阻止内容显示等, 如图 3.16.1 所示。这些注释只会在用文本编辑器或浏览器的“查看源代码”选项打开文档时显示出来。访问者在浏览器中是看不到它们的, 如图 3.16.2 所示。

```
...
<!-- ==== 开始主体内容 ==== -->
<main role="main">
  <article class="architect">
    <h1 id="gaudi">Barcelona's Architect
    → </h1>

    <!-- 这一段不会显示出来, 因为它被注释
    → 掉了
    <p>Antoni Gaudí's incredible
    → buildings bring millions of
    → tourists to Barcelona each year.
    → </p>
    -->

    <p>Gaudí's non-conformity, already
    → visible in his teenage years...</p>
    ...
  </article>
</main>
<!-- 结束主体内容 -->

<!-- ==== 开始附注栏 ==== -->
... [附注栏内容] ...
<!-- 结束附注栏 -->
...
```

图 3.16.1 这段示例代码包括五个注释。其中有四个一起标记了两个区块的开始和结束位置。另一个“注释掉”了第一段, 这样它就不会显示在页面中 (如果希望永久性地移除该段, 最好将它从 HTML 中删除)

在 HTML 页面中添加注释的步骤

(1) 在 HTML 文档中希望插入注释的位

置, 输入 <!--。

(2) 输入注释。

(3) 输入 --> 结束注释文本。



图 3.16.2 注释是不可见的。类似地, 如果将一些内容“注释掉”, 它们也不会显示出来。这段代码中的第一个段落并不会显示在浏览器中

提示 在主要区块的开头和结尾处添加注释是一种常见的做法, 这样可以让你或一起合作的开发人员将来修改代码变得更加容易。笔者喜欢使用一种比结束注释更为醒目的格式编写开始注释, 从而更容易区分这两个位置。

提示 在发布网站之前, 应该用浏览器查看一下加了注释的页面。这样能帮你避免由于弄错注释格式导致私人化的注释内容直接暴露给公众访问者的情况。

提示 对特别私人化的注释要格外小心。尽管通常通过浏览器访问页面不会看见注释内容, 但通过浏览器的“查看源代码”功能就能看见它们, 如果用户将网页保存为 HTML 源代码, 他们也能看见这些注释。

提示 注释不能嵌套在其他注释里。

本章内容

- 添加段落
- 指定细则
- 标记重要或强调的文本
- 创建图
- 指明引用或参考
- 引述文本
- 指定时间
- 解释缩写词
- 定义术语
- 创建上标和下标
- 添加作者联系信息
- 标注编辑和不准确的文本
- 标记代码
- 使用预格式化的文本
- 突出显示文本
- 创建换行
- 创建 `span`
- 其他元素

除非网站添加了太多视频和图片，否则网页的大部分内容还是文本。本章将说明针对不同的文本类型，尤其是（但不仅仅是）那些句子或短语里的文本，应该选择哪些合适的 HTML 语义化元素。

例如，`em` 元素用于标识强调的文本，`cite` 元素用于标识对艺术作品、电影、图书等内容的引用。

浏览器通常会为这些文本元素添加不同的样式，以区别于普通文本。例如，`em` 和 `cite` 元素中的文本都是斜体的。又如，`code` 元素专门用来格式化脚本或程序中的代码，该元素中的文本默认使用等宽字体。

内容显示的样子与为其使用的标记没有关系。因此，不应该为了让文字变为斜体就使用 `em` 或 `cite`，添加样式是 CSS 的事情。

相反，应该选择能描述内容的 HTML 元素。如果浏览器默认添加的样式与你想用 CSS 设置的样式相同，那只不过是一种额外的奖励。如果不相同，直接自己编写 CSS 覆盖默认样式就可以了。

4.1 添加段落

HTML 会忽略你在文本编辑器中输入的回车符和其他额外的空格。要在网页中开始一个新的段落，应该使用 `p` 元素，如图 4.1.1 和图 4.1.2 所示。

```

...
<body>

<h1>Antoni Gaudí</h1>
<p>Many tourists are drawn to Barcelona
→ to see Antoni Gaudí's incredible
→ architecture.</p>

<p>Barcelona celebrated the 150th
→ anniversary of Gaudí's birth in
→ 2002.</p>

<h2 lang="es">La Casa Milà</h2>
<p>Gaudí's work was essentially useful.
→ <span lang="es">La Casa Milà</span> is
→ an apartment building and real people
→ live there.</p>

<h2 lang="es">La Sagrada Família</h2>
<p>The complicatedly named and curiously
→ unfinished Expiatory Temple of the
→ Sacred Family is the most visited
→ building in Barcelona.</p>

</body>
</html>

```

图 4.1.1 毫无疑问，p 是最常使用的 HTML 元素之一（注意：在实践中，我通常会使用 article 包住这些内容。这里省略了该标记，是为了让例子显得更为通用，同时避免产生 p 元素必须嵌套在 article 中的印象）



图 4.1.2 这是段落通常默认呈现方式。默认情况下，浏览器会在标题和段落之间，以及不同的段落之间添加垂直间距。使用 CSS 可以控制所有内容元素的格式

创建新段落的步骤

- (1) 输入 <p>。
- (2) 输入新段落的内容。
- (3) 输入 </p> 结束这个段落。

提示 可以为段落（以及页面上的其他文本）添加样式，包括字体、字号、颜色等。详细内容参见第 10 章。

提示 要控制段内行间距，参见 10.7 节。要控制段前段后的间距，参见 11.8 节和 11.10 节。

提示 可以通过 CSS 改变段落文本的对齐方式，包括左对齐、右对齐和居中对齐（参见 10.13 节）。

4.2 指定细则

根据 HTML5，small 表示细则一类的旁注（side comment），“通常包括免责声明、注意事项、法律限制、版权信息等。有时我们还可以用它来表示署名，或者满足许可要求。”

small 通常是行内文本中的一小块，而不是包含多个段落或其他元素的大块文本，参见图 4.2.1 和图 4.2.2。

指定细则的步骤

- (1) 输入 <small>。
- (2) 输入表示免责声明、注解、署名等类型的文本。
- (3) 输入 </small>。

提示 一些浏览器会将 small 包含的文本显示为小字号（图 4.2.2）。不过，一定要在符合内容语义的情况下使用该元素，而不是为了减小字号而使用。总是可以用 CSS 控制字号（如果你想，甚至可以让 small 包含的文本的字号变大），更多信息参见 10.6 节。

```

...
<body>

<p>Order now to receive free shipping.
<small>(Some restrictions may apply.)
→ </small></p>

...

<footer role="contentinfo">
  <p><small>&copy; 2013 The Super
    → Store. All Rights Reserved.
    → </small></p>
</footer>

</body>
</html>

```

图 4.2.1 在下面的两个例子中，我们用 small 元素来表示简短的法律声明。在第二个例子中，small 表示的是包含在页面级 footer 里的版权声明，这是一种常见的用法

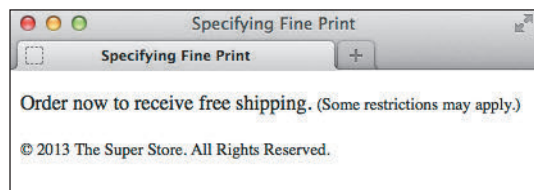


图 4.2.2 在一些浏览器中，small 元素中文本的字号会比普通文本的字号小，不过，视觉上的大小与是否该用它标记内容毫无关系

提示 用 small 标记页面的版权信息是一种常见的做法（图 4.2.1、图 4.2.2）。不过，small 只适用于短语，因此不要用它标记长的法律声明，如“使用条款”和“隐私政策”页面。根据需要，应该用段落或其他语义标签标记这些内容。

4.3 标记重要和强调的文本

strong 元素表示内容的重要性，而 em 则

表示内容的着重点。根据内容需要，这两个元素既可以单独使用，也可以一起使用，参见图 4.3.1 和图 4.3.2。

```

...
<body>

<p><strong>Warning: Do not approach the
→ zombies <em>under any circumstances</em>
→ </strong>. They may <em>look</em>
→ friendly, but that's just because they want
→ to eat your arm.</p>

</body>
</html>

```

图 4.3.1 第一个句子既有 strong 又有 em，而第二个句子只有 em

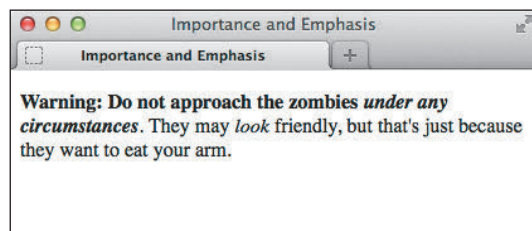


图 4.3.2 浏览器通常将 strong 文本以粗体显示，而将 em 文本以斜体显示。如果 em 是 strong 的子元素（如图 4.3.1 中的第一个句子），将同时以斜体和粗体显示文本

标记重要文本

- (1) 输入 。
- (2) 输入想标记为重要内容的文本。
- (3) 输入 。

强调文本

- (1) 输入 。
- (2) 输入想强调的文本。
- (3) 输入 。

提示 不要使用 `b` 元素代替 `strong`，也不要使用 `i` 元素代替 `em`。尽管它们在浏览器中显示的样式是一样的，但它们的含义却很不一样（参见补充材料“HTML5 中重新定义的 `b` 和 `i` 元素”）。

提示 就像在说话时强调某些词语一样，`em` 在句子中的位置会影响句子的含义。例如，`<p>em>Run</em> over here.</p>` 和 `<p>Run over <em>here</em>.</p>` 表达的意思是不同的。

提示 可以在标记为 `strong` 的短语中再嵌套 `strong` 文本。如果这样做，作为另一个 `strong` 的子元素的 `strong` 文本的重要程度会递增。这种规则对嵌套在另一个 `em` 里的 `em` 文本也适用。例如，在语句 `<p><strong>Remember that entries are <strong>due by March 12th</strong>.</strong></p>` 中，“due by March 12th”要比其他 `strong` 文本更为重要。

提示 可以用 CSS 将任何文本变为粗体或斜体，也可以覆盖 `strong` 和 `em` 等元素的浏览器默认显示样式（图 4.3.2）。更多细节参见 10.4 节和 10.5 节。

提示 如果你有过编写旧版本 HTML 的经验，你或许知道，那时 `strong` 所表示文本的强调程度比 `em` 表示的文本要高。不过，在 HTML5 中，`em` 是表示强调的唯一元素，而 `strong` 表示的则是重要程度。

HTML5 中重新定义的 `b` 和 `i` 元素

HTML5 强调元素的语义，而非表现。`b` 和 `i` 元素是早期 HTML 遗留下来的产物，它们分别用于将文本变为粗体和斜体（那时 CSS 还未出现）。HTML 4 和 XHTML 1 当然抛弃了它们，因为它们本质上是用于表现的。当时的规范建议编码人员用 `strong` 替代 `b`，用 `em` 替代 `i`。不过，事实证明，`em` 和 `strong` 有时在语义上并不合适。为此，HTML5 重新定义了 `b` 和 `i`。

传统出版业里的某些排版规则在现有的 HTML 语义中还找不到对应物，其中就包括用斜体表示植物学名（如“*Ulmus americana* is the Massachusetts state tree.”）、具体的交通工具名称（如“We rode the *Orient Express*.”）及外来语（如“The couple exhibited a *joie de vivre* that was infectious.”）。这些词语不是为了强调而加上斜体的，只是样式上的惯例。

为了应对这些情况，HTML5 没有创建一些新的语义化元素（进一步把事情搞复杂），而是采取了一种很实际的做法，直接利用现有元素：`em` 用于所有层次的强调，`strong` 用于表示重要性，而其他情况则使用 `b` 和 `i`。

这意味着, 尽管 **b** 和 **i** 并不包含任何明显的语义, 但读者仍能发现它们与周边文字的差别。而且你还可以通过 CSS 改变它们粗体或斜体的样式。HTML5 强调, **b** 和 **i** 应该是其他元素 (如 **strong**、**em**、**cite** 等) 都不适用时的最后选择。

◎ **b** 元素简介

HTML5 将 **b** 重新定义为:

b 元素表示出于实用目的提醒读者注意的一块文字, 不传达任何额外的重要性, 也不表示其他的语态和语气, 用于如文档摘要里的关键词、评论中的产品名、基于文本的交互式软件中指示操作的文字、文章导语等。

例如:

```
<p>The <b>XR-5</b>, also dubbed the <b>Extreme Robot 5</b>, is the best robot we've ever  
→ tested.</p>
```

b 元素默认显示为粗体。

◎ **i** 元素简介

HTML5 将 **i** 重新定义为:

i 元素表示一块不同于其他文字的文字, 具有不同的语态或语气, 或其他不同于常规之处, 用于如分类名称、技术术语、外语里的惯用语、翻译的散文、西方文字中的船舶名称等。

例如:

```
<p>The <i lang="la">Ulmus americana</i> is the Massachusetts state tree.</p>  
<p>We rode the <i>Orient Express</i>.</p>  
<p>The couple exhibited a <i lang="fr">joie de vivre</i> that was infectious.</p>
```

i 元素默认显示为斜体。

4.4 创建图

你一定在报纸、杂志、报告等其他媒介上看到过图。通常, 图是由页面上的文本引述出来的。本书的大多数页面都有这样的例子。

在 HTML5 出现之前, 没有专门实现这个目的的元素, 因此一些开发人员想出了他们自己的解决办法 (通常会使用不那么理想的、没有语义的 **div** 元素)。通过引入 **figure** 和 **figcaption**, HTML5 改变了这种情况 (参见图 4.4.1 和图 4.4.2)。图可以是图表、照片、图形、插图、代码片段, 以及其他类似的独立内容。

可以由页面上的其他内容引出 **figure** (如图 4.4.1 和图 4.4.2 所示)。**figcaption** 是 **figure** 的标题, 可选, 出现在 **figure** 内容的开头或结尾处。

创建图及其标题的步骤

- (1) 输入 **<figure>**。
- (2) 作为可选步骤, 输入 **<figcaption>** 开始图的标题。
- (3) 输入标题文字。
- (4) 如果在第 (2)、(3) 步创建了标题, 就输入 **</figcaption>**。
- (5) 添加图像、视频、数据表格等内容的

代码创建图。

(6) 如果没有在 figure 内容之前包含 figcaption，则可以在内容之后重复第 (2) ~ (4) 步。

(7) 输入 `</figure>`。

```
...
<body>
...
<article>
  <h1>2013 Revenue by Industry</h1>

  <p>... [报告内容] ...</p>

  <figure>
    <figcaption><b>Figure 3:</b>
    → Breakdown of Revenue by
    → Industry</figcaption>

    
  </figure>

  <p>As Figure 3 illustrates, ... </p>

  <p>... [其他报告内容] ...</p>
</article>
...
</body>
</html>
```

图 4.4.1 这个 figure 只有一个图表，不过放置多个图像或其他类型的内容（如数据表格、视频等）也是允许的。figcaption 元素并不是必需的，但如果包含它，它就必须是 figure 元素内嵌的第一个或最后一个元素。除了在现代浏览器中从新的一行开始显示，figure 没有默认样式，如图 4.4.2 所示（注意：figure 元素并不一定要包含在 article 里，但在大多数情况下这样做比较合适）

提示 通常，figure 是引用它的内容的一部分（图 4.4.1），但它也可以位于页面的其他部分，或位于其他页面（如附录）。

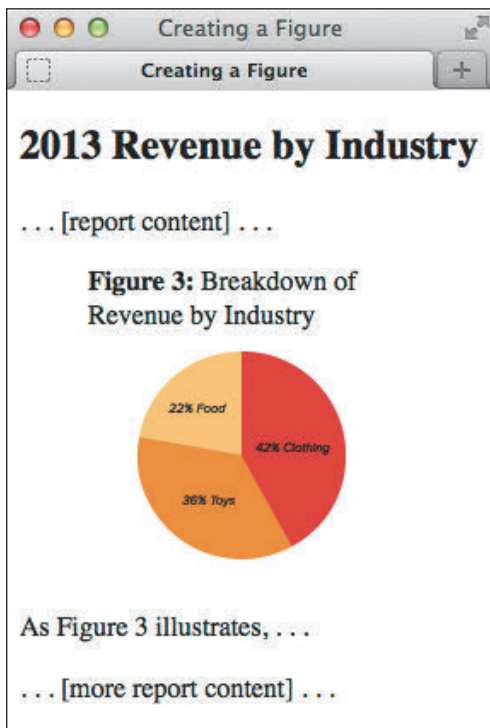


图 4.4.2 这个包含图表及其标题的 figure 出现在 article 文本中间。图以缩进的形式显示，这是浏览器的默认样式（参见最后一条提示）

提示 figure 元素可以包含多个内容块。例如，图 4.4.1 中可以包含两个图表：一个表示收入，一个表示利润。不过要记住，不管 figure 里有多少内容，只允许有一个 figcaption。

提示 不要简单地将 figure 作为在文本中嵌入独立内容实例的方法。这种情况下，通常更适合用 aside 元素（参见 3.10 节）。

提示 要了解如何结合使用 blockquote 和 figure 元素，参见 4.6 节。

提示 可选的 `figcaption` 必须与其他内容一起包含在 `figure` 里面，不能单独出现在其他位置。

提示 `figcaption` 中的文本不必以 Figure 3、Exhibit B 等字样开头，对内容的一句简短描述即可，就像照片的描述文本。

提示 现代浏览器在默认情况下会为 `figure` 添加 40px 宽的左右外边距（图 4.4.3）。可以使用 CSS 的 `margin-left` 和 `margin-right` 属性修改这一样式。例如，使用 `margin-left: 0`；可以让图直接抵到页面左边缘。还可以使用 `figure { float: left; }` 让包含 `figure` 的文本环绕在它周围（这样，文本就会围在图的右侧）。可能还需要为 `figure` 设置 `width`，使之不至于占据太大的水平空间。本书将从第 7 章开始阐述 CSS，`float` 和 `width` 属性会在第 11 章中讲到。

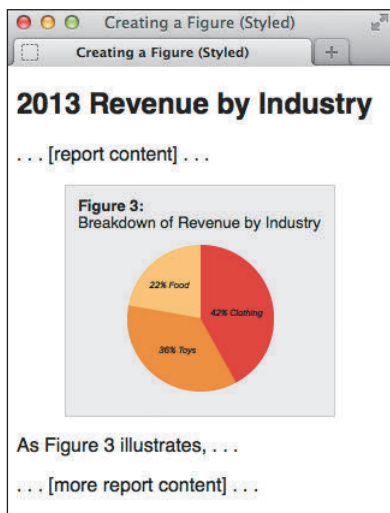


图 4.4.3 使用一点 CSS 就可以让 `figure` 与它旁边的文本区分开来。这个简单的例子见 www.htmlcssvqs.com/8ed/figure-styled/

4.5 指明引用或参考

使用 `cite` 元素可以指明对某内容源的引用或参考。例如，戏剧、脚本或图书的标题，歌曲、电影、照片或雕塑的名称，演唱会或音乐会，规范、报纸或法律文件等（参见图 4.5.1 和图 4.5.2）。

```
...
<body>

<p>He listened to <cite>Abbey Road</cite>
→ while watching <cite>A Hard Day's Night
→ </cite> and reading <cite>The Beatles
→ Anthology</cite>.

<p>When he went to The Louvre, he learned
→ that <cite>Mona Lisa</cite> is also
→ known as <cite lang="it">La Gioconda
→ </cite>.</p>

</body>
</html>
```

图 4.5.1 在这个例子中，`cite` 元素标记的是音乐专辑、电影、图书和艺术作品的标题（注意：最后一个例子中的 `lang="it"` 表明 `cite` 文本的语言是意大利语）

图 4.5.2 `cite` 元素默认以斜体显示

引用参考的步骤

- (1) 输入 `<cite>`。
- (2) 输入参考的名称。
- (3) 输入 `</cite>`。

提示 对于要从引用来源中引述内容的情况，使用 `blockquote` 或 `q` 元素标记引述的文本（参见 4.6 节）。要弄清楚的是，`cite` 只用于参考源本身，而不是从中引述的内容。

HTML5 与使用 `cite` 元素表示名称

HTML5 声明，不应使用 `cite` 作为对人名的引用（这是开发社区众多意见分歧中的一个），但 HTML 以前的版本允许这样做，而且很多设计和开发人员仍在这样做。

HTML4 的规范有以下例子（已将元素名称由大写字母改为小写）：

```
As <cite>Harry S. Truman</cite> said,
<q lang="en-us">The buck stops here.</q>
```

除了这些例子，有的网站经常用 `cite` 标记在博客和文章中发表评论的访问者的名字（WordPress 的默认主题就是这样做的）。

很多开发人员表示他们将继续对与页面中的引文有关的名称使用 `cite`，因为 HTML5 没有提供他们认为可接受的其他元素（即 `span` 和 `b` 元素）。Jeremy Keith 提供了一个很好的例子，见 <http://24ways.org/2009/incite-a-riot/>。

4.6 引述文本

有两个特殊的元素用以标记引述的文本。`blockquote` 元素表示单独存在的引述（通常更长，但也可能不是），参见图 4.6.1 和图 4.6.2，它默认显示在新的一行，如图 4.6.3 所示。而 `q` 元素则用于短的引述，如句子里面的引述（图 4.6.4）。

```
...
<body>

<p>He especially enjoyed this selection from
→ <cite>The Adventures of Huckleberry Finn
→ </cite> by Mark Twain:</p>

<blockquote cite="http://www.
→ marktwainbooks.edu/the-adventures-of-
→ huckleberry-finn/">
  <p>We said there warn't no home like a
  → raft, after all. Other places do seem
  → so cramped up and smothery, but a
  → raft don't. You feel mighty free and
  → easy and comfortable on a raft.</p>
</blockquote>

<p>It reminded him of his own youth exploring
→ the county by river in the summertime.</p>

</body>
</html>
```

图 4.6.1 根据需要，`blockquote` 可长可短。可以包含 `cite` 属性（不要与第一段中出现的 `cite` 元素混淆）提供引述文本的位置

```
...

<figure>
  <blockquote>
    I want all my senses engaged. Let
    → me absorb the world's variety and
    → uniqueness.
  </blockquote>
  <figcaption>— Maya Angelou</figcaption>
</figure>

...
```

图 4.6.2 如果要添加署名，署名应该放在 `blockquote` 外面。可以把署名放在 `p` 里面，不过使用 `figure` 和 `figcaption` 可以更好地将引述文本与其来源关联起来，如下所示（参见 4.5 节）

浏览器应对 `q` 元素中的文本自动加上特定语言的引号，但不同浏览器的效果并不相同（参见图 4.6.5）。一定要阅读本节提示，了解 `q` 元素的替代方法。

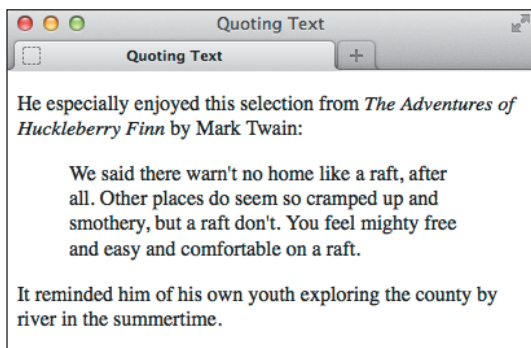


图 4.6.3 浏览器默认对 blockquote 文本进行缩进，cite 属性的值则不会显示出来（对此，参见第二条提示中的推荐做法）。不过，所有的浏览器都支持 cite 元素，通常对其中的文本以斜体显示，如图所示。所有这些默认样式都可以被 CSS 覆盖

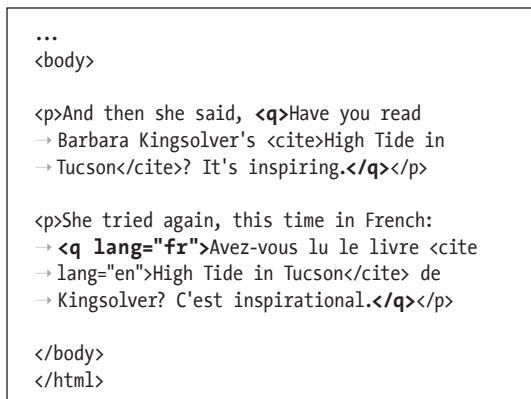


图 4.6.4 这里是两个 q 的例子。如果引述文本的语言与页面默认语言（通过 html 元素的 lang 属性指定）不同，就在 q 元素中加上 lang 属性

1. 引述块级文本的步骤

- (1) 输入 <blockquote> 开始一个块级引述。
- (2) 如果需要，输入 cite="url"，其中 url 为引述来源的地址。
- (3) 输入 > 以结束开始标签。
- (4) 输入希望引述的文本，并用段落等适当的元素包围。
- (5) 输入 </blockquote>。

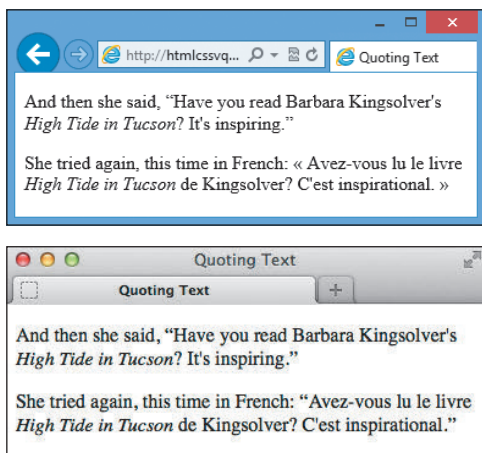


图 4.6.5 浏览器应该自动在 q 元素周围加上特定语言的引号。在这个例子中，这意味着英文和法文分别使用不同的引号。IE（上图）和 Chrome 的处理是正确的。Firefox（下图）对英文的处理是正确的，而法文则不正确。Opera 和 Safari 对两种语言的处理都不正确，它们都显示直引号（包括法文）。这样的不一致影响了 q 元素的有效性

2. 引述行内文本的步骤

- (1) 输入 <q> 开始引述字词或短语。
- (2) 如果需要，输入 cite="url"，其中 url 为引述来源的地址。
- (3) 如果引述内容的语言与页面默认语言（通过 html 元素的 lang 属性指定）不同，输入 lang="xx"，其中 xx 是引述内容语言的两个字母的代码。这个代码用于判断需要使用的引号的类型（英语为 " "，而很多欧洲语言则为 « »），但浏览器对引号的呈现方式可能不同。

- (4) 输入 > 以结束开始标签。
- (5) 输入要引述的文本。
- (6) 输入 </q>。

提示 如果 blockquote 中仅包含一个单独的段落或短语，可以不必将其包在 p 中再放入 blockquote。

提示 可以对 `blockquote` 和 `q` 使用可选的 `cite` 属性，提供引述内容来源的 URL。尽管浏览器通常不会将 `cite` 的 URL 呈现给用户（参见图 4.6.3），但理论上讲，该属性对搜索引擎或其他收集引述文本及其引用的自动化工具来说还是有用的。如果要让访问者看到这个 URL，可以在内容中使用链接（`a` 元素）重复这个 URL。也可以使用 JavaScript 将 `cite` 的值暴露出来（可以从网上搜到示例代码），但这样做的效果稍差一些。

提示 `q` 元素引用的内容不能跨越不同的段落，在这种情况下应使用 `blockquote`。

提示 不要仅仅因为需要在字词两端添加引号就使用 `q`。例如，`<p>Every time I hear the word <q>soy</q>, I jump for joy.</p>` 是不正确的，因为 `soy` 并不是对某参考源的引用。

提示 `blockquote` 和 `q` 元素可以嵌套。例如，`<p>The short story began, <q>When she was a child, she would say, <q>Howdy, stranger!</q> to everyone she passed.</p>`。嵌套的 `q` 元素应该自动加上正确的引号（例如，在英语中外面的是双引号，里面的是单引号）。由于内外引号在不同语言中的处理方式不一样，因此要根据需要在 `q` 元素中加上 `lang` 属性（参见图 4.6.4）。不幸的是，浏览器对嵌套 `q` 元素的支持程度并不相同，其实浏览器对非嵌套 `q` 元素的支持也不同（参见图 4.6.5）。

提示 由于 `q` 元素的跨浏览器问题（参见图 4.6.5），很多（可能是大多数）开发人员避免使用 `q` 元素，而是选择直接输入正确的引号或使用字符实体。

4.7 指定时间

我们可以使用 `time` 元素标记时间、日期或时间段，这是 HTML5 新增的元素。呈现这些信息的方式有多种，参见图 4.7.1 至图 4.7.3。

```
...
<body>

<p>The train arrives at <time>08:45</time>
→ and <time>16:20</time> on <time>
→ 2017-03-19</time>.</p>

<p>They made their dinner reservation for
→ <time datetime="2013-11-20T18:30:00">
→ tonight at 6:30</time>.</p>

<p>We began our descent from the peak of
→ Everest on <time datetime="1952-06-12T
→ 11:05:00">June 12, 1952 at 11:05 a.m.
→ </time></p>

<p>The film festival is <time datetime=
→ "2014-07-13">July 13</time>-<time
→ datetime="2014-07-16">16</time>.</p>

<!-- 不包含年份的例子 -->
<p>Her birthday is <time datetime="03-29">
→ March 29th</time>.</p>

<!-- 时间段的例子 -->
<p>The meeting lasted <time>2h 41m 3s
→ </time> instead of the scheduled <time
→ datetime="2h 30m">two hours and thirty
→ minutes</time>.</p>

</body>
</html>
```

图 4.7.1 如第一个例子所示，`time` 元素最简单的用法是不包含 `datetime` 属性。在忽略 `datetime` 属性的情况下，它们的确提供了具备有效的机器可读格式的时间和日期。在其余的例子中，由于提供了 `datetime` 属性，`time` 标签中的文本并未严格使用有效的格式

`time` 中包含的文本内容（即 `<time>text</time>`）会出现在屏幕上，对用户可见（参见图 4.7.2 和图 4.7.4），而可选的 `datetime`

属性则是为机器准备的。该属性需要遵循特定的格式，本节后面的补充材料“理解有效的时间格式”说明了相关的基本知识，第一条提示则说明了另一种格式要求的特定情形。

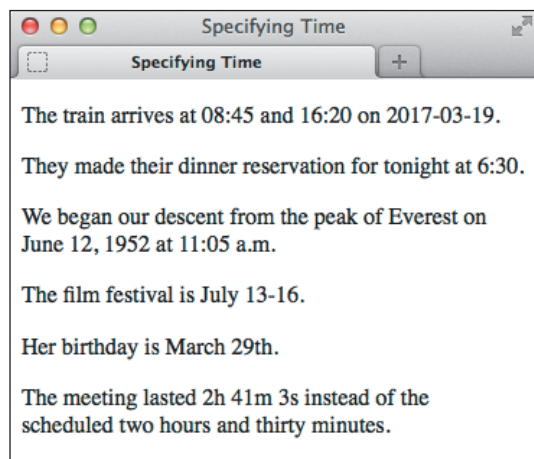


图 4.7.2 浏览器只显示 time 元素的文本内容，而不会显示 datetime 的值

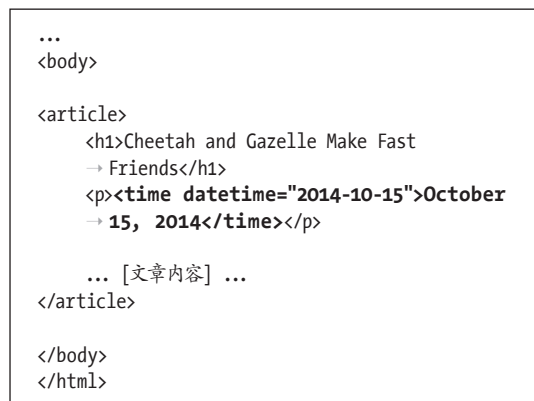


图 4.7.3 这里展示了如何在博客或新闻文章中包含日期。datetime 属性中的值表示的是文本内容的机器可读格式

指定准确时间、日期或时间段的步骤

- (1) 输入 <time 开始 time 元素。
- (2) 如果需要，输入 datetime="time"，其中 time 表示的是第 (4) 步中文本的机器可读

格式(参见补充材料“理解有效的时间格式”)。

(3) 输入 > 结束开始标签。

(4) 输入要显示在浏览器中的表示时间、日期或时间段的文本(如果第 (2) 步中没有包含 datetime 属性，参见第一条提示)。

(5) 输入 </time>。



图 4.7.4 恰如预期，日期显示在标题的下方

提示 如果忽略 datetime 属性，文本内容就必须是合法的日期或时间格式。也就是说，图 4.7.1 中的第一个例子不能写成 <p>The train arrives at <time>8:45 a.m.</time> and <time>4:20 p.m.</time> on <time>April 20th, 2015</time>.</p>，因为这三处 time 文本的格式都不正确。不过，如果包含了 datetime，文本内容就可以按你希望的任何形式表示日期、时间或时间段了，就像图 4.7.1 中的其他例子那样。

提示 datetime 属性不会单独产生任何效果，但它可以用于在 Web 应用(如日历应用)之间同步日期和时间。这就是必须使用标准的机器可读格式的原因，这样，程序之间就可以使用相同的“语言”来共享信息。

提示 不能在 time 元素中嵌套另一个 time 元素，也不能在没有 datetime 属性的 time 元素中包含其他元素(只能包含文本)。

提示 在早期的 HTML5 说明中，time 元素可以包含一个名为 pubdate 的可选属性（要知道，HTML 这门语言还在不断地向前发展）。不过，pubdate 已不再是 HTML5 的一部分。在这里提到这个属性是为了让你偶然在较早的教程或书（如本书的第 7 版）中碰到该属性时不必犹豫是否应该使用它（当然不应该使用）。

理解有效的时间格式

datetime 属性（或者没有 datetime 属性的 time 元素）必须提供特定的机器可读格式的日期和时间。这可以简化为下面的形式：

YYYY-MM-DDThh:mm:ss

例如（当地时间）：

1985-11-03T17:19:10

表示“当地时间 1985 年 11 月 3 日下午 5 时 19 分 10 秒”。小时部分使用 24 小时制，因此表示下午 5 点应使用 17 而非 05。如果包含时间，秒是可选的。（也可以使用 hh:mm:sss 格式提供时间的毫秒数。注意毫秒数之前的符号是一个点。）

如果要表示时间段，则格式稍有不同。有好几种语法，不过最简单的形式为：

nh mm ns

（其中，三个 *n* 分别表示小时数、分钟数和秒数。）

图 4.7.1 中的最后一个例子就是表示时间段的实例。

◎ 全球日期和时间及时差

你也可以将日期和时间表示为世界时。在末尾加上字母 Z，就成了 UTC（Coordinated Universal Time，全球标准时间）。UTC 是主要的全球时间标准。（参见 https://en.wikipedia.org/wiki/Coordinated_Universal_Time。）

例如（使用 UTC 的世界时）：

1985-11-03T17:19:10Z

也可以通过相对 UTC 时差的方式表示时间。这时不写字母 Z，写上 -（减）或 +（加）及时差即可。

例如（含相对 UTC 时差的世界时）：

1985-11-03T17:19:10-03:30

表示“纽芬兰标准时（NST）1985 年 11 月 3 日下午 5 时 19 分 10 秒”（NST 比 UTC 晚 3 个半小时）。UTC 时区列表参见 http://en.wikipedia.org/wiki/List_of_time_zones_by_UTC_offset。

提醒一下，如果确实要包含 datetime，也不必像图 4.7.1 中的示例那样提供时间的完整信息。

4.8 解释缩写词

缩写词很常见，如 Jr.、M.D.，甚至 good ol' HTML。可以使用 `abbr` 元素标记缩写词并解释其含义（图 4.8.1 至图 4.8.3）。不必对每个缩写词都使用 `abbr`，只在需要帮助访问者了解该词含义的时候使用。

```
...
<body>

<p>The <abbr title="National Football
→ League">NFL</abbr> promised a <abbr
→ title="light amplification by
→ stimulated emission of radiation">
→ laser</abbr> show at 9 p.m. after every
→ night game.</p>

<p>But, that's nothing compared to what
→ <abbr>MLB</abbr> (Major League
→ Baseball) did. They gave out free
→ <abbr title="self-contained underwater
→ breathing apparatus">scuba</abbr> gear
→ during rain delays.</p>

</body>
</html>
```

图 4.8.1 使用可选的 `title` 属性提供缩写词的全称。另外，也可以将全称放在缩写词后面的括号里（这样做或许更好）。还可以同时使用这两种方式，并使用一致的全称。对于像 `laser`（激光）、`scuba`（水中呼吸器）这样的词汇，大多数人都很熟悉了，因此其实并无必要对它们使用 `abbr` 并提供 `title`，这里只是用它们来演示示例。

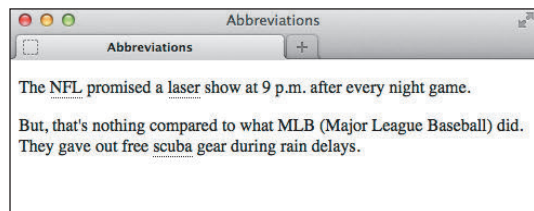


图 4.8.2 当缩写词有 `title` 属性时，Firefox 和 Opera 会添加虚线下划线以引起注意。可以通过 CSS 让其他浏览器也这样显示（参见提示）。

解释缩写词的步骤

- (1) 输入 `<abbr>`。
- (2) 作为可选的一步，输入 `title="expansion"`，其中 *expansion* 是缩写词的全称。
- (3) 输入 `>`。
- (4) 然后输入缩写词本身。
- (5) 最后输入 `</abbr>` 结束标签。
- (6) 作为可选的一步，输入一个空格和 (*expansion*)，其中 *expansion* 是缩写词的全称。

提示 通常，仅在缩写词第一次出现在屏幕上时给出其全称（通过 `title` 或括号的方式）。

提示 用括号提供缩写词的全称是解释缩写词最直接的方式，能让尽可能多的访问者看到这些内容（图 4.8.1）。例如，使用智能手机和平板电脑等触摸屏设备的用户可能无法移到 `abbr` 元素上查看 `title` 的提示框。因此，如果要提供缩写词的全称，应该尽量将它放在括号里。

提示 如果使用复数形式的缩写词，全称也要使用复数形式。

提示 作为对用户的视觉提示，Firefox 和 Opera 等浏览器会对带 `title` 的 `abbr` 文字使用虚线下划线（图 4.8.2）。想在其他浏览器中也这样显示，可以在样式表中加上这条语句：`abbr[title] { border-bottom: 1px dotted #000; }`。无论 `abbr` 是否添加了下划线样式，浏览器都会将 `title` 属性内容以提示框的形式显示出来（图 4.8.3）。

提示 如果看不到 `abbr` 有虚线下划线，试着为其父元素的 CSS 添加 `line-height` 属性（参见第 10 章）。