清华社“视频大讲堂”大系
网络开发视频大讲堂

1DVD

HTML5 + CSS3 精通

NLIC2970904634

从入门到精通

配套视频讲解227节

中小典型实例225个 综合实战案例31个 学习参考手册11部 实用网页模版83类

李东博◎编著

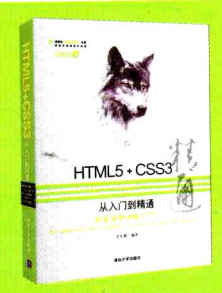
清华大学出版社

清华社“视频大讲堂”大系
网络开发视频大讲堂

执着于专业，精细于品质

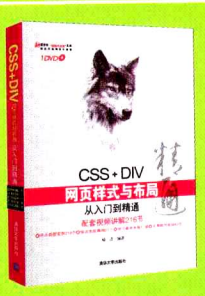
ISBN 978-7-302-30667-2

定价：79.80元



ISBN 978-7-302-30881-2

定价：69.80元



ISBN 978-7-302-30671-9

定价：69.80元

ISBN 978-7-302-30666-5

定价：79.80元

清华社“视频大讲堂”大系简介

“视频大讲堂”大系是清华社第六事业部重点打造的精品大系，该大系自2008年出版“软件开发视频大讲堂”以来，先后出版了多个子系列，本着“宁缺勿滥，件件精品”的原则，该大系取得了如下业绩：

- 4个品种荣获“全行业优秀畅销品种”
- 1个品种荣获2012年清华大学出版社“专业畅销书”一等奖
- 绝大多数品种在“全国计算机零售图书排行榜”同品种排行中名列前茅
- 截至目前该大系累计销售超过55万册
- 该大系已成为近年来清华社计算机专业基础类零售图书最畅销的品牌之一

清华大学出版社数字出版网站
WQBook 书文同泉
www.wqbook.com

ISBN 978-7-302-30881-2

定价：69.80元

（1DVD，含配套视频、参考手册、网页模板、教材源程序等）

内 容 简 介

《HTML5+CSS3 从入门到精通》（清华社“视频大讲堂”大系）通过基础知识+中小实例+综合案例的方式，讲述了用 HTML5+ CSS3 设计构建网站的必备知识，相对于权威指南、高级程序设计、开发指南同类图书，本书是一本适合快速入手的自学教程。内容有：创建 HTML5 文档，实战 HTML5 表单，实战 HTML5 绘画，HTML5 音频与视频，Web 存储，离线应用，Workers 多线程处理，Geolocation 地理位置等技术。CSS3 部分主要介绍了 CSS3 概述，CSS 选择器，文本、字体与颜色，背景和边框，2D 变形，设计动画，网页布局、用户界面以及 CSS3 的其他新特性。

《HTML5+CSS3 从入门到精通》内容涵盖了 HTML5+CSS3 的所有重要特性，通过大量实际案例对 HTML5+CSS3 的重要特性进行了详细讲解，内容全面丰富，易于理解，能够帮助读者提升实际应用技能。

本书内容翔实、结构清晰、循序渐进，基础知识与案例实战紧密结合，既可作为 HTML5+CSS3 初学者的入门教材，也适合作为中高级用户对新技术作进一步学习的参考用书。本书显著特色有：

1. 同步视频讲解，让学习更为直观高效。227 节大型高清同步视频讲解，先看视频再学习效率更高。
2. 海量精彩实例，用实例学更轻松快捷。225 个精彩实例，模仿练习是最快捷的学习方式。
3. 精选实战案例，为高薪就业牵线搭桥。31 个实战案例展示可为以后就业积累经验。
4. 完整学习套餐，为读者提供贴心服务。参考手册 11 部、网页模版 83 类、素材源程序，让学习更加方便。
5. 讲解通俗翔实，看得懂学得会才是硬道理。

图书在版编目（CIP）数据

HTML5+CSS3 从入门到精通/李东博编著. —北京：清华大学出版社，2013
（清华社“视频大讲堂”大系　网络开发视频大讲堂）

ISBN 978-7-302-30881-2

I. ①H…　II. ①李…　III. ①超文本标记语言-程序设计　②网页制作工具　IV. ①TP312　②TP393.092

中国版本图书馆 CIP 数据核字（2012）第 291406 号

责任编辑：赵洛育
封面设计：李志伟
版式设计：文森时代
责任校对：张兴旺
责任印制：王静怡

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：清华大学印刷厂
装 订 者：北京市密云县京文制本装订厂
经　　销：全国新华书店
开　　本：203mm×260mm　　**印　　张**：35　　**字　　数**：983 千字
（附 1DVD，含配套视频、参考手册、网页模板、素材源程序等）
版　　次：2013 年 6 月第 1 版　　**印　　次**：2013 年 6 月第 1 次印刷
印　　数：1～5000
定　　价：69.80 元

产品编号：044322-01

前　言

Preface

互联网技术日新月异。2011 年以前，HTML5 和 CSS3 看起来还遥不可及，如今很多公司都已经开始运用这些技术了，Chrome、Safari、Firefox 和 Opera 等主流浏览器已经开始逐步实现对它们的支持。从前端开发技术看，互联网发展经历了三个阶段：第一阶段是 Web 1.0 的以内容为主的网络，主流技术是 HTML 和 CSS；第二阶段是 Web 2.0 的 Ajax 应用，热门技术是 JavaScript/DOM/异步数据请求；第三阶段是即将迎来的 HTML5+CSS3 技术，这两者相辅相成，使互联网又进入了一个崭新的时代。

HTML5+CSS3 奠定了打造下一代 Web 应用的基础，这两项技术可以让网站更易开发、更易维护、更具用户友好性。HTML5 被设计为跨平台的技术，只需要一个所有主流操作系统上都有的免费现代浏览器。最新版本的 Apple Safari、Google Chrome、Mozilla Firefox、Opera 以及 Microsoft Internet Explorer 都支持 HTML5 的许多特性。在 iPhone、iPad 及 Android 移动设备上预装的浏览器也都对 HTML5 提供了极好的支持。

本书系统地讲解了 HTML5 和 CSS3 的基础理论和实际运用技术，通过大量实例对 HTML5 和 CSS3 进行深入浅出的分析，着重讲解如何用 HTML5+CSS3 进行 Web 应用和网页布局。全书注重实际操作，使读者在学习技术的同时，掌握 Web 开发和设计的精髓，提高综合应用的能力。

本书特色

☑ **系统的基础知识**

本书系统地讲解了 HTML5+CSS3 技术在网页设计中各种应用的知识，从为什么要用 HTML5 开始讲解，循序渐进，配合大量实例帮助读者奠定坚实的理论基础，做到知其所以然。

☑ **大量的案例实战**

本书设置大量应用实例，重点强调具体技术的灵活应用，并且全书结合了作者长期的网页设计制作和教学经验，使读者真正做到学以致用。

☑ **深入解剖 HTML5+CSS3 应用开发和布局**

本书用相当多的篇幅重点介绍了用 HTML5+CSS3 进行应用开发和布局的方法和技巧，配合经典的布局案例，帮助读者掌握 HTML5+CSS3 最核心的应用技术。

☑ **精选综合实例**

本书每章都会精选多个常见 Web 开发和设计的综合实例，帮助读者总结前面所学知识，综合应用各种技术、方法和技巧，提高读者综合应用的能力。

本书内容

本书分为两大部分，共 19 章，具体结构划分如下。

第一部分：HTML5 部分，包括第 1 章～第 10 章。这部分主要介绍了 HTML5 相关基础知识，包括 HTML5 概述、Web 开发历史、HTML5 文档结构异同与创建 HTML5 文档、设计 HTML5 表单、使

用 HTML5 绘画、HTML5 音频与视频、如何实现 Web 本地存储、如何实现 Web 离线应用、如何设计多线程应用、如何实现移动互联网中地图定位等技术。

第二部分：CSS3 部分，包括第 11 章～第 19 章。这部分主要讲解了 CSS3 的新特性和新用法，以实现在简单的代码中能够设计更加精彩的网页效果。主要内容包括 CSS3 概述，CSS 选择器，定义文本、字体与颜色，设计背景和边框，使用 2D 变形，设计动画，设计多列和流动网页布局，优化用户界面以及 CSS3 的其他新特性。

Note

本书读者

- ☑ 希望系统学习网页设计、网站制作的初学者、进阶者
- ☑ 从事网页设计制作和网站建设的专业人士
- ☑ 各大中专院校相关专业的老师、学生
- ☑ 相关培训机构的学员

本书约定

本书主要面向熟悉 HTML 和 CSS 的 Web 开发人员。初学者也可以从本书获益，读者还应该具备基本的 HTML、CSS、JavaScript 知识，我们会用这些知识来创建各种应用方案。

本书中上机练习的示例要用到 Firefox、Chrome 和 Opera 浏览器，因此，为了测试所有内容，读者需要安装上述所有类型的最新版本浏览器，因为各种浏览器的实现都稍有差异。

读者还需要安装 Internet Explorer 来测试自己的站点，以便确保示例的兼容性。如需针对不同版本的 Internet Explorer 测试示例，可以下载 IE Tester Windows 版，因为它可同时支持 IE6、IE7 和 IE8。对于非 Windows 用户，可以考虑使用 VirtualBox 或者 VMware 等虚拟机，或者使用 CrossBrowserTesting 和 MogoTest 等服务。

为了给读者提供更多的学习资源，同时弥补本书篇幅有限的遗憾，本书提供了很多参考链接，许多本书中无法详细介绍的问题都可以通过这些链接找到答案。因为这些链接地址会因时间而有所变动或调整，所以在此说明，这些链接地址仅供参考，无法保证这些地址都是长期有效的。

本书所列的插图可能会与读者实际环境中的操作界面有所差别，这可能是由于操作系统平台、浏览器版本等不同而引起的，在此特别说明，读者应该以实际情况为准。

为了帮助读者快速上手，在一般情况下，读者可以在程序和文档中自由使用本书中的示例代码。

关于我们

参与本书编写的人员包括咸建勋、奚晶、文菁、李静、钟世礼、李增辉、甘桂萍、刘燕、杨凡、李爱芝、余乐、孙宝良、余洪萍、谭贞军、孙爱荣、何子夜、赵美青、牛金鑫、孙玉静、左超红、蒋学军、邓才兵、袁江和李东博等。由于作者水平有限，书中疏漏和不足之处在所难免，欢迎读者朋友不吝赐教。广大读者如有好的建议、意见，或在学习本书时遇到疑难问题，可以联系我们，我们会尽快为您解答，服务邮箱为：design1993@163.com，liulm75@163.com。

编　者

目 录

Contents

Web 开发新时代

从 2010 年开始，HTML5 和 CSS3 就一直是互联网技术中最受关注的两个话题。2010 年 MIX 10 大会上微软的工程师在介绍 IE9 时，从前端技术的角度把互联网的发展分为 3 个阶段：第一阶段是以 Web 1.0 为主的网络阶段，前端主流技术是 HTML 和 CSS；第二阶段是 Web 2.0 的 Ajax 应用阶段，热门技术是 JavaScript/DOM/异步数据请求；第三阶段是即将迎来的 HTML5+CSS3 阶段，这两者相辅相成，使互联网又进入了一个崭新的时代。本章将重点介绍 HTML5 语言基础，为将来系统地学习 HTML5+CSS3 编程奠定扎实的基础。

1.1 HTML5 概述

Note

2004 年成立的 Web 超文本应用技术工作组（WHATWG）创立了 HTML5 规范，同时开始专门针对 Web 应用开发新的功能。2006 年，W3C 介入 HTML5 的开发，并于 2008 年发布了 HTML5 的工作草案。2009 年，W3C 停止了对 XHTML2 的更新。2010 年 HTML5 开始用于解决实际问题。这时各大浏览器厂商开始对旗下产品进行升级以支持 HTML5 的新功能，因此，HTML5 规范也得到了持续性的完善。

1.1.1 HTML5 新特性

HTML5 是基于各种全新的理念进行设计的，这些设计理念体现了对 Web 应用的可能性和可行性的新认识，下面简单介绍 HTML5 语言的特征和优势，以便提高读者自学 HTML5 的动力，明确学习目标。

1. 兼容性

考虑到互联网上 HTML 文档已经存在二十多年了，因此支持所有现存 HTML 文档是非常重要的。HTML5 不是颠覆性的革新，它的核心理念就是要保持与过去技术的兼容和过渡。一旦浏览器不支持 HTML5 的某项功能，针对该功能的备选行为就会悄悄运行。

2. 合理性

HTML5 新增加的元素都是对现有网页和用户习惯进行跟踪、分析和概括而推出的。例如，Google 分析了上百万的页面，从中分析出了 DIV 标签的通用 ID 名称，并且发现其重复量很大，如很多开发人员使用 来标记页眉区域，为了解决实际问题，HTML5 就直接添加一个<header>标签。也就是说，HTML5 新增的很多元素、属性或者功能都是根据现实互联网中已经存在的各种应用进行技术精炼，而不是在实验室中理想化地虚构新功能。

3. 效率

HTML5 规范是基于用户优先准则编写的，其宗旨是用户即上帝，这意味着在遇到无法解决的冲突时，规范会把用户放到第一位，其次是页面作者，再次是实现者（或浏览器），接着是规范制定者（W3C/WHATWG），最后才考虑理论的纯粹性。因此，HTML5 的绝大部分是实用的，只是在有些情况下还不够完美。例如，下面的几种代码写法在 HTML5 中都能被识别。

```
id="prohtml5"
id=prohtml5
ID="prohtml5"
```

当然，上面几种写法比较混乱，不够严谨，但是从用户开发角度考虑，用户不在乎代码怎么写，根据个人习惯书写反而提高了代码编写效率。当然，我们并不提倡初学者一开始写代码就这样随意、不严谨。

4. 安全性

为保证安全性，HTML5 规范中引入了一种新的基于来源的安全模型，该模型不仅易用，而且各

种不同的 API 都可通用。这个安全模型可以不需要借助于任何所谓聪明、有创意却不安全的 hack 就能跨域进行安全对话。

5. 分离

在清晰分离表现与内容方面，HTML5 迈出了很大一步。HTML5 在所有可能的地方都努力进行了分离，包括 HTML 和 CSS。实际上，HTML5 规范已经不支持老版本 HTML 的大部分表现功能了。

Note

6. 简化

HTML5 要的就是简单，避免不必要的复杂性。为了尽可能简化，HTML5 做了以下改进：

- ☑ 以浏览器原生能力替代复杂的 JavaScript 代码。
- ☑ 简化的 DOCTYPE。
- ☑ 简化的字符集声明。
- ☑ 简单而强大的 HTML5 API。

7. 通用性

通用访问的原则可以分成 3 个概念。

- ☑ 可访问性：出于对残障用户的考虑，HTML5 与 WAI（Web 可访问性倡议）和 ARIA（可访问的富 Internet 应用）做到了紧密结合，WAI-ARIA 中以屏幕阅读器为基础的元素已经被添加到 HTML 中。
- ☑ 媒体中立：如果可能的话，HTML5 的功能在所有不同的设备和平台上应该都能正常运行。
- ☑ 支持所有语种：如新的<ruby>元素支持在东亚页面排版中会用到的 Ruby 注释。

8. 无插件

在传统 Web 应用中，很多功能只能通过插件或者复杂的 hack 来实现，但在 HTML5 中提供了对这些功能的原生支持。插件的方式存在很多问题：

- ☑ 插件安装可能失败。
- ☑ 插件可以被禁用或屏蔽（如 Flash 插件）。
- ☑ 插件自身会成为被攻击的对象。
- ☑ 插件不容易与 HTML 文档的其他部分集成，因为存在插件边界、剪裁和透明度问题。

以 HTML5 中的 canvas 元素为例，以前在 HTML4 的页面中较难画出对角线，而有了 canvas 元素就可以很轻易地实现了。基于 HTML5 的各类 API 的优秀设计，可以轻松地对它们进行组合应用。例如，从 video 元素中抓取的帧可以显示在 canvas 中，用户单击 canvas 即可播放这帧对应的视频文件。

1.1.2　HTML5 组织

HTML5 开发主要由下面 3 个组织负责和实施。

- ☑ WHATWG：由来自 Apple、Mozilla、Google 和 Opera 等浏览器厂商的人员组成，成立于 2004 年。WHATWG 开发 HTML 和 Web 应用 API，同时为各浏览器厂商以及其他有意向的组织提供开放式合作。
- ☑ W3C：W3C 下辖的 HTML 工作组，目前负责发布 HTML5 规范。
- ☑ IETF（因特网工程任务组）：这个任务组下辖 HTTP 等，是负责开发 Internet 协议的团队。HTML5 定义的一种新 API（WebSocket API）所依赖的 WebSocket 协议，正由 IETF 工作组负责开发。

1.1.3 HTML5 构成

Note

HTML5 主要包括下面这些功能。

- ☑ Canvas（2D 和 3D）
- ☑ Channel 消息传送
- ☑ Cross-document 消息传送
- ☑ Geolocation
- ☑ MathML
- ☑ Microdata
- ☑ Server-Sent Events
- ☑ Scalable Vector Graphics（SVG）
- ☑ WebSocket API 及协议
- ☑ Web Origin Concept
- ☑ Web Storage
- ☑ Web SQL database
- ☑ Web Workers
- ☑ XMLHttpRequest Level 2

HTML5 发展的速度非常快，因此不用担心浏览器的支持问题。读者可以访问 www.caniuse.com，该网站按照浏览器的版本提供了详尽的 HTML5 的功能支持情况。如果通过浏览器访问 www.html5test.com，该网站会直接显示用户浏览器对 HTML5 规范的支持情况。另外，还可以使用 Modernizr（JavaScript 库）进行特性检测，它提供了非常先进的 HTML5 和 CSS3 检测功能。建议读者使用 Modernizr 检测当前浏览器是否支持某些特性。

1.2 HTML5 设计原理

设计原理是 Web 发展背后的驱动力，也是通过 HTML5 反映出来的某种思维方式。软件就像所有技术一样，具有天然的独裁性。代码必然会反映作者的选择、偏见和期望。任何开放的标准，都应该追求以下几点：

- ☑ 简化最常见的任务，让不常见的任务不至于太麻烦。
- ☑ 只为 80%设计。
- ☑ 给内容创建者最大的权力。
- ☑ 默认设置智能化。

在制定设计原理时，很多人用了很多时间却抓不住重点，因为他们想取悦所有人。关键在于我们不是要取悦所有人，而是要明确哪些人是最重要的。

意大利经济学家帕累托提出“世界上 20%的人口拥有 80%的财富”。这个比例又暗合了自然界各个领域的幂律分布现象。总之，无论是编写软件，还是制造什么东西，都是一样的，即 20%的努力可以触及 80%的用例，最后 20%的用例则需要付出 80%甚至更多的努力。因此，有时候据此确定只为 80%设计是很合理的，因为我们知道为此只要付出 20%的努力即可。

1.2.1 HTML 开发历程

Note

HTML 最早是从 2.0 版开始的，实际上并没有 HTML 1.0 版官方规范。HTML Tags 文档可以算作 HTML 的第一个版本，但它却不是一个正式的版本。第一个正式版本 HTML 2.0 也不是出自 W3C 之手，而是由 IETF（Internet Engineering Task Force，因特网工程任务组）制定的，从第三个版本开始，W3C（World Wide Web Consortium，万维网联盟）开始接手并负责后续版本的制定工作。

20 世纪 90 年代，HTML 有过几次快速的发展。众所周知，那时构建网站是一项十分复杂的工程，浏览器大战曾令人头疼不已，市场竞争的结果就是各家浏览器里都塞满了各种专有的特性，都试图在专有特性上胜人一筹。当时的混乱程度不堪回首，HTML 还重不重要，或者它作为 Web 格式的前景如何，谁都说不清楚。

从 1997 年到 1999 年，HTML 的版本从 3.2 到 4.0，再到 4.01，经历了非常快的发展。问题是到了 4.01 的时候，W3C 的认识发生了倒退，W3C 并没有停止开发这门语言，只不过他们对 HTML 不再感兴趣了。在 HTML 4.01 之后，W3C 提出了 XHTML 1.0 的概念。虽然听起来完全不同，但 XHTML 1.0 与 HTML 4.01 其实是一样的。

这两个规范的内容是一样的，词汇表是一样的，所有的元素是一样，所有的属性也都是一样的，唯一不同的就是 XHTML 1.0 要求使用 XML 语法。也就是说，所有属性都必须使用小写字母，所有元素也必须使用小写字母，所有属性值都必须加引号，所有的标签都必须有结束标签，对 img 和 br 等孤标签，要使用自结束标签。

从规范本身的内容来看，本质是相同的，不同之处就是编码风格，因为浏览器读取符合 HTML 4.01、HTML 3.2 或者 XHTML 1.0 规范的网页都没有问题，对浏览器来说这些网页都是一样的，都会生成相同的 DOM 树，只不过用户更喜欢 XHTML 1.0，因为不少人认同它比较严格的编码风格。

到了 2000 年，Web 标准项目（Web Standards Project）的活动开展得如火如荼，开发人员对浏览器里包含的那些乱七八糟的专有特性已经忍无可忍了。当时 CSS 有了长足的发展，而且与 XHTML 1.0 的结合也很紧密，CSS+XHTML 1.0 可以算是最佳实践了。虽然 HTML 4.01 与 XHTML 1.0 没有本质上的区别，但是大部分开发人员接受了这种组合。专业的开发人员能做到元素全部小写，属性全部小写，属性值也全部加引号。由于专业人员起到了模范带头作用，越来越多的人也都开始支持这种语法。

XHTML 1.0 之后是 XHTML 1.1，只是小数点后面的数字变成了 1，而且从词汇表的角度看，规范本身没有什么新内容，元素、属性也都相同，唯一的变化就是必须把文档标记为 XML 文档。而在使用 XHTML 1.0 的时候，还可以把文档标记为 HTML。

但是，这样做就带来很多问题：首先，把文档标记为 XML 后，IE 浏览器不能处理。当然，IE9 及其以上版本是可以处理的。作为全球领先的浏览器，IE 无法处理接收到的 XML 文档类型的文档，而规范又要求以 XML 文档类型来发送文档，这对于广大用户来说，是一件很痛苦的事情。

所以说 XHTML 1.1 有点脱离现实，而用户不想把文档以 XML 格式发送给那些能够理解 XML 的浏览器，则是因为 XML 的错误处理模型。XML 的语法，无论是属性小写，元素小写，还是始终要给属性值加引号，这些都没有问题，但 XML 的错误处理模型却是这样的：如果解析器遇到错误，停止解析。如果把 XHTML 1.1 标记为 XML 文档类型，假设用 Firefox 打开这个文档，而文档中有一个符号没有正确编码，就算整个页面中只有这一处错误，浏览器也会死掉，用户将看不到任何网页内容。根据 XML 规范，这样处理是正确的，对 Firefox 而言，遇到错误就停止解析，并且不呈现其他任何内容是严格按照 XML 规范处理的。因为它不是 HTML，HTML 根本没有错误处理模型，但根据 XML 规范，这样做没错。这就是为什么人们不会把文档标记为 XML 的另一个原因。

Note

接下来，新的版本是 XHTML2，但是这个版本并没有完成。从理论的角度来说，XHTML2 是一个非常好的规范。如果所有人都同意使用的话，也一定是一个非常好的格式。只不过，它还不够实际。

首先，XHTML2 仍然使用 XML 错误处理模型，用户必须保证以 XML 文档类型发送文档；其次，XHTML2 中有意不再向后兼容已有的 HTML 的各个版本，甚至曾经讨论过废除 img 元素，这对每天都在做 Web 开发的人来说确实有点难以接受，理论上分析，使用 object 元素可能会更好。

因此，无论 XHTML2 在理论上是多么完美的一种格式，却从未有机会付诸实践。之所以难以将其付诸实践，就是因为开发人员永远不会支持它，它不向后兼容。同样，浏览器厂商也不会支持它，必须要保证浏览器向后兼容。

为什么 XHTML 1.1 没有像 XML 那样得到真正广泛地应用？为什么 XHTML2 从未落到实处？因为它违反了一条设计原理，这条设计原理就是著名的伯斯塔尔法则：发送时要保守，接收时要开放。

接收的时候要开放，而这也正是 Web 得以构建的基础。开发浏览器的人必须敞开胸怀，接收所有发送给浏览器的东西，因为它们过去一直都在接收那些不够标准的东西，Web 上的很多文档都不规范，但那正是 Web 发展的动力。从某种角度讲，Web 走的正是一条混沌发展之路，虽然混沌，却非常美丽诱人。在 Web 上，格式不规范的文档随处可见，如果所有人都能够写出精准的 XML，所有文档的格式都十分正确，那当然好，可是那不现实。

作为专业人士，在发送文档的时候应该保守一些，尽量采用最佳实践，尽量确保文档格式良好，但从浏览器的角度说，它们必须以开放的姿态去接收任何文档。

XHTML 1.1 和 XHTML2 都使用 XML 错误处理模型，但这个错误处理模型太苛刻了，它不符合“接收时开放”这个法则，遇到一个错误就停止解析怎么能叫开放呢？我们只能说它与伯斯塔尔法则是对立的。

1.2.2 HTML5 开发动力

在 20 世纪末期，W3C 琢磨着改良 HTML 语言，“HTML 也许还可以更长寿一点，只要把我们放在 XHTML 上的时间和精力拿出一部分来，就可以提升一下 HTML 中的表单，可以让 HTML 更接近编程语言，就可以让它更上一层楼。”

于是，在 2004 年 W3C 成员内部的一次研讨会上，Opera 公司的代表伊恩·希克森（Ian Hickson）提出了一个扩展和改进 HTML 的建议。他建议新任务组可以跟 XHTML2 并行，但是在已有 HTML 的基础上开展工作，目标是对 HTML 进行扩展。但是 W3C 投票表示反对，因为 HTML 已经死了，XHTML2 才是未来的方向。然后，Opera、Apple 等浏览器厂商以及其他一些成员脱离了 W3C，他们成立了 WHATWG （Web Hypertext Applications Technology Working Group，Web 超文本应用技术工作组），这就为 HTML5 将来的命运埋下了伏笔。

WHATWG 决定完全脱离 W3C，在 HTML 的基础上开展工作，向其中添加一些新东西。这个工作组的成员里有浏览器厂商，因此他们可以保证实现各种新奇、实用的点子。结果，大家不断提出一些好点子，并且逐一做到了浏览器中。

WHATWG 的工作效率很高，不久就初见成效。在此期间，W3C 的 XHTML2 没有什么实质性的进展。在 2006 年，蒂姆·伯纳斯·李（Tim Berners-Lee）写了一篇博客反思 HTML 的发展历史，“你们知道吗？我们错了。我们错在企图一夜之间就让 Web 跨入 XML 时代，我们的想法太不切实际了，是的，也许我们应该重新组建 HTML 工作组了。”

W3C 在 2007 年组建了 HTML5 工作组。这个工作组面临的第一个问题，毫无疑问就是“我们是从头开始做起呢，还是在 2004 年成立的那个叫 WHATWG 的工作组既有成果的基础上开始工作呢？”

Note

答案是显而易见的，他们当然希望从已经取得的成果着手，以之为基础展开工作。于是他们又投了一次票，同意在 WHATWG 工作成果的基础上继续开展工作。

第二个问题就是如何理顺两个工作组之间的关系。W3C 这个工作组的编辑应该由谁担任？是不是还让 WHATWG 的编辑，也就是现在 Google 的伊恩 • 希克森来兼任？于是他们又投了一次票，赞成让伊恩 • 希克森担任 W3C HTML5 规范的编辑，同时兼任 WHATWG 的编辑，更有助于新工作组开展工作。

这就是他们投票的结果，也就是我们今天看到的局面：一种格式，两个版本。WHATWG 的网站上有这个规范，而 W3C 的站点上同样也有一份。

如果不了解内情，你很可能会产生这样的疑问："哪个版本才是真正的规范？"当然，这两个版本内容是一样的，基本上相同，但这两个版本将来还会分道扬镳。现在已经有分道扬镳的迹象了：W3C 最终要制定一个具体的规范，这个规范会成为一个工作草案，定格在某个历史时刻，而 WHATWG 还在不断地迭代，即使是目前的 HTML5 也不能完全涵盖 WHATWG 正在从事的工作。最准确的理解就是 WHATWG 正在开发一项简单的 HTML 或 Web 技术，因为这才是他们工作的核心目标。然而，同时存在两个这样的工作组，这两个工作组同时开发一个基本相同的规范，这无论如何也容易让人产生误解，误解就可能造成麻烦。

其实这两个工作组背后各自有各自的流程，因为它们的理念完全不同。在 WHATWG，可以说是一种独裁的工作机制。伊恩 • 希克森是编辑，他会听取各方意见，在所有成员各抒己见，充分陈述自己的观点之后，他批准自己认为正确的意见。而 W3C 则截然相反，可以说是一种民主的工作机制。所有成员都可以发表意见，而且每个人都有投票表决的权利。这个流程的关键在于投票表决。从表面上看，WHATWG 的工作机制让人难以接受，W3C 的工作机制听起来让人很舒服，至少体现了人人平等的精神。但在实践中，WHATWG 的工作机制运行得非常好，这主要归功于伊恩 • 希克森。他在听取各方意见时，始终可以做到丝毫不带个人感情色彩。

从原理上讲，W3C 的工作机制很公平，而实际上却非常容易在某些流程或环节上卡壳，造成工作停滞不前，一件事情要达成决议往往需要花费很长时间。那到底哪种工作机制最好呢？最好的工作机制是将二者结合起来。而事实也是两个规范制定主体在共同制定一份相同的规范，这倒是非常有利于两种工作机制相互取长补短。

两个工作组之所以能够同心同德，主要原因是 HTML5 的设计思想。因为他们从一开始就确定了设计 HTML5 所要坚持的原则。结果，我们不仅看到了 HTML5 语言规范，也就是 W3C 站点上公布的那份文档，还在 W3C 站点上看到了另一份文档，也就是 HTML5 设计原理。

1.2.3 HTML5 设计理念

HTML5 是一个里程碑式的规范，它为下一代 Web 发展指明了方向，下面我们就来探析 HTML5 语言的设计理念。这个设计理念相当于共同行动纲领，无论 W3C 与 WHATWG 之间有多大的分歧，他们都将遵循这个设计理念。

☑ 避免不必要的复杂性

例如，使用 HTML 4.01 规范时，如果定义 doctype，需要输入很长的字符串：

```
<!DOCTYPE html PUBLIC "-//W3C/DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
```

很少有人能够记住这行代码，本来它要告诉浏览器的是"这个文档是 XHTML 1.0 的文档"。那么在 HTML5 中，省掉不必要的复杂性，doctype 就简化成了：

```
<!DOCTYPE html>
```

第一次看到这个 doctype 的时候，或许很难想到，这个 doctype 要告诉浏览器什么呢？这个文档是 HTML 吗？难道这是有史以来唯一一个 HTML 版本吗？HTML 今后永远不会再有新版本了吗？

Note

实际上这个 doctype 并没有 HTML 版本区分的意思。我们先搞清楚为什么文档一开头就要写 doctype。这不是写给浏览器看的，而是写给验证器看的。也就是说，之所以要在文档一开头写那行 XHTML 1.0 的 doctype，是为了让验证器按照该 doctype 来验证我的文档。

浏览器反倒无所谓了。假设写的是 HTML 3.2 文档，文档开头写的是 HTML 3.2 的 doctype，而在文档中某个地方使用了 HTML 4.01 中才出现的一个元素，浏览器会怎么处理这种情况？它会因为这个元素出现在比 doctype 声明的 HTML 版本更晚的规范中，就不解释呈现该元素吗？不会，它照样会解释呈现该元素，别忘了伯斯塔尔法则，浏览器在接收的时候必须要开放。因此，它不会检查任何格式类型，而验证器会，验证器才关心格式类型。这才是存在 doctype 的真正原因。

而按照 HTML5 的另一个设计原理，它必须向前向后兼容，兼容未来的 HTML 版本，不管是 HTML6、HTML7，还是其他版本，都要与当前的 HTML5 兼容。因此，把一个版本号放在 doctype 里面没有多大的意义，对验证器也一样。

但是有一种情况下，使用 doctype 会影响到浏览器，这也是为了达到某种特殊的目的才使用 doctype。当初微软在引入 CSS 的时候，走在了标准的前头，他们率先在浏览器中支持 CSS，也推出了自己的盒模型。后来标准发布了，但标准中使用了不一样的盒模型。微软想支持标准，但也想向后兼容自己过去推出的编码方式。于是，他们想出了一个非常巧妙的主意，那就是利用有效的 doctype 来触发标准模式，而不是兼容模型（quiks mode）。这个想法非常巧妙。我们今天也都是在这样做，在向文档中加入 doctype 时，就相当于声明了使用标准模式，但这并不是发明 doctype 的本意。这只是为了达到特殊的目的在利用 doctype。

HTML5 规范的本质是不追求理论上的完美。HTML5 所体现的不是给用户一个简短好记的 doctype，好记的 doctype 也无法适应现有的浏览器。因此，doctype 不仅从理论上看简短好记，而且在实践中仍然可以触发标准模式。

还有一个例子，同样可以说明规范是如何省略不必要的复杂性，避免不必要的复杂性的。如果前面的文档使用的是 HTML 4.01，假设要指定文档的字符编码，理想的方式是通过服务器在头部信息中发送字符编码，不过也可以在文档这个级别上指定：

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

如果想指定文档使用 UTF-8 编码，只能添加上面一行代码。在 HTML 4.01 中需要这样做。要是在 XHTML 1.0 指定同样的编码，就得在一个开始的 XML 标签中声明 meta 元素。

```
<?xml version="1.0" encoding="UTF-8" ?>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

在 HTML5 中，只要输入下面代码即可。

```
<meta charset="utf-8">
```

同样，这样写也是有效的。它不仅适用于最新版本的浏览器，仍在使用的以往版本的浏览器中也同样有效，因为在把这些 meta 元素输入浏览器时，浏览器都会正确解释。

关于省略不必要的复杂性的例子还有很多，但关键是既能避免不必要的复杂性，又不会妨碍在现有浏览器中使用。在 HTML5 中，如果使用 link 元素链接到一个样式表，先定义 rel="stylesheet"，然后再定义 type="text/css"，这样就重复了。对浏览器而言，不需要同时看到这两个属性，只要看到 rel="stylesheet"就够了，因为它可以猜出来要链接的是一个 CSS 样式表，所以就不用再指定 type 属

性了。

同样，如果使用 script 元素，定义 type="text/javascript"，浏览器也可以识别。对 Web 开发而言，谁还使用其他的脚本语言吗？愿意的话，可以添加一个 type 属性，也可以什么都不写，浏览器自然会假设在使用 JavaScript。

Note

☑　支持已有的内容

支持已有的内容。这一点非常重要，因为很多人都认为 HTML5 很新，它应该代表着未来发展的方向，应该把 Web 推向一个新的发展阶段。当然我们都会考虑让 Web 的未来发展得更好，但工作组则必须考虑过去。别忘了 WHATWG 这个工作组中有很多人代表的是浏览器厂商，他们肯定是要考虑支持已有内容的。只要想构建一款浏览器，就必须记住一个原则：必须支持已有的内容。

【示例 1】本示例展示了编写同样内容的 4 种不同方式。上面是一个 img 元素，下面是带一个属性的段落元素，4 种写法唯一的不同点就是语法。把其中任何一段代码交给浏览器，浏览器都会生成相同的 DOM 树，没有任何问题。从浏览器的角度看，这 4 种写法没有区别，因此在 HTML5 中可以随意使用下列语法。

```
<img src="foo" alt="bar" />
<p class="foo">Hello world</p>
<img src="foo" alt="bar">
<p class="foo">Hello world
<IMG SRC="foo" ALT="bar">
<P CLASS="foo">Hello world</P>
<img src=foo alt=bar>
<p class=foo>Hello world</p>
```

HTML5 同时允许这些写法，对于那些写了很多年 XHTML 1.0 代码，已经非常适应严格的语法的开发人员来说，是难以适应的，但站在浏览器的角度上，这些写法实际上都是一样的，确实没有什么问题。

HTML5 必须支持已经存在的内容，而已有的内容就是这个样子的。根据伯斯塔尔法则，浏览器没有别的选择。

有人可能会说“这样不行。我觉得语言本身应该提供一种开关，让作者能够表明自己想做什么。”比如说，想使用某种特定的语法，像 XHTML，而不是使用其他语法。这些人的想法可以理解，但没必要在语言里设置开关。因为我们讨论的只是编码风格或者写作风格，跟哪种语法正确与否无关。专业人士可以使用 lint 工具（一种软件质量保证工具，或者说是一种更加严格的编译器。它不仅可以像普通编译器那样检查出一般的语法错误，还可以检查出那些虽然完全合乎语法要求，但很可能是潜在的、不易发现的错误）。

对其他技术我们也在使用 lint 工具，比如对 JavaScript 使用 lint 工具。JavaScript 同样也是比较混乱、不严谨的典型，但它非常强大，恰恰是因为它混乱、不严谨，而且有很多不同的编码方式。在 JavaScript 中可以在每条语句末尾加上分号，但不是必需的，因为 JavaScript 会自动插入分号。

正因为如此，才有了像 JSlint 这样的工具。在道格拉斯 • 克劳克福德（Douglas Crockford）的网站 jslint.org 上写着“JSlint 可能会伤害你的感情”，但这确实是个非常棒的工具，它可以把 JavaScript 代码变得完美无瑕。如果通过 JSlint 运行 JavaScript，它会告诉你你的 JavaScript 代码有效，但写法不妥。特别是对团队，对于要使用统一的编码风格的团队，JSlint 是非常方便的工具。

个人认为，不仅对团队来说，就算是自己写代码，也要坚持一种语法风格。从浏览器解析的角度讲，不存在哪种语法比另一种更好的问题，但作为专业人士，必须能够自信地讲“这就是我的编码风格”。然而，语言里并不需要内置这种开关，用户可以使用 lint 工具来统一编码风格。现在就来说说

lint 工具。大家可以登录 htmllint.com，在其中运行 HTML5 文档，它会帮用户检查属性值是否加了引号，元素是否小写，还可以通过选中复选框来设置其他检查项。

☑ 解决现实的问题

HTML5 的另一个设计原理是解决现实的问题。显而易见的是，解决各种问题的格式和规范已经比比皆是了，因此，这个原理其实是要解决理论问题，而非解决现实问题。这条设计原理是要从理论上承认人们普遍存在的问题，消除敏感问题。但是，那些格式和规范要解决的都是理论问题，而非现实问题。这条设计原理才是真正要解决今天人们所面临的现实问题、令人头疼的问题。

【示例 2】假设使用 HTML 4 或 XHTML 1，页面中已经有了一块内容，想给整块内容添加一个链接，怎么办？问题是这块内容里包含一个标题，一个段落，也许还有一张图片。如果想使它们全部都可以生成链接，必须使用 3 个链接元素。于是，得先把光标放在标题（比如说 h2 元素）中，写一个链接标签，然后再选中所有要包含到链接里的文本。接着，把光标放在段落里，写一个链接标签，然后把段落中的文本放在链接里。

```
<h2><a href="#">标题文本</a></h2>
<p><a href="#">段落文本</a></p>
```

在 HTML5 中，只要简单地把所有内容都写在一个链接元素中就可以了。

```
<a href="#">
    <h2>标题文本</h2>
    <p>段落文本</p>
</a>
```

链接包含的都是块级元素，但现在可以用一个元素包含它们。这样，碰到类似的情形，就必须给几个块级元素加上相同的链接，因为这样做解决了一个现实的问题：不必因此重新写代码来支持这种写法。这种写法其实早就存在于浏览器中了，因为早就有人这样写了，当然以前这样写是不合乎规范的。所以说 HTML5 解决了现实的问题，允许这样写了。

☑ 求真务实

求真务实对于 HTML 的含义是：在解决那些令人头痛的问题之前，先看看人们为应对这些问题都想出了哪些办法。集中精力去理解这些解决方案才是当务之急。

HTML5 中新的语义元素就是遵循求真务实原理的反映。新增的元素不算多，谈不上无限的扩展性，但却不失为一件好事，尽管数量屈指可数，但意义却非同一般。这些新元素涉及头部（header）、脚部（footer）、分区（section）、文章（article）等，相信大家都不会觉得陌生。即便不使用 HTML5，也应该熟悉这些名称，它们都是曾经使用过的类名，如 class="header"/"head"/"heading"，或 class="footer"/"foot"。当然，也可能是 ID，如 id="header"或 id="footer"。

```
<body>
    <div id="header">...</div>
    <div id="navigation">...</div>
    <div id="main">...</div>
    <div id="sidebar">...</div>
    <div id="footer">...</div>
</body>
```

在 HTML5 中，这些元素都可以换掉。说起新增的语义元素，它们价值的一方面就是可以用 HTML5 新增的元素把这些 div 都替换掉。

```
<body>
```

```
    <header>...</header>
    <nav>...</nav>
    <div id="main">...</div>
    <aside>...</aside>
    <footer>...</footer>
</body>
```

Note

在文档级别上使用这些元素没有问题，但是，假如新增这些元素仅仅是为了取代原来的 div，那就没有必要了。

虽然在这个文档中用这些新元素来替换 ID，但在个人看来，将它们作为类的替代品更有价值，因为这些元素在一个页面中不止可以使用一次，而是可以使用多次。没错，也可以为文档添加一个头部（header），再添加一个脚部（footer），但文档中的每个分区（section）照样也都可以有一个头部和一个脚部，而每个分区里还可以嵌套另一个分区，被嵌套的分区仍然可以有自己的头部和脚部。

这 4 个新元素是 section、article、aside 和 nav，之所以说它们强大，原因在于它们代表了一种新的内容模型，一种 HTML 中前所未有的内容模型——给内容分区。迄今为止，我们一直都在用 div 来组织页面中的内容，但与其他类似的元素一样，div 本身并没有语义；但 section、article、aside 和 nav 实际上是在明确地告诉用户，这一块就像文档中的另一个文档一样。位于这些元素中的任何内容，都可以拥有自己的概要、标题和脚部。

其中最为通用的 section，可以说是与内容最相关的一个，而 article 则是一种特殊的 section。aside 和 nav 也是特殊的 section。

当然，现在仍然可以使用 div 和类来描述页面中不同的部分：

```
<div class="item">
    <h2>...</h2>
    <div class="meta">...</div>
    <div class="content">
    </div>
    <div class="links">...</div>
</div>
```

其中包含的可能是有关内容作者的元数据，而下面会给出一些链接。在 HTML5 中，完全可以说这块内容就是一个文档，通过对内容分区，使用 section、article 或 aside，这一块完全是可以独立存在的。因此，可以使用 header 和 footer。

```
<section class="item">
    <header><h1>...</h1></header>
    <footer class="meta">...</footer>
    <div class="content">
    </div>
    <nav class="links">...</nav>
</section>
```

注意： 即便是 footer，也不一定非要出现在下面。header、footer、aside 和 nav 元素，最重要的是它们的语义，与位置没有关系。如果阅读 HTML5 规范，就会发现这些元素只跟内容有关。因此，放在 footer 中的内容也可以是署名、文章作者之类的信息，它只是所使用的一个元素，并没有说这个元素必须放在文档或者分区的下面。

在 HTML5 中，只要建立一个新的内容块，不管用 section、article、aside、nav，还是用别的元素，

Note

都可以在其中使用 H1，而不必担心这个块里的标题在整个页面中应该排在什么级别；使用 H2、H3 也都没有问题，因为现在可以把每个内容分区想象成一个独立的、能够从页面中拿出来的部分。此时，根据上下文的不同，这个独立部分中的 H1 在整个页面中也许会扮演 H2 或 H3 的角色，这取决于它在文档中出现的位置。面对这个突如其来的变化，也许有的人暂时不能理解，但这才是 HTML5 中这些新语义标记的真正价值所在。换句话说，现在有了独立的元素，这些元素中的标题级别可以重新定义。

文档中可能会包含一个分区，这个分区中可能会嵌套另一个分区，或者一篇文章，然后文章再嵌套分区，分区再嵌套文章、嵌套分区，文章再嵌套文章。而且每个分区和文章都可以拥有自己的 H1~H6。从这个意义上讲，H 元素真可谓层层嵌套。但是，在编写内容或者内容管理系统的时候，它们又都是完全独立的内容块。

这个点子并不是 HTML5 工作组想出来的，也不是 W3C 最近才提出的，实际上蒂姆·伯纳斯·李在 1991 年的一封邮件中就提到这种想法。他在邮件中解释了对 HTML 的理解，他说“我认为 H1、H2 这样单调地排下去不好，我希望它成为一种可以嵌套的元素，或者说一个通用的 H 元素，我们可以在其中嵌套不同的层次。”但后来，我们没有看到通用的 H 元素，而是一直在使用 H1 和 H2，那是因为我们一直在支持已有的内容。今天，这个理想终于实现了。

☑ 平稳退化

渐进增强的另一面就是平稳退化。HTML5 遵循这条原理的例子，就是使用 type 属性增强表单。下面列出了可以为 type 属性指定的新值，有 number、search 和 range 等。

```
input type="number"
input type="search"
input type="range"
input type="email"
input type="date"
input type="url"
```

最关键的问题在于浏览器在看到这些新 type 值时会如何处理。现有的浏览器是无法理解这些新 type 值的，但它们会将 type 的值解释为 text。

无论你写的是 input type="foo"，还是 input type="bar"，现有的任何浏览器都会把它理解为"text"。因而，从现在开始就可以使用这些新值，而且也可以放心，那些不理解它们的浏览器会把新值看成 type="text"，而这正是一个浏览器实践平稳退化原理的好例子。

例如，现在输入了 type="number"。假设需要一个输入数值的文本框，那么可以把这个 input 的 type 属性设置为 number，然后理解它的浏览器就会呈现一个可爱的小控件，如带小箭头图标的微调控件之类的，而在不理解它的浏览器中，则会看到一个文本框，那么为什么不能说输入 type="number"就会得到一个带小箭头图标的微调控件呢？

当然，还可以设置最小值和最大值属性，它们同样可以平稳退化。

再看 input type="search"。你也可以考虑一下这种输入框，因为这种输入框在 Safari 中会被呈现为一个系统级的搜索控件，右边还有一个单击即可清除搜索关键词的 X。而在其他浏览器中，得到的则是一个文本框，就像所写的是 input type="text"一样。那为什么还不使用 input type="search"呢？它不会有什么副作用。

HTML5 还为输入元素增加了新的属性，如 placeholder（占位符），它表示在文本框中预先放一些文本。占位符就是文本框可以接受的示例内容，一般颜色是灰色的。只要一单击文本框，它就消失了。如果把已经输入的内容全部删除，然后单击文本框外部，它又会出现。使用 JavaScript 编写一些代码

当然也可以实现这个功能，但 HTML5 只用一个 placeholder 属性就解决了问题。

当然，对于不支持这个属性的浏览器，还是可以使用 JavaScript 来实现占位符功能的。通过 JavaScript 来测试浏览器是否支持该属性也非常简单。如果不支持，可以让 JavaScript 来模拟这个功能。

下面就来看看这个新的 video 元素，它设计得既简单又实用。一个开始的 video 元素，加一个结束的 video 元素，中间可以放后备内容。注意，是后备内容，而不是保证可访问性的内容。下面就是针对不支持 video 元素的浏览器写的代码。

Note

```
<video src="movie.mp4">
    <!-- 后备内容 -->
</video>
```

那么，在后备内容里面放些什么东西呢？可以放 Flash 影片，这样，HTML5 的视频与 Flash 的视频就可以协同起来了，用户不需要作出选择。

```
<video src="movie.mp4">
    <object data="movie.swf">
        <!-- 后备内容 -->
    </object>
</video>
```

因为这里使用了 H264，部分浏览器支持这种视频格式，但有的浏览器不支持，对于不支持的情况，仅需要把后备内容放在那儿而已，后备内容可以包含多种视频格式。如果愿意的话，可以使用 source 元素而非 src 属性来指定不同的视频格式。

```
<video>
    <source src="movie.mp4">
    <source src="movie.ogv">
    <object data="movie.swf">
        <a href="movie.mp4">download</a>
    </object>
</video>
```

上面的代码中包含了 4 个不同的层次。

☑ 如果浏览器支持 video 元素，也支持 H264，则直接用第一个视频。

☑ 如果浏览器支持 video 元素，也支持 Ogg，那么用第二个视频。

☑ 如果浏览器不支持 video 元素，就要试试 Flash 影片了。

☑ 如果浏览器不支持 video 元素，也不支持 Flash，代码中还给出了下载链接。

总之，无论是 HTML5 和 Flash 一个也不能少。如果只使用 video 元素提供视频，难免会遇到问题。而如果只提供 Flash 影片，性质是一样的。所以还是应该两者兼顾。

为什么要兼顾这两种技术呢？假设你需要面向某些不支持 Flash 的手持设备提供视频，当然希望手持设备的用户能够看到视频了。

至于为什么要使用不同的格式？当然，面对竞争性的视频格式和不同的编码方式，只向浏览器发送用一种方式编码的视频是行不通的，而这也正是 Flash 在视频和音频领域如此成功的原因。只要把 Flash 影片发送给浏览器，安装了插件的浏览器都能正常播放。

☑ 最终用户优先

这个设计原理本质上是一种解决冲突的机制。换句话说，当面临一个要解决的问题时，如果 W3C 和 WHATWG 给出了不同的解决方案，一旦遇到冲突，最终用户优先，然后是作者，其次是实现者，

再次是标准制定者，最后才是理论上的完满。

理论上的完满，大致是指尽可能创建出最完美的格式。标准制定者指的是 WHATWG、W3C 等。实现者指的是浏览器厂商。作者就是开发人员。用户是第一位的。

Note

根据最终用户优先的原理，开发人员在链条中的位置高于实现者，假如我们发现了规范中的某些地方有问题，又不支持实现这个特性，那么就等于把相应的特性给否定了，规范里就得删除，因为我们的声音具有更高的权重。本质上是我们拥有了更大的发言权，我认为开发人员就应该拥有更多的发言权。

这应该是最重要的一条设计原理了，因为它承认了用户的权利，无论是设计一种格式，还是设计软件，这条原理保证了用户的发言权。而这条原理也正道出了事物运行的本质，用户的权利大于作者，作者的权利大于实现者，实现者的权利大于标准制定者。然而，反观其他规范，如 XHTML2，就会发现完全相反的做法——把追求理论的完满放在第一位，而把用户放在了链条的最底端，需要忍受严格错误处理带来的各种麻烦。这是一种完全不同的思维方式。

因此，无论是构建像 HTML5 这样的格式，还是构建一个网站，亦或一个内容管理系统，明确设计原理都至关重要。

1.3 编写第一个 HTML5 页面

尽管各主流厂商的最新版浏览器都对 HTML5 提供了很好的支持，但毕竟 HTML5 是一种全新的 HTML 标记语言，许多新的功能必须在搭建好相应的浏览环境后才可以正常浏览。为此，我们在正式执行一个 HTML5 页面之前，必须先搭建支持 HTML5 的浏览器环境，并检查浏览器是否支持 HTML5 标记。

1.3.1 搭建上机练习环境

目前，Microsoft 的 IE 系列（IE9+）浏览器，以及 Mozlilla 的 Firefox 与 Google 的 Chrome 浏览器等都可以很好地支持 HTML5。

本书所有的应用示例，主要执行的浏览器为 Chrome，其对应的版本号是 15.0.874.120。如需运行本书中的示例，则要安装最新的 Chrome 浏览器，以浏览相应的示例页面效果。

1.3.2 检测浏览器是否支持

安装相应的浏览器以后，为了能进一步了解浏览器支持 HTML 新标签功能的情况，还可以在引入新标签前，通过编写 JavaScript 代码来检测浏览器是否支持该标签。

浏览器在加载 Web 页面时会构造一个文本对象模型（DOM），然后通过该对象模型来表示页面中的各个 HTML 元素，这些元素被表示为不同的 DOM 对象。全部的 DOM 对象都共享一些公共或特殊的属性，如 HTML5 的某些特性，如果在支持该属性的浏览器中打开页面，就可以很快检测出这些 DOM 对象是否支持这些特性。

下面以加入画布标记为例，简单地说明检测浏览器的整个过程。在 HTML 页面中插入一段 HTML5 画布标记，当浏览器支持该标记时，将出现一个矩形；反之，则在页面中显示“该浏览器不支持 HTML5 的画布标记！”的提示。

【示例】在 Dreamweaver 中新建一个 HTML 页面，保存为 index.html，代码如下所示。

Note

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
<title></title>
<style type="text/css">
#myCanvas {
    background:red;
    width:200px;
    height:100px;
}
</style>
</head>
<body>
<canvas id="myCanvas">该浏览器不支持 HTML5 的画布标记！ </canvas>
</body>
</html>
```

将页面文件 index.html 在 IE8 浏览器中执行。由于 IE 8 浏览器暂不支持 HTML5 的画布标记，因此将显示如图 1.1 所示的页面效果。但是在 Chrome 浏览器中执行，由于该浏览器支持 HTML5 的画布标记，因此将显示如图 1.2 所示的页面效果。

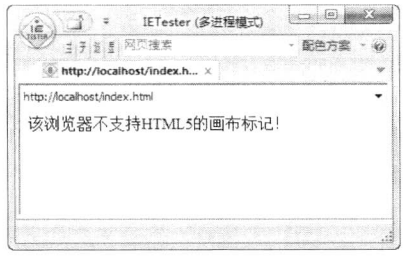


图 1.1　IE8 不支持 HTML5 的画布标记

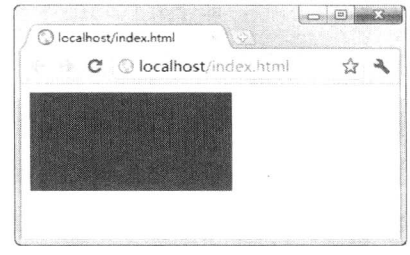


图 1.2　Chrome 支持 HTML5 的画布标记

虽然是同样一个页面，但由于不同的浏览器对 HTML5 标记的支持情况不同，其显示的页面效果也各异。因此，在编写 HTML5 新标记时，有必要先检测浏览器是否支持该标记。

接下来将利用 HTML5 结构编写一个简单的程序，从中体会 HTML5 的代码简洁所在。

1.3.3　使用 HTML5 编写简单的 Web 页面

HTML5 中不仅增加了很多新的页面标记，而且与 HTML 4 相比，整体页面的结构也发生了根本的变化。下面使用 HTML5 新结构来编写一个简单的页面，实现在页面中输出“Hello,World”的字样。

【示例】在 Dreamweaver 中新建一个 HTML 页面保存为 index.html，加入代码如下所示。

```
<!DOCTYPE html>
<META charset="utf-8">
<TITLE>第一个 HTML5 页面</TITLE>
<P>Hello,World</P>
```

该页面在 Chrome 浏览器中预览效果如图 1.3 所示。

图 1.3　第一个 HTML5 页面效果

通过短短几行代码就完成了一个页面的开发，这充分说明了 HTML5 语法的简洁，同时，HTML5 不是一种 XML 语言，其语法也很随意，下面从这两方面进行逐句分析。

第一行代码如下：

```
<!DOCTYPE html>
```

短短几个字符，甚至不包括版本号，就能告诉浏览器需要一个 doctype 来触发标准模式，可谓简明扼要。接下来需要说明文档的字符编码，否则浏览器将不能正确解析，导致安全隐患，为此加入如下一行代码：

```
<META charset="utf-8">
```

同样也是几个字符，便说明了该文档的字符编码。同时，HTML5 不区分字母大小写、标记结束符及属性是否加引号，即下列代码是等效的。

```
<meta charset="utf-8">
<META charset="utf-8" />
 <META charset=utf-8>
```

在主体中，可以省略主体标记，直接编写需要显示的内容，代码如下：

```
<P>Hello,World</P>
```

虽然在编写代码时省略了<html>、<head>和<body>标记，但在浏览器进行解析时，将会自动进行添加，如图 1.4 所示。

图 1.4　自动添加<html>、<head>与<body>标记后的源文件

考虑到代码的可维护性，在编写代码时，应该尽量增加这些在 HTML5 中可选的元素，从而最大限度地实现页面代码的简洁与完整。

1.4 HTML5 页面的特征

Note

上节简单介绍了一个 HTML5 页面的创建过程，而实际上，HTML5 页面的特征远不止这些，下面通过一个较为完整的页面来介绍 HTML5 的页面特征。

1.4.1 使用 HTML5 结构化元素

通过研究 Web 页面发现，如果使用一些带有语义性的标记，可以加快浏览器解释页面中元素的速度，如早期的<samp>、<var>元素；HTML5 继承了这些元素，并根据用户使用最为频繁的类名称和 ID 号不断开发新的标记，因为这些标记能真正体现开发者真实意图所在。下面通过示例说明 HTML5 是如何使用这些全新的 HTML5 特征来结构化元素的。

【示例】本示例将页面分成上、中、下 3 部分。上部用于显示导航；中部又分成两个部分，左边设置菜单，右边显示文本内容；下部显示页面版权信息。演示效果如图 1.5 所示。

```
<!DOCTYPE html>
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title></title>
<style type="text/css">
#header, #siderLeft, #siderRight,#footer{
    border:solid 1px #666;
    padding:10px;
    margin:6px;
}
#header { width:500px }
#siderLeft {
    float:left;
    width:60px;
    height:100px
}
#siderRight {
    float:left;
    width:406px;
    height:100px
}
#footer {
    clear:both;
    width:500px
}
</style>
</head>
<body>
<div id="header">导航</div>
<div id="siderLeft">菜单</div>
<div id="siderRight">内容</div>
```

```
<div id="footer">底部说明</div>
</body>
</html>
```

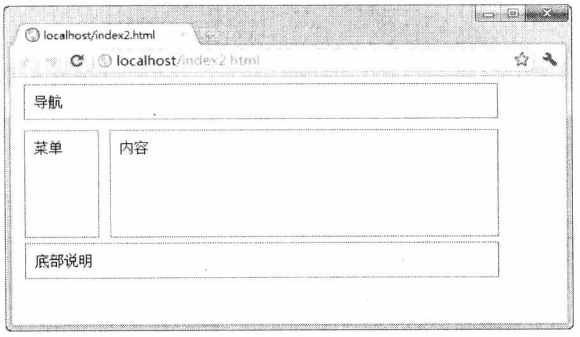


图 1.5　简单的网页布局

尽管上述代码不存在任何错误，还可以在 HTML5 环境中很好地运行，但该页面结构的很多部分对于浏览器来说都是未知的，这是因为浏览器是通过 ID 号定位元素的。因此，只要开发者不同，那么就允许元素的 ID 号各异，这会造成浏览器不能很好地表明元素在页面中的位置，必然影响页面解析的速度。幸好 HTML5 中新增的元素可以很快地定位某个标记，明确地表示页面中的位置。将上述代码修改成 HTML5 支持的页面代码，如下所示。演示效果如图 1.6 所示。

```
<!DOCTYPE html>
<html>
<head>
<meta charset=utf-8>
<title></title>
<style type="text/css">
header, nav, article,footer{
      border:solid 1px #666;
      padding:10px;
      margin:6px;
}
header { width:500px }
nav {
      float:left;
      width:60px;
      height:100px
}
article {
      float:left;
      width:406px;
      height:100px
}
footer {
      clear:both;
      width:500px
}
</style>
</head>
```

```
<body>
<header>导航</header>
<nav>菜单</nav>
<article>内容</article>
<footer>底部说明</footer>
</body>
</html>
```

图 1.6 简单的网页布局

虽然两段代码不一样，但在 Chrome 浏览器下实现的页面效果相同。从上述两段代码来看，使用 HTML5 新增元素创建的页面代码更加简单和高效。

可以看出，使用 、 、 和 这些标记元素没有任何实现的意义，即浏览器不能根据标记的 ID 号属性来推断这个标记的真正含义，因为 ID 号是可以变化的，不利于寻找。

而 HTML5 中的新增元素<header>可以明确地告诉浏览器此处是页头，<nav>标记用于构建页面的导航，<article>标记用于构建页面内容的一部分，<footer>元素表明页面已到页脚或根元素部分，并且这些标记都可以重复使用，极大地提高了开发者的工作效率。

此外，有些新增的 HTML5 元素还可以单独成为一个区域，如下列代码所示。

```
<header>
    <article>
        <h1>内容 1</h1>
    </article>
</header>
<header>
    <article>
        <h2>内容 2</h2>
    </article>
</header>
```

在 HTML5 中，一个<article>可以创建一个新的节点，并且每个节点都可以有自己的单独元素，如<h1>或<h2>，这样不仅使内容区域各自分段、便于维护，而且代码简单，局部修改方便。

1.4.2 使用 CSS 美化 HTML5 文档

在支持 HTML5 新元素的浏览器中，样式化各新增元素变得十分简单，可以对任意一个元素应用 CSS，包括直接设置或引入 CSS 文件。需要说明的是，在默认情况下，CSS 会默认元素的 display 属

性值为 inline，因此，为了正确地显示设置的页面效果，需要将元素的 display 属性设置为 block。下面通过一个简单示例来说明这一点。

【示例】在页面中设置相关样式，显示一段文章的内容。该页面在 Chrome 浏览器下预览效果如图 1.7 所示。

Note

```
<!DOCTYPE html >
<head>
<meta charset="utf-8" />
<style type="text/css">
article { display:block }
article header p { font-size:13px }
article header h1 { font-size:16px }
.p-date { font-size:11px }
</style>
</head>
<body>
<article>
    <header>
        <h1><a>谷歌推出多项新搜索功能 避谈 Google+</a></h1>
        <p class="p-date">日期：2012-08-11</p>
        <p>网易科技讯 8 月 11 日消息，据国外媒体报道，谷歌在主打社交网络服务 Google+一年后终于重归“老本行”，推出一系列和社交网络没有关联的搜索引擎新功能。</p>
    </header>
</article>
</body>
```

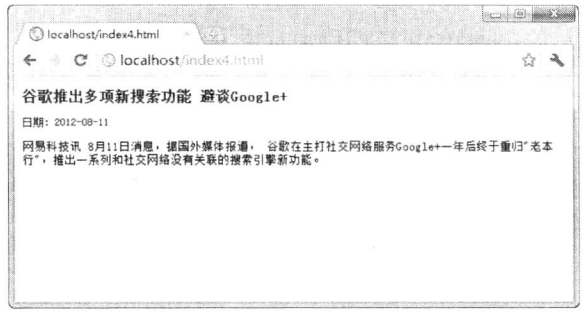


图 1.7　使用 CSS 美化 HTML5 页面

由于有些浏览器并不支持 HTML5 中的新增元素，如 IE8 或更早版本，其 CSS 只应用 IE 支持的那些元素。因此，为了能使新增的 HTML5 元素应用样式，可以在头部标记<head>中加入如下 JavaScript 代码，这样就可以应用样式了。

```
<script type="text/javascript">
document.createElement('article');
document.createElement('header');
</script>
```

考虑到各浏览器的兼容性不一样，可以对上述的 JavaScript 代码进行优化，即使用条件语句包含该 JavaScript 代码，使浏览器只在不支持 HTML5 的情况下才执行这段脚本。

从 HTML、XHTML 到 HTML5

（视频讲解：2 小时）

1993 年，HTML 首次以草案的形式发布，20 世纪 90 年代是 HTML 发展速度最快的时期，直到 1999 年的 4.01 版。在这个过程中，W3C 主要负责 HTML 规范的制定。HTML 4.01 发布之后，业界普遍认为 HTML 已经到了穷途末路，对 Web 标准的焦点也开始转移到了 XML 和 XHTML 上，HTML 被放在了次要位置。HTML5 以 HTML 为基础，对 HTML 进行了大量的修改。本章将从总体上介绍 HTML5 与 HTML 前面版本的不同，以及 HTML5 与 HTML4 之间比较大的区别。

2.1 HTML 基础

Note

HTML 是目前在网络上应用最为广泛的语言，也是构成网页文档的主要语言。HTML 文档是由 HTML 标签组成的描述性文本，HTML 标签可以标识文字、图形、动画、声音、表格和链接等。

2.1.1 HTML 简介

1982 年，美国人蒂姆·伯纳斯·李为了方便世界各地的物理学家能够进行合作研究以及信息共享，创造了 HTML 语言。1990 年他又发明了世界上第一个浏览器。1991 年 3 月，他把这个发明介绍给了一起在 CERN 公司工作的朋友，当时网页浏览器被 CERN 公司在世界各地的成员用来浏览 CERN 庞大的电话薄。1993 年，NCSA 推出了 Mosaic 浏览器并迅速爆红，成为世界上第一个广泛应用的浏览器，推动着互联网迅猛发展。在随后的 5 年里，Netscape 和 Microsoft 两个软件巨头掀起了一场互联网浏览器大战。这场战争最后以 Microsoft 的 Internet Explorer 完胜告终，但它极大地推动了互联网的发展，把网络带到了千千万万普通用户面前。从 1993 年互联网工程工作小组（IETF）工作草案发布，到 1999 年 W3C 发布 HTML 4.01 标准，HTML 共经历过 5 个版本。如今的 HTML 不仅成为 Web 上最主要的文档格式，而且在个人及商业应用中都发挥着它的作用。

HTML 是 Hypertext Markup Language 的缩写，中文翻译为超文本标识语言。使用 HTML 标签编写的文档称为 HTML 文档，目前最新版本是 HTML 5.0，使用最广泛的是 HTML 4.01 版本。

早期版本的 HTML 语言不适合构建标准化网页，因为它把结构和表现混淆在一起，例如，HTML 把不同类型的元素，如描述性元素 color、i 等和结构性元素 div、table 等，以及元素属性放在一起，为以后的维护和管理埋下隐患。

XHTML 是 The Extensible HyperText Markup Language 的缩写，中文翻译为可扩展标识语言，实际上它是 HTML 语言的升级版本，目前遵循的是 W3C 于 2000 年 1 月推荐的 XHTML 1.0 标准（参考 http://www.w1.org/TR/xhtml1）。XHTML 和 HTML 在语法和标签使用方面差别不大。熟悉 HTML 语言，再稍加熟悉标准结构和规范，也就熟悉了 XHTML 语言。XHTML 具有如下特点：

- ☑ 用户可以扩展元素，从而扩展功能，但在目前 1.0 版本下，用户只能够使用固定的预定义元素，这些元素基本上与 HTML 4.01 版本元素相同，但删除了部分属性描述性的元素。
- ☑ 能够与 HTML 很好地沟通，可以兼容当前不同的网页浏览器，实现 XHTML 页面的正确浏览。

2.1.2 HTML 特性

HTML 作为一种网页内容标识语言，易学易懂，熟悉使用该语言可以制作功能强大、美观大方的网页。HTML 语言的主要作用说明如下。

- ☑ 使用 HTMl 语言标识文本。例如，定义标题文本、段落文本、列表文本和预定义文本等。
- ☑ 使用 HTML 语言建立超链接，通过超链接可以访问互联网上的所有信息，当使用鼠标单击超链接时，会自动跳转到链接页面。
- ☑ 使用 HTML 语言创建列表，把信息有序组织在一起，以方便浏览。
- ☑ 使用 HTMl 语言在网页中显示图像、声音、视频、动画等多媒体信息，把网页设计得更富冲击力。

- ☑ 使用 HTML 语言可以制作表格，以便显示大量数据。
- ☑ 使用 HTML 语言制作表单，允许在网页内输入文本信息，执行其他用户操作，方便信息互动。

Note

2.1.3 HTML 结构

HTML 文档一般都应包含两部分：头部区域和主体区域。HTML 文档基本结构由 3 个标签负责组织：<html>、<head>和<body>。其中<html>标签标识 HTML 文档，<head>标签标识头部区域，而<body>标签标识主体区域。一个完整的 HTML 文档基本结构如下所示。

```
<html> <!--语法开始-->
    <head>
        <!--头部信息，如<title>标签定义的网页标题-->
    </head>
    <body>
        <!--主体信息，包含网页显示的内容-->
    </body>
</html> <!--语法结束-->
```

可以看到，每个标签都是成对组成的，第一个标签（如<html>）表示标识的开始位置，而第二个标签（如</html>）表示标识的结束位置。<html>标签中包含<head>和<body>标签，而<head>和<body>标签则是并列排列的。

如果把上面的字符代码放置在文本文件中，然后另存为“test.html”，就可以在浏览器中浏览了。当然，由于这个简单的 HTML 文档还没有包含任何可显示信息，所以在浏览器中是看不到任何内容的。

2.1.4 HTML 语法

编写 HTML 文档时，必须遵循 HTML 语法规范。HTML 文档实际上就是一个文本文件，它由标签和信息混合组成，当然这些标签和信息必须遵循一定的组合规则，否则浏览器是无法解析的。

HTML 语言的规范条文不多，相信读者也很容易理解。从逻辑上分析，这些标签包含的内容就表示一类对象，也可以称为网页元素。从形式上分析，这些网页元素通过标签进行分隔，然后表达一定的语义。很多时候，我们把网页标签和网页元素混为一谈，而实际上，网页文档就是由元素和标签组成的容器。

- ☑ 所有标签都包含在“<”和“>”起止标识符中，构成一个标签如<style>、<head>、<body>和 等。
- ☑ 在 HTML 文档中，绝大多数元素都有起始标签和结束标签，在起始标签和结束标签之间包含的是元素主体，例如，<body>和</body>中间包含的就是网页内容主体。
- ☑ 起始标签包含元素的名称以及可选属性，也就是说元素的名称和属性都必须包含在起始标签中。结束标签以反斜杠开始，然后附加上元素名称。例如：

```
<tag>元素主体</tag>
```

- ☑ 元素的属性包含属性名称和属性值两部分，中间通过等号进行连接，多个属性之间通过空格进行分隔。属性与元素名称之间也通过空格进行分隔。例如：

```
<tag a1="v1" a2="v2" a3="v3" … an="vn">元素主体</tag>
```

☑ 少数元素的属性也可能不包含属性值，仅包含一个属性名称。例如：

```
<tag a1 a2 a3 … an>元素主体</tag>
```

Note

☑ 一般属性值应该包含在引号内，虽然不加引号浏览器也能够解析，但是读者应该养成良好的习惯。

☑ 属性是可选的，元素包含多少个属性也是不确定的，这主要根据不同元素而定。不同的元素会包含不同的属性。HTML 也为所有元素定义了公共属性，如 title、id、class 和 style 等。

虽然大部分标签都是成对出现，但是也有少数标签不是成对的，这些孤立的标签被称为空标签。空标签仅包含起始标签，没有结束标签。例如：

```
<tag>
```

同样，空标签也可以包含很多属性，用来标识特殊效果或者功能，例如：

```
<tag a1="v1" a2="v2" … an="vn">
```

☑ 标签可以相互嵌套，形成文档结构。嵌套必须匹配，不能交错嵌套，例如， 就是不合法的。合法的嵌套应该是包含或被包含的关系，例如， 或 。

☑ HTML 文档所有信息必须包含在<html>标签中，所有文档的元信息应包含在<head>子标签中，而 HTML 传递信息和网页显示内容应包含在<body>子标签中。

对于 HTML 文档来说，除了必须符合基本语法规范外，还必须保证文档结构信息的完整性。完整文档结构如下所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w1.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w1.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>文档标题</title>
</head>
<body></body>
</html>
```

HTML 文档应主要包括如下内容：

☑ 必须在首行定义文档的类型，过渡型文档可省略。

☑ <html>标签应该设置文档名字空间，过渡型文档可省略。

☑ 必须定义文档的字符编码，一般使用<meta>标签在头部定义，常用字符编码包括中文简体（gb2312）、中文繁体（big5）和通用字符编码（utf-8）。

☑ 应该设置文档的标题，可以使用<title>标签在头部定义。

HTML 文档扩展名为.htm 或.html，保存时必须正确使用扩展名，否则浏览器无法正确地解析。如果要在 HTML 文档中增加注释性文本，则可以添加在“<!--”和“-->”标识符之间，例如：

```
<!--单行注释-->
```

或

```
<!----------------
多行注释
---------------->
```

2.1.5　HTML 标签

HTML 定义的标签很多，下面对常用标签进行说明，随着读者学习不断深入，相信会完全掌握 HTML 所有标签的用法和使用技巧。

Note

1. 文档结构标签

此类标签主要用来标识文档的基本结构，主要包括以下几种。

☑ <html>...</html>：标识 HTML 文档的起始和终止。

☑ <head>...</head>：标识 HTML 文档的头部区域。

☑ <body>...</body>：标识 HTML 文档的主体区域。

【示例 1】设计最简单的网页文档。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>无标题文档</title>
</head>
<body> 网页正文写在这里……
</body>
</html>
```

2. 文本格式标签

这些标签主要用来标识文本区块，并附带一定的显示格式，主要标签说明如下。

☑ <title>...</title>：标识网页标题。

☑ <hi>...</hi>：标识标题文本，其中 i 表示 1、2、3、4、5、6，分别表示一级、二级、三级等标题。

☑ ... ：标识段落文本。

☑ <pre>...</pre>：标识预定义文本。

☑ <blockquote>...</blockquote>：标识引用文本。

【示例 2】分别使用<h1>和 标签标识网页标题和段落文本。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<h1>文本格式标签</h1>
<p>&lt;p&gt;标签标识段落文本</p>
</body>
</html>
```

3. 字符格式标签

字符格式标签主要用来标识部分文本字符的语义，很多字符格式标签可以呈现一定的显示效果。例如，加粗显示、斜体显示或者下划线显示等。主要标签说明如下。

☑ ...：标识强调文本，以加粗效果显示。

Note

☑ ...：标识引用文本，以斜体效果显示。

☑ <blink>...</blink>：标识闪烁文本，以闪烁效果显示（IE 浏览器不支持该标签）。

☑ <big>...</big>：标识放大文本，以放大效果显示。

☑ <small>...</small>：标识缩小文本，以缩小效果显示。

☑ ^{...}：标识上标文本，以上标效果显示。

☑ _{...}：标识下标文本，以下标效果显示。

☑ <cite>...</cite>：标识引用文本，以引用效果显示。

【示例 3】分别使用各种字符格式标签显示一个数学方程式的解法。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<p>例如，针对下面这个一元二次方程：</p>
<p><i>x</i><sup>2</sup>-<b>5</b><i>x</i>+<b>4</b>=0</p>
<p>我们使用<big><b>分解因式法</b></big>来演示解题思路如下：</p>
<p><small>由：</small>(<i>x</i>-1)(<i>x</i>-4)=0</p>
<p><small>得：</small><br /><i>x</i><sub>1</sub>=1<br />
    <i>x</i><sub>2</sub>=4</p>
</body>
</html>
```

4. 列表标签

在 HTML 文档中，列表结构可以分为两种类型：有序列表和无序列表。无序列表使用项目符号来标识列表，而有序列表则使用编号来标识列表的项目顺序。具体使用标签说明如下。

☑ ...：标识无序列表。

☑ ...：标识有序列表。

☑ ...：标识列表项目。

【示例 4】使用无序列表分别显示一元二次方程求解的 4 种方法。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<h1>解一元二次方程</h1>
<p>一元二次方程求解有 4 种方法：</p>
<ul>
    <li>直接开平方法 </li>
    <li>配方法 </li>
    <li>公式法 </li>
    <li>分解因式法</li>
</ul>
</body>
</html>
```

另外，还可以定义定义列表。定义列表是一种特殊的结构，它包括词条和解释两块内容。包含的标签说明如下。

☑ <dl>...</dl>：标识定义列表。

☑ <dt>...</dt>：标识词条。

☑ <dd>...</dd>：标识解释。

Note

【示例 5】使用定义列表显示两个成语的解释。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<h1>成语词条列表</h1>
<dl>
    <dt>知无不言，言无不尽</dt>
    <dd>知道的就说，要说就毫无保留。</dd>
    <dt>智者千虑，必有一失</dt>
    <dd>不管多聪明的人，在很多次的考虑中，也一定会出现个别错误。</dd>
</dl>
</body>
</html>
```

5. 链接标签

链接标签可以实现把多个网页联系在一起。主要结构如下。

☑ <a>...：标识超链接。

【示例 6】使用<a>标签定义一个超链接，单击该超链接可以跳转到百度首页。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<a href="http://www.baidu.com/">去百度搜索</a>
</body>
</html>
```

<a>标签还可以定义锚点。锚点是一类特殊的超链接，它可以定位到网页中某个具体的位置。例如，在下面示例中单击超链接文本，就可以跳转到网页的底部。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<a href="#btm">跳转到底部</a>
<div id="box" style="height:2000px; border:solid 1px red;">撑开浏览器滚动条</div>
<span id="btm">底部锚点位置</span>
```

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

6. 多媒体标签

多媒体标签主要用于引入外部多媒体文件并进行显示。多媒体标签主要包括以下几种。

☑ <img />：嵌入图像。

☑ <embed>...</embed>：嵌入多媒体。

☑ <object>...</object>：嵌入多媒体。

7. 表格标签

表格标签用来组织和管理数据。主要包括下面几种标签。

☑ <table>...</table>：定义表格结构。

☑ <caption>...</caption>：定义表格标题。

☑ <th>...</th>：定义表头。

☑ <tr>...</tr>：定义表格行。

☑ <td...</td>：定义表格单元格。

【示例 7】使用表格结构显示 5 行 3 列的数据集。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<table summary="ASCII 是英文 American Standard Code for Information Interchange 的缩写。ASCII 编码是目前计算机最通用的编码标准。因为计算机只能接受数字信息，ASCII 编码将字符转换为数字来表示，以便计算机能够接受和处理。">
    <caption>ASCII 字符集（节选）</caption>
    <tr>
        <th>十进制</th>
        <th>十六进制</th>
        <th>字符</th>
    </tr>
    <tr>
        <td>9</td>
        <td>9</td>
        <td>TAB（制表符）</td>
    </tr>
    <tr>
        <td>10</td>
        <td>A</td>
        <td>换行</td>
    </tr>
    <tr>
        <td>13</td>
        <td>D</td>
        <td>回车</td>
    </tr>
    <tr>
```

```
            <td>32</td>
            <td>20</td>
            <td>空格</td>
        </tr>
</table>
</body>
</html>
```

Note

8. 表单标签

表单标签主要用来制作交互式表单。主要包括下面几种标签。

☑ <form>...</form>：定义表单结构。

☑ <input />：定义文本域、按钮和复选。

☑ <textarea>...</textarea>：定义多行文本框。

☑ <select>...</select>：定义下拉列表。

☑ <option>...</option>：定义下拉列表中的选择项目。

【示例 8】分别定义单行文本框、多行文本框、复选框、单选按钮、下拉菜单和提交按钮的表单。

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>示例代码</title>
</head>
<body>
<form id="form1" name="form1" method="post" action="">
    <p>单行文本域：<input type="text" name="textfield" id="textfield" /></p>
    <p>密码域：<input type="password" name="passwordfield" id="passwordfield" /></p>
    <p>多行文本域：<textarea name="textareafield" id="textareafield"> </textarea></p>
    <p>复选框：复选框 1<input name="checkbox1" type="checkbox" value="" />
              复选框 2<input name="checkbox2" type="checkbox" value="" />
    </p>
    <p>单选按钮：
        <input name="radio1" type="radio" value="" />按钮 1
        <input name="radio2" type="radio" value="" />按钮 2
    </p>
    <p>下拉菜单：
        <select name="selectlist">
            <option value="1">选项 1</option>
            <option value="2">选项 2</option>
            <option value="3">选项 3</option>
        </select>
    </p>
    <p><input type="submit" name="button" id="button" value="提交" /></p>
</form>
</body>
</html>
```

2.1.6　HTML 属性

HTML 元素包含的属性众多，这里无法列出所有元素的全部属性，当然也没有个必要，仅就公共

属性进行分析。公共属性大致可分为基本属性、语言属性、键盘属性、内容属性和延伸属性等类型。

1. 基本属性

Note

基本属性主要包括以下 3 个，这 3 个基本属性大部分元素都拥有。

class	定义类规则或样式规则
id	定义元素的唯一标识
style	定义元素的样式声明

下面这些元素不拥有基本属性。

html、head	文档和头部基本结构
title	网页标题
base	网页基准信息
meta	网页元信息
param	元素参数信息
script、style	网页的脚本和样式

这些元素一般位于文档头部区域，用来标识网页元信息。

2. 语言属性

语言属性主要用来定义元素的语言类型，包括两个属性。

lang	定义元素的语言代码或编码
dir	定义文本的方向，包括 ltr 和 rtl 取值，分别表示从左向右和从右向左

下面这些元素不拥有语言语义属性。

frameset、frame、iframe	网页框架结构
br	换行标识
hr	结构装饰线
base	网页基准信息
param	元素参数信息
script	网页的脚本

【示例 1】分别为网页代码定义简体中文的语言，字符对齐方式为从左到右。第二行代码为 body 定义了美式英语。

```
<html xmlns="http://www.w3.org/1999/xhtml" dir="ltr" xml:lang="zh-CN">
<body id="myid" lang="en-us">
```

3. 键盘属性

键盘属性定义元素的键盘访问方法，包括两个属性。

accesskey	定义访问某元素的键盘快捷键
tabindex	定义元素的 Tab 键索引编号

使用 accesskey 属性可以使用快捷键（Alt+字母）访问指定 URL，但是浏览器不能很好地支持，在 IE 中仅激活超链接，需要配合 Enter 键确定，而在 Firefox 中没有反应。

【示例 2】在导航菜单中设置快捷键。

```
<a href="http://www.html5.cn/" accesskey="a">按住 Alt 键，再按 A 键可以链接到样吧首页</a>
```

tabindex 属性用来定义元素的 Tab 键访问顺序，可以使用 Tab 键遍历页面中的所有链接和表单元素。遍历时会按照 tabindex 的大小决定顺序，当遍历到某个链接时，按 Enter 键即可打开链接页面。例如：

Note

```
<a href="#" tabindex="1">Tab 1</a>
<a href="#" tabindex="3">Tab 3</a>
<a href="#" tabindex="2">Tab 2</a>
```

4. 内容属性

内容属性定义元素包含内容的附加信息，这些信息对于元素来说具有重要补充作用，避免元素本身包含信息不全而被误解，包括 5 个属性。

alt	定义元素的替换文本
title	定义元素的提示文本
longdesc	定义元素包含内容的大段描述信息
cite	定义元素包含内容的引用信息
datetime	定义元素包含内容的日期和时间

alt 和 title 是两个常用的属性，分别定义元素的替换文本和提示文本，但是很多设计师习惯于混用这两个属性，没有刻意去区分它们的语义性。实际上，除了 IE 浏览器，其他标准浏览器都不会支持它们的混用，但是由于 IE 浏览器的纵容，才导致了很多设计师误以为 alt 属性就是设置提示文本的。

```
<a href="URL" title="提示文本">超链接</a>
<img src="URL" alt="替换文本" title="提示文本" />
```

替换文本（Alternate Text）并不是用来做提示（Tool Tip）的，或者更加确切地说，它并不是为图像提供额外说明信息的。相反，title 属性才负责为元素提供额外说明信息。

当图像无法显示时，必须准备替换的文本来替换无法显示的图像，这对于图像和图像热点是必须的，因此 alt 属性只能用在 img、area 和 input 元素中（包括 applet 元素）。对于 input 元素，alt 属性用来替换提交按钮的图片。

```
<input type="image" src="URL" alt="替换文本" />.
```

为什么要设置替换文本呢？这主要是因为浏览器被禁止显示、不支持或无法下载图像时，通过替换文本为那些不能看到图像的浏览者提供文本说明，这是一个很重要的预防和补救措施。另外，还应该考虑到网页对于视觉障碍者，或者使用其他用户代理，如屏幕阅读器、打印机等代理设备的影响。当然，从语义角度考虑，替换文本应该提供图像的简明信息，并保证在上下文中有意义，而对于那些修饰性的图片可以使用空值（alt=""）。

title 属性为元素提供提示性的参考信息，这些信息是一些额外的说明，具有非本质性，因此该属性也不是一个必须设置的属性。当鼠标指针移到元素上面时，即可看到这些提示信息，但是 title 属性不能够用在下面的元素上。

html、head	文档和头部基本结构
title	网页标题
base、basefont	网页基准信息
meta	网页元信息
param	元素参数信息
script	网页的脚本和样式

相对而言，title 属性可以比 alt 属性设置更长的文本，不过有些浏览器可能会限制提示文本的长

度，但是不管怎么规定，提示文本一定要简明、扼要，并用在恰当的地方，而不是为所有元素都定义一个提示文本，那样就显得画蛇添足了。提示文本一般多用在超链接上，特别是对图标按钮必须提供提示性说明信息，否则用户就会不明白这些图标按钮的作用。

Note

如果要为元素定义更长的描述信息，则应该使用 longdesc 属性。longdesc 属性可以用来提供链接到一个包含图片描述信息的单独页面或者长段描述信息。其用法如下。

```
<img src="URL" alt="人物照" title="张三于 2010-5-1 中国馆留念" longdesc="这是张三于 2010 年 5 月 1 日在中国馆前的留影，当时天很热，穿着短裤，手里拿着矿泉水，到处都是云集于此的世博会开幕式观众，场面热闹非凡" />
```

或

```
<img src="UTL" alt="替换文本" longdesc="详细描述图像的网页.html" />
```

这种方法意味着从当前页面链接到另一个页面，由此可能会造成理解上的困难。另外浏览器对于 longdesc 属性的支持也不一致，因此应该避免使用。如果感觉对图片的长描述信息很有用，那么不妨考虑把这些信息简单地显示在同一个文档里，而不是链接到其他页面或者藏起来，这样能够保证每个人都可以阅读。

cite 一般用来定义引用信息的 URL。例如，下面一段文字引自 http://www.html5.cn/csslayout/ndex.htm，所以可以这样来设置：

```
<blockquote cite="http://www.html5.cn/csslayout/index.htm">
    <p>CSS 的精髓是布局，而不是样式，布局是需要缜密的结构分析和设计的</p>
</blockquote>
```

datetime 属性定义包含文本的时间，这个时间表示信息的发布时间，也可能是更新时间，例如：

```
<ins datetime="2010-5-1 8:0:0">2010 年上海世博会</ins>
```

2.2 XHTML 基础

XHTML 语言是在 HTML 语言基础上发展而来的，但是为了兼容数以万计的现存网页和不同浏览器，XHTML 文档与 HTML 文档没有太大区别，只是添加了 XML 语言的基本规范和要求。

2.2.1 XHTML 结构

完整的 XHTML 文档结构如下。

```
<!--[XHTML 文档基本框架]-->
<!--定义 XHTML 文档类型-->
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<!--XHTML 文档根元素，其中 xmlns 属性声明文档命名空间-->
<html xmlns="http://www.w3.org/1999/xhtml">
<!--头部信息结构元素-->
<head>
<!--设置文档字符编码-->
<meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
```

```
<!--设置文档标题-->
<title>无标题文档</title>
</head>
<!--主体内容结构元素-->
<body>
</body>
</html>
```

Note

XHTML 代码不排斥 HTML 规则，在结构上也基本相似，但如果仔细比较，它们有两点不同。

1. 定义文档类型

在 XHTML 文档第一行新增了<!DOCTYPE>元素，该元素用来定义文档类型。DOCTYPE 是 document type（文档类型）的简写，用于设置 XHTML 文档的版本。使用时应注意该元素的名称和属性必须大写。

DTD（如 xhtml1-transitional.dtd）表示文档类型定义，里面包含了文档的规则，网页浏览器会根据预定义的 DTD 来解析页面元素，并把这些元素所组织的页面显示出来。要建立符合网页标准的文档，DOCTYPE 声明是必不可少的关键组成部分，除非所编写的 XHTML 确定了一个正确的 DOCTYPE，否则页面内的元素和 CSS 不能正确生效。

2. 声明命名空间

在 XHTML 文档根元素中必须使用 xmlns 属性声明文档的命名空间。xmlns 是 XHTML NameSpace 的缩写，中文翻译为命名空间（也有人翻译为名字空间、名称空间）。命名空间是收集元素类型和属性名字的一个详细 DTD，它允许通过一个 URL 地址指向来识别命名空间。

XHTML 是 HTML 向 XML 过渡的标识语言，它需要符合 XML 规则，因此也需要定义命名空间。又因为 XHTML 1.0 还不允许用户自定义元素，因此它的命名空间都相同，就是 http://www.w3.org/1999/xhtml，这也是每个 XHTML 文档的 xmlns 值都相同的原因。

2.2.2 XHTML 语法

XHTML 是根据 XML 语法简化而来的，因此它遵循 XML 文档规范。同时 XHTML 又大量继承 HTML 语言语法规范，因此与 HTML 语言非常相似，不过它对代码的要求更加严谨。遵循这些要求，对于培养良好的 XHTML 代码书写习惯是非常重要的。

- ☑ 在文档的开头必须定义文档类型。
- ☑ 在根元素中应声明命名空间，即设置 xmlns 属性。
- ☑ 所有标签都必须是闭合的。在 HTML 中，用户可能习惯书写独立的标签，如 、
，而不习惯写对应的 和
来关闭它们，但在 XHTML 中这是不合法的。XHTML 要求有严谨的结构，所有标签都必须关闭。如果是单独不成对的标签，应在标签的最后加一个“/”来关闭它，如
。
- ☑ 所有元素和属性都必须小写。这与 HTML 不同，XHTML 对大小写是敏感的，<title>和<TITLE>表示不同的标签。
- ☑ 所有的属性必须用引号（""）括起来。在 HTML 中，可以不给属性值加引号，但是在 XHTML 中必须加引号，如<table height="80"></table>。特殊情况下，可以在属性值里使用双引号或单引号。
- ☑ 所有标签都必须合理嵌套。这是因为 XHTML 要求有严谨的结构，因此所有的嵌套都必须

按顺序，详细讲解请参阅最后一章。

☑ 所有属性都必须被赋值，没有值的属性就用自身来赋值。例如：

错误写法：

Note

```
<td nowrap>
```

正确写法：

```
<td nowrap="nowrap">
```

☑ 所有特殊符号都用编码表示，例如，小于号（<）不是元素的一部分，必须被编码为“<”；大于号（>）不是元素的一部分，必须被编码为“>”。

☑ 不要在注释内容中使用“--”。“--”只能出现在 XHTML 注释的开头和结束，也就是说在内容中它们不再有效。例如：

错误写法：

```
<!--注释---------注释-->
```

正确写法：

```
<!--注释————————注释-->
```

☑ XHTML 规范废除了 name 属性，而使用 id 属性作为统一的名称。在 IE 4.0 及以下版本中应保留 name 属性，使用时可以同时使用 id 和 name 属性。

上面列举的几点是 XHTML 最基本的语法要求，习惯于 HTML 的读者，应克服代码书写中的随意性，相信好的习惯会影响你的一生。

2.2.3 XHTML 类型

XHTML 1.0 支持 3 种 DTD（文档类型定义）声明：过渡型（Transitional）、严格型（Strict）和框架型（Frameset）。

1. 过渡型

这种文档类型对于标签和属性的语法要求不是很严格，允许在页面中使用 HTML 4.01 的标签（符合 XHTML 语法标准）。过渡型 DTD 语句如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w1.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

2. 严格型

这类文档类型对于文档内的代码要求比较严格，不允许使用任何表现层的标签和属性。严格型 DTD 语句如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w1.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

在严格型文档类型中，以下元素将不被支持：

```
center                    居中（属于表现层）
font                      字体样式，如大小、颜色和样式（属于表现层）
strike                    删除线（属于表现层）
s                         删除线（属于表现层）
```

Note

u	文本下划线（属于表现层）
iframe	嵌入式框架窗口（专用于框架文档类型或过渡型文档）
isindex	提示用户输入单行文本（与 input 元素语义重复）
dir	定义目录列表（与 dl 元素语义重复）
menu	定义菜单列表（与 ul 元素语义重复）
basefont	定义文档默认字体属性（属于表现层）
applet	定义插件（与 object 元素语义重复）

在严格型文档类型中，以下属性将不被支持：

align（支持 table 包含的相关元素：tr、td、th、col、colgroup、thead、tbody、tfoot）
language
background
bgcolor
border（table 元素支持）
height（img 和 object 元素支持）
hspace
name（在 HTML 4.01 Strict 中支持，在 XHTML 1.0 Strict 中的 form 和 img 元素不支持）
noshade
nowrap
target
text、link、vlink 和 alink
vspace
width（img、object、table、col 和 colgroup 元素支持）

3. 框架型

这是一种专门针对框架页面所使用的 DTD，当页面中含有框架元素时，就应该采用这种 DTD。框架型 DTD 语句如下。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w1.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

使用严格的 DTD 来制作页面，当然是最理想的方式，但是，对于没有深入了解 Web 标准的网页设计者来说，比较适合使用过渡型 DTD。因为过渡型 DTD 还允许使用表现层元素和属性，比较适合大多数网页制作人员使用。

对于大多数标准网页设计师来说，过渡型 DTD（XHTML 1.0 Transitional）是比较理想的选择。因为这种 DTD 允许使用描述性的元素和属性，也比较容易通过 W3C 的代码校验。

2.2.4　DTD 解析

在 XHTML 文档中，DOCTYPE 是一个必要元素，它决定了网页文档的显示规则。DOCTYPE 是 Document Type 的简写，中文翻译为文档类型。在网页中通过在首行代码中定义文档类型，来指定页面所使用的 HTML 的版本类型。在构建符合标准的网页时，只有使用正确的 DOCTYPE（文档类型），HTML 文档的结构和样式才能被正常解析和呈现。

实际上，DTD 是一套关于标签的语法规则。DTD 文件是一个 ASCII 的文本文件，后缀名为.dtd。利用 DOCTYPE 声明中的 URL 可以访问指定类型的 DTD 详细信息。例如，对于 XHTML 1.0 过渡型 DTD 的 URL 为 http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd，在 Dreamweaver 中打开 XHTML 1.0 过渡型 DTD 文档，代码显示如图 2.1 所示。

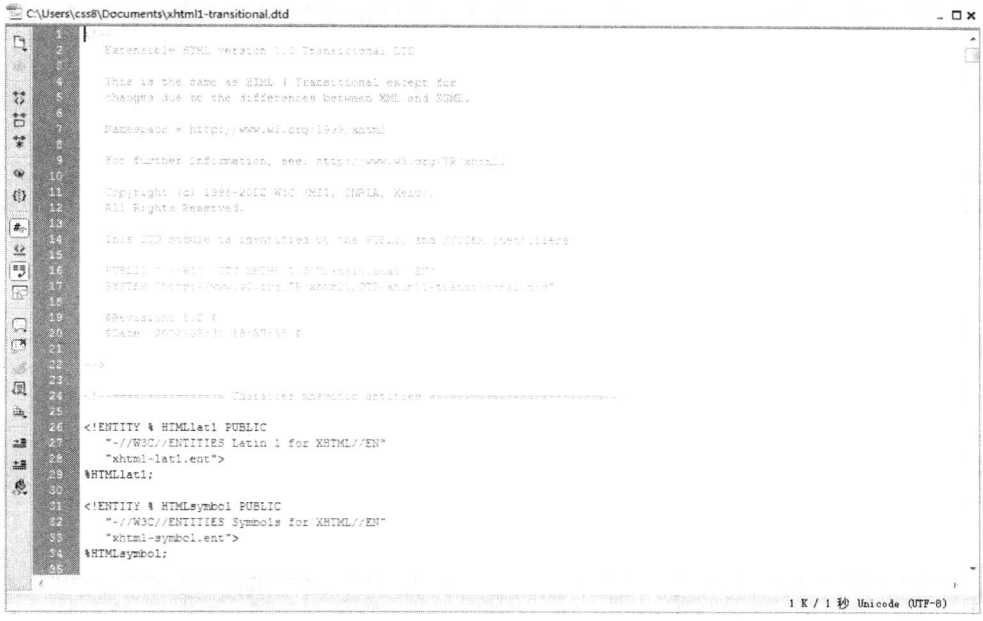


图 2.1　XHTML 1.0 过渡型 DTD 文档

一个 DTD 文档包含元素的定义规则，元素间关系的定义规则，元素可使用的属性、实体或符号规则。这些规则用于标识 Web 文档的内容。此外还包括了一些其他规则，它们规定了哪些标签能出现在其他标签中。文档类型不同，它们对应的 DTD 也不相同。

如果页面中没有显示声明 DOCTYPE，则不同浏览器就会自动采用各自默认的 DOCTYPE 规则来解析文档中的各种标签和 CSS 样式码。因此，从浏览器兼容性来考虑，声明 DOCTYPE 是必须的。DOCTYPE 声明必须放在（X）HTML 文档的顶部，在文档类型声明语句的上面不能够包含任何 HTML 代码，也不能包括 HTML 注释标签。DOCTYPE 声明语句的说明如图 2.2 所示。

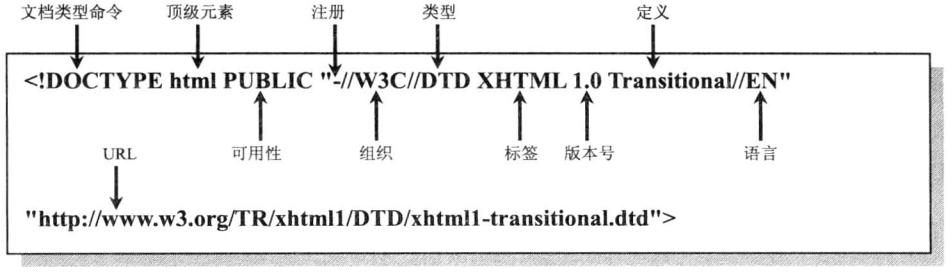

文档类型命令
顶级元素
注册
类型
定义
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
URL
可用性
组织
标签
版本号
语言
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

图 2.2　DOCTYPE 结构图

DOCTYPE 声明中各个部分说明如下。

- ☑ 顶级元素：指定 DTD 中声明的顶级元素类型，这与声明的 SGML 文档类型相对应。HTML 文档默认顶级元素为 html。
- ☑ 可用性：指定正式公开标识符（FPI）是可公开访问的对象（PUBLIC）还是系统资源（SYSTEM）。默认为 PUBLIC，SYSTEM 系统资源包括本地文件或 URL。
- ☑ 注册：指定组织是否由国际标准化组织（ISO）注册。“+”（默认）表示组织名称已注册，“-”表示组织名称未注册。W3C 是属于未注册 ISO 的组织，以显示为“-”符号。

Note

- ☑ 组织：指定在 DOCTYPE 声明引用的 DTD（文档类型定义）的创建和维护的团体或组织的名称。XHTML 语言规范的创建和维护组织为 W3C。
- ☑ 类型：指定公开文本的类，即所引用的对象类型。XHTML 默认为 DTD。
- ☑ 标签：指定公开文本的描述，即对所引用的公开文本的唯一描述性名称，后面可附带版本号。HTML 默认为 HTML，XHTML 默认为 XHTML，后面跟随的是语言版本号。
- ☑ 定义：指定文档类型定义，包含 Frameset（框架集文档）、Strict（严格型文档）和 Transitional（过渡型文档）。Strict（严格型文档）禁止使用 W3C 规范中指定将逐步淘汰的元素和属性，而 Transitional（过渡型文档）可以包含除 frameset 元素以外的全部内容。
- ☑ 语言：指定公开文本的语言，即用于创建所引用对象的自然语言编码系统。该语言定义已编写为 ISO 639 语言代码（两个字母要大写），默认为 EN（英语）。
- ☑ URL：指定所引用对象的位置。

从上面的结果分析中可以看到 DOCTYPE 声明语句的写法是严格遵循一定的规则的，只有这样，浏览器才能够调用对应文档类型的规则集来解释文档中的标签。所谓的文档类型规则集也就是 W3C 公开发布的一个文档类型定义（DTD）中包含的规则。

2.2.5　命名空间

在 XHTML 文档中，读者还需要注意另一个容易忽略的问题，即给<html>标签定义命名空间。例如：

```
<html xmlns="http://www.w1.org/1999/xhtml">
```

xmlns 是 html 元素的一个特殊属性。这个 xmlns 属性是 XHTML Name Space 的缩写，中文翻译为命名空间，该属性声明了 html 顶级元素的命名空间。那么命名空间在文档中是必须的吗？它有什么作用呢？

在标准设计中，命名空间是必须设置的一个属性，用来定义该顶级元素以及其包含的各级子元素的唯一性。命名空间声明允许用户通过一个网址指向来识别文档内标签的唯一性。

由于 XML 语言允许用户自定义标签，这样就可能存在所定义的标签与别人定义的标签名称发生冲突的情况。虽然标签名称不同，但是标签所表示的语义可能相同。当这些文档在网上自由传播或者相互交换文件时，由于名称相同可能会发生语义冲突。为此需要为各自的文档指定其语义的限制空间，于是 xmlns 属性就派上了用场。

【示例】分别定义张三和李四两个人的文档。

```
<!--张三：自定义文档 -->
<document>
    <name>书名</name>
    <author>作者</author>
    <content>目录</content>
</document>
<!--李四：自定义文档 -->
<document>
    <title>论文题目</title>
    <author>作者</author>
    <content>论文内容</content>
</document>
```

文档的根元素都是 document，同时文档中包含有很多相同的元素名。如果文档都在网上共享就会发生语义冲突。如果使用 xmlns 分别为它们定义一个命名空间，就不会发生冲突。例如：

Note

```
<!--张三：自定义文档 -->
<document xmlns="http://www.html5.cn/zhangsan">
    <name>书名</name>
    <author>作者</author>
    <content>目录</content>
</document>
<!--李四：自定义文档 -->
<document xmlns="http://www.html5.cn/lisi">
    <title>论文题目</title>
    <author>作者</author>
    <content>论文内容</content>
</document>
```

在上面的代码中，张三的文档命名空间为 http://www.html5.cn/zhangsan，而李四的命名空间为 http://www.html5.cn/lisi，虽然他们的文档存在相同的标签，但是借助顶级元素中定义的命名空间，相互之间就不会发生语义冲突。通俗地说，命名空间就是给文档做一个标签，标明该文档是属于哪个网站的。对于 HTML 文档来说，由于它的元素是固定的，不允许用户进行定义，所以指定的命名空间永远为 http://www.w1.org/1999/xhtml。

2.3 HTML5 基础

HTML5 以 HTML4 为基础，对 HTML4 进行了大量的修改。下面简单介绍 HTML5 对 HTML4 进行了哪些修改，以及 HTML5 与 HTML4 之间较大的区别是什么。

2.3.1 HTML5 语法

1. 内容类型

HTML5 的文件扩展名与内容类型保持不变。也就是说，扩展名仍然为.html 或.htm，内容类型仍然为 text/html。

2. 文档类型声明

根据 HTML5 设计化繁为简的准则，文档类型和字符说明都进行了简化。DOCTYPE 声明是 HTML 文件中必不可少的，位于文件第一行。在 HTML4 中，它的声明方法如下。

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

在 HTML5 中，刻意不使用版本声明，一份文档将会适用于所有版本的 HTML。HTML5 中的 DOCTYPE 声明方法（不区分大小写）如下。

```
<!DOCTYPE html>
```

另外，当使用工具时，也可以在 DOCTYPE 声明方式中加入 SYSTEM 识别符，声明方法如下面

的代码所示。

```
<!DOCTYPE HTML SYSTEM "about:legacy-compat">
```

在 HTML5 中，像这样的 DOCTYPE 声明方式是允许的，不区分大小写，引号不区分单引号和双引号。

Note

提示：使用 HTML5 的 DOCTYPE 会触发浏览器以标准兼容模式显示页面。众所周知，网页都有多种显示模式，如怪异模式（Quirks）、近标准模式（Almost Standards）和标准模式（Standards）。其中标准模式也被称为非怪异模式（no-quirks）。浏览器会根据 DOCTYPE 来识别该使用哪种模式，以及使用什么规则来验证页面。

3. 字符编码

在 HTML4 中，使用<meta>元素的形式指定文件中的字符编码，如下所示。

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

在 HTML5 中，可以使用对<meta>元素直接追加 charset 属性的方式来指定字符编码，如下所示。

```
<meta charset="UTF-8">
```

两种方法都有效，可以继续使用前面一种方式，即通过 content 元素的属性来指定。但是不能同时混合使用两种方式。在以前的网站代码中可能会存在下面代码所示的标记方式，但在 HTML5 中，下面这种字符编码方式将被认为是错误的。

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

从 HTML5 开始，对于文件的字符编码推荐使用 UTF-8。

4. 版本兼容性

HTML5 的语法是为了保证与之前的 HTML 语法达到最大程度的兼容而设计的。简单说明如下。

☑　可以省略标记的元素

在 HTML5 中，元素的标记可以省略。具体来说，元素的标记分为 3 种类型：不允许写结束标记、可以省略结束标记、开始标记和结束标记全部可以省略。下面简单介绍这 3 种类型各包括哪些 HTML5 新元素。

第一，不允许写结束标记的元素有：area、base、br、col、command、embed、hr、img、input、keygen、link、meta、param、source、track 和 wbr。

第二，可以省略结束标记的元素有：li、dt、dd、p、rt、rp、optgroup、option、colgroup、thead、tbody、tfoot、tr、td 和 th。

第三，可以省略全部标记的元素有：html、head、body、colgroup 和 tbody。

提示：不允许写结束标记的元素是指不允许使用开始标记与结束标记将元素括起来的形式，只允许使用<元素/>的形式进行书写。例如，
...</br>的书写方式是错误的，正确的书写方式为
。当然，HTML5 之前的版本中
这种写法可以被沿用。

可以省略全部标记的元素是指该元素可以完全被省略。注意，即使标记被省略了，该元素还是以隐式的方式存在的。例如，将 body 元素省略不写时，它在文档结构中还是存在的，可以使用 document.body 进行访问。

☑ 具有布尔值的属性

对于具有 boolean 值的属性，如 disabled 与 readonly 等，当只写属性而不指定属性值时，表示属性值为 true；如果想要将属性值设为 false，可以不使用该属性。另外，要想将属性值设定为 true，也可以将属性名设定为属性值，或将空字符串设定为属性值。例如：

Note

```
<!--只写属性，不写属性值，代表属性为 true-->
<input type="checkbox" checked>
<!--不写属性，代表属性为 false-->
<input type="checkbox">
<!--属性值=属性名，代表属性为 true-->
<input type="checkbox" checked="checked">
<!--属性值=空字符串，代表属性为 true-->
<input type="checkbox" checked="">
```

☑ 省略引号

属性值两边既可以用双引号，也可以用单引号。HTML5 在此基础上做了一些改进，当属性值不包括空字符串、<、>、=、单引号、双引号等字符时，属性值两边的引号可以省略。例如，下面的写法都是合法的。

```
<input type="text">
<input type='text'>
<input type=text>
```

【示例】通过上面介绍的 HTML5 语法知识，下面完全用 HTML5 编写一个文档，在该文档中省略了<html>、<head>、<body>等元素。可以通过本示例复习一下 HTML5 的 DOCTYPE 声明、用<meta>元素的 charset 属性指定字符编码、<p>元素的结束标记的省略、使用<元素/>的方式来结束<meta>元素，以及
元素等本节中所介绍到的知识要点。

```
<!DOCTYPE html>
<meta charset="UTF-8">
<title>HTML5 基本语法</title>
<h1>HTML5 的目标</h1>
<p>HTML5 的目标是为了能够创建更简单的 Web 程序，书写出更简洁的 HTML 代码。
<br/>例如，为了使 Web 应用程序的开发变得更容易，提供了很多 API；为了使 HTML 变得更简洁，开发出了新的属性、新的元素等。总体来说，为下一代 Web 平台提供了许许多多新的功能。
```

这段代码在 IE 浏览器中的运行结果如图 2.3 所示。

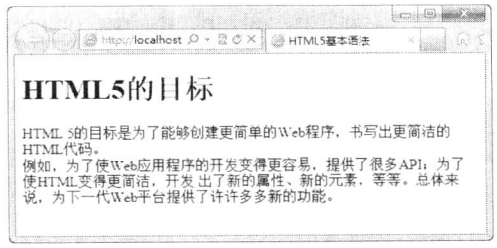


图 2.3 第一个 HTML5 文档

2.3.2 HTML5 元素

HTML5 中引入了很多新的标记元素，根据内容类型的不同，这些元素被分成了 7 大类，如表 2.1

所示。

表 2.1　HTML5 的内容类型

内 容 类 型	说　　明
内嵌	在文档中添加其他类型的内容，如 audio、video、canvas 和 iframe 等
流	在文档和应用的 body 中使用的元素，如 form、h1 和 small 等
标题	段落标题，如 h1、h2 和 hgroup 等
交互	与用户交互的内容，如音频和视频的控件、button 和 textarea 等
元数据	通常出现在页面的 head 中，设置页面其他部分的表现和行为，如 script、style 和 title 等
短语	文本和文本标记元素，如 mark、kbd、sub 和 sup 等

Note

上表中所有类型的元素都可以通过 CSS 来设定样式。虽然 canvas、audio 和 video 元素在使用时往往需要其他 API 来配合，以实现细粒度控制，但它们同样可以直接使用。

1. 新增的结构元素

HTML5 定义了一组新的语义化标记来描述元素的内容。虽然语义化标记也可以使用 HTML 标记进行替换，但是它可以简化 HTML 页面设计，并且搜索引擎在抓取和索引网页的时候，也会利用这些元素。在目前主流的浏览器中已经可以用这些元素了，新增的语义化标记元素如表 2.2 所示。

表 2.2　HTML5 新增的语义化标记元素

元 素 名 称	说　　明
header	标记头部区域的内容（用于整个页面或页面中的一块区域）
footer	标记脚部区域的内容（用于整个页面或页面中的一块区域）
section	Web 页面中的一块区域
article	独立的文章内容
aside	相关内容或者引文
nav	导航类辅助内容

根据 HTML5 效率优先的设计理念，它推崇表现和内容的分离，所以在 HTML5 的实际编程中，开发人员必须使用 CSS 来定义样式。

【示例】在本示例中分别使用 HTML5 提供的各种语义化结构标记重新设计一个网页，效果如图 2.4 所示。

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8" >
<title>HTML5 结构元素</title>
<link rel="stylesheet" href="html5.css">
</head>
<body>
<header>
    <h1>网页标题</h1>
    <h2>次级标题</h2>
    <h4>提示信息</h4>
```

```
</header>
<div id="container">
    <nav>
        <h3>导航</h3>
        <a href="#">链接 1</a> <a href="#">链接 2</a> <a href="#">链接 3</a> </nav>
    <section>
        <article>
            <header>
                <h1>文章标题</h1>
            </header>
            <p>文章内容.....</p>
            <footer>
                <h2>文章注脚</h2>
            </footer>
        </article>
    </section>
    <aside>
        <h3>相关内容</h3>
        <p>相关辅助信息或者服务.....</p>
    </aside>
    <footer>
        <h2>页脚</h2>
    </footer>
</div>
</body>
</html>
```

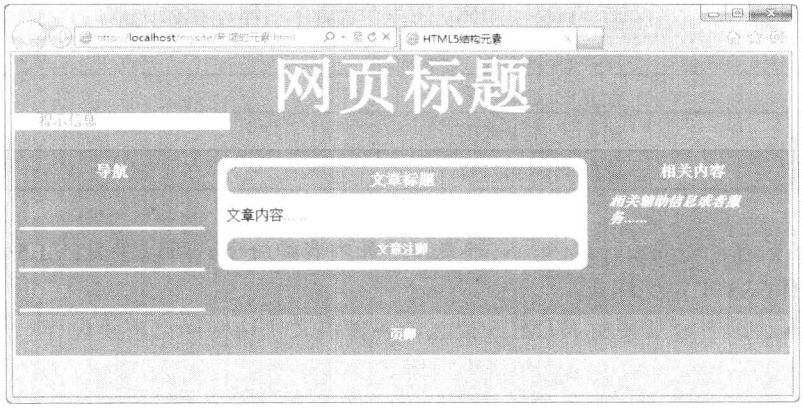


图 2.4　HTML5 语义化结构网页

上面示例中使用了 CSS3 的一些新特性，如圆角（border-radius）和旋转变换（transform:rotate()）等，相关介绍请参阅后面章节的内容。

2. HTML5 新增功能元素

☑ hgroup 元素：用于对整个页面或页面中一个内容区块的标题进行组合。例如：

```
<hgroup>...</hgroup>
```

在 HTML4 中表示为：

```
<div>...</div>
```

☑ figure 元素：表示一段独立的流内容，一般表示文档主体流内容中的一个独立单元。使用 figcaption 元素可为 figure 元素组添加标题。例如：

```
<figure>
    <figcaption>标题</figcaption>
    <p>内容...</p>
</figure>
```

Note

在 HTML4 中表示为：

```
<dl>
    <h1>标题</h1>
    <p>内容...</p>
</dl>
```

☑ video 元素：定义视频，比如电影片段或其他视频流。例如：

```
<video src="movie.ogg" controls="controls">video 元素</video>
```

在 HTML4 中表示为：

```
<object type="video/ogg" data="movie.ogv">
    <param name="src" value="movie.ogv">
</object>
```

☑ audio 元素：定义音频，比如音乐或其他音频流。例如：

```
<audio src="someaudio.wav">audio 元素</audio>
```

在 HTML4 中表示为：

```
<object type="application/ogg" data="someaudio.wav">
    <param name="src" value="someaudio.wav">
</object>
```

☑ embed 元素：用来插入各种多媒体，格式可以是 MIDI、WAV、AIFF、AU、MP3 等。例如：

```
<embed src="horse.wav" />
```

在 HTML4 中表示为：

```
<object data="flash.swf" type="application/x-shockwave-flash"></object>
```

☑ mark 元素：主要用来在视觉上向用户呈现那些需要突出显示或高亮显示的文字。mark 元素的一个比较典型的应用就是在搜索结果中向用户高亮显示搜索关键词。例如：

```
<mark></mark>
```

在 HTML4 中表示为：

```
<span></span>
```

☑ time 元素：表示日期或时间，也可以同时表示两者。例如：

```
<time></time>
```

在 HTML4 中表示为：

```
<span></span>
```

☑ canvas 元素：表示图形，如图表和其他图像。这个元素本身没有行为，仅提供一块画布，但它把一个绘图 API 展现给客户端 JavaScript，以使脚本能够把想绘制的东西绘制到这块画布上。例如：

Note

```
<canvas id="myCanvas" width="200" height="200"></canvas>
```

在 HTML4 中表示为：

```
<object data="inc/hdr.svg" type="image/svg+xml" width="200" height="200">
</object>
```

☑ output 元素：表示不同类型的输出，比如脚本的输出。例如：

```
<output></output>
```

在 HTML4 中表示为：

```
<span></span>
```

☑ source 元素：为媒介元素（比如<video>和<audio>）定义媒介资源。例如：

```
<source>
```

在 HTML4 中表示为：

```
<param>
```

☑ menu 元素：表示菜单列表。当希望列出表单控件时使用该标签。例如：

```
<menu>
    <li><input type="checkbox" />Red</li>
    <li><input type="checkbox" />blue</li>
</menu>
```

在 HTML4 中，menu 元素不被推荐使用。

☑ ruby 元素：表示 ruby 注释（中文注音或字符）。例如：

```
<ruby>汉<rt><rp>(</rp>ㄏㄢˋ<rp>)</rp></rt></ruby>
```

☑ rt 元素：表示字符（中文注音或字符）的解释或发音。例如：

```
<ruby>汉<rt> ㄏㄢˋ</rt></ruby>
```

☑ rp 元素：在 ruby 注释中使用，以定义不支持 ruby 元素的浏览器所显示的内容。例如：

```
<ruby>汉<rt><rp>(</rp>ㄏㄢˋ<rp>)</rp></rt></ruby>
```

☑ wbr 元素：表示软换行。wbr 元素与 br 元素的区别是 br 元素表示此处必须换行；而 wbr 元素的意思是浏览器窗口或父级元素的宽度足够宽时（没必要换行时）不进行换行，而当宽度不够时，主动在此处进行换行。

```
<p> TW3C invites media, analysts, and other attendees of Mobile World Congress (MWC) <wbr> 2012 to meet
with W3C and learn how the Open Web Platform <wbr>is transforming industry. From 27 February through 1 March
W3C will </p>\
```

☑ command 元素：表示命令按钮，如单选按钮、复选框或按钮。例如：

```
<command onclick=ucut()" label="cut">
```

☑ details 元素：表示用户要求得到并且可以得到的细节信息，可以与 summary 元素配合使用。summary 元素提供标题或图例。标题是可见的，用户单击标题时，会显示出细节信息。summary 元素应该是 details 元素的第一个子元素。例如：

Note

```
<details>
    <summary>HTML5</summary>
    For the latest updates from the HTML WG, possibly including important bug fixes, please look at the editor's draft instead. There may also be a more up-to-date Working Draft with changes based on resolution of Last Call issues..
</details>
```

☑ datalist 元素

datalist 元素表示可选数据的列表，与 input 元素配合使用，可以制作出输入值的下拉列表。例如：

```
<datalist></datalist>
```

☑ datagrid 元素：表示可选数据的列表，它以树形列表的形式来显示。例如：

```
<datagrid></datagrid>
```

☑ keygen 元素：表示生成密钥。例如：

```
<keygen>
```

☑ progress 元素：表示运行中的进程，可以使用 progress 元素来显示 JavaScript 中耗费时间的函数的进程。例如：

```
<meter></meter>
```

☑ email：表示必须输入 E-mail 地址的文本输入框。
☑ url：表示必须输入 URL 地址的文本输入框。
☑ number：表示必须输入数值的文本输入框。
☑ range：表示必须输入一定范围内数字值的文本输入框。
☑ Date Pickers：HTML5 拥有多个可供选取日期和时间的新型输入文本框：
- date—选取日、月、年。
- month—选取月和年。
- week—选取周和年。
- time—选取时间（小时和分钟）。
- datetime—选取时间、日、月、年（UTC 时间）。
- datetime-local—选取时间、日、月、年（本地时间）。

3. HTML5 中废除的元素

在 HTML5 中废除了 HTML4 中过时的元素，简单介绍如下。

☑ 能使用 CSS 替代的元素

对于 basefont、big、center、font、s、strike、tt 和 u 这些元素，由于它们的功能都是表现文本效果，而 HTML5 中提倡把呈现性功能放在 CSS 样式表中统一编辑，所以将这些元素废除了，并使用编辑 CSS、添加 CSS 样式表的方式进行替代。其中 font 元素允许由“所见即所得”的编辑器来插入，s 元素、strike 元素可以由 del 元素替代，tt 元素可以由 CSS 的 font-family 属性替代。

☑ 不再使用 frame 框架

对于 frameset 元素、frame 元素与 noframes 元素，由于 frame 框架对网页可用性存在负面影响，在 HTML5 中已不支持 frame 框架，只支持 iframe 框架，或者用服务器方创建的由多个页面组成的复

合页面的形式，同时将以上这 3 个元素废除。

☑ 只有部分浏览器支持的元素

对于 applet、bgsound、blink 和 marquee 等元素，由于只有部分浏览器支持这些元素，特别是 bgsound 元素以及 marquee 元素，只被 IE 所支持，所以在 HTML5 中被废除。其中 applet 元素可由 embed 元素或 object 元素替代，bgsound 元素可由 audio 元素替代，marquee 可以由 JavaScript 编程的方式替代。

Note

其他被废除元素还有：

☑ 使用 ruby 元素替代 rb 元素。
☑ 使用 abbr 元素替代 acronym 元素。
☑ 使用 ul 元素替代 dir 元素。
☑ 使用 form 元素与 input 元素相结合的方式替代 isindex 元素。
☑ 使用 pre 元素替代 listing 元素。
☑ 使用 code 元素替代 xmp 元素。
☑ 使用 GUIDS 替代 nextid 元素。
☑ 使用“text/plian”MIME 类型替代 plaintext 元素。

2.3.3 HTML5 增加及废除的属性

HTML5 同时增加和废除了很多属性。简单说明如下。

1. 增加的属性

☑ 表单属性

➢ 为 input（type=text）、select、textarea 与 button 元素新增加 autofocus 属性。它以指定属性的方式让元素在画面打开时自动获得焦点。
➢ 为 input 元素（type=text）与 textarea 元素新增加 placeholder 属性，它会对用户的输入进行提示，提示用户可以输入的内容。
➢ 为 input、output、select、textarea、button 与 fieldset 新增加 form 属性，声明它属于哪个表单，然后将其放置在页面上任何位置，而不是表单之内。
➢ 为 input 元素（type=text）与 textarea 元素新增加 required 属性。该属性表示在用户提交的时候进行检查，检查该元素内一定要有输入内容。
➢ 为 input 元素增加 autocomplete、min、max、multiple、pattern 和 step 属性。同时还有一个新的 list 元素与 datalist 元素配合使用。datalist 元素与 autocomplete 属性配合使用。multiple 属性允许在上传文件时一次上传多个文件。
➢ 为 input 元素与 button 元素增加了新属性 formaction、formenctype、formmethod、formnovalidate 与 formtarget，它们可以重载 form 元素的 action、enctype、method、novalidate 与 target 属性。为 fieldset 元素增加了 disabled 属性，可以把它的子元素设为 disabled（无效）状态。
➢ 为 input 元素、button 元素和 form 元素增加了 novalidate 属性，该属性可以取消提交时进行的有关检查，表单可以被无条件地提交。

☑ 链接属性

➢ 为 a 元素与 area 元素增加了 media 属性，该属性规定目标 URL 是为哪种类型的媒介或设备进行优化的，只能在 href 属性存在时使用。
➢ 为 area 元素增加了 hreflang 属性与 rel 属性，以保持与 a 元素、link 元素的一致。

- ➢ 为 link 元素增加了新属性 sizes。该属性可以与 icon 元素结合使用（通过 rel 属性），用于指定关联图标（icon 元素）的大小。
- ➢ 为 base 元素增加了 target 属性，主要目的是保持与 a 元素的一致性。

☑ 其他属性

- ➢ 为 ol 元素增加 reversed 属性，用于指定列表倒序显示。
- ➢ 为 meta 元素增加 charset 属性，因为这个属性已经被广泛支持了，而且为文档的字符编码的指定提供了一种比较良好的方式。
- ➢ 为 menu 元素增加了两个新的属性——type 与 label。label 属性为菜单定义一个可见的标注，type 属性让菜单可以以上下文菜单、工具条或列表菜单这 3 种形式出现。
- ➢ 为 style 元素增加 scoped 属性，用来规定样式的作用范围，譬如只对页面上某个树起作用。
- ➢ 为 script 元素增加 async 属性，用于定义脚本是否异步执行。
- ➢ 为 html 元素增加 manifest 属性，开发离线 Web 应用程序时，它与 API 结合使用，定义一个 URL，在这个 URL 上描述文档的缓存信息。
- ➢ 为 iframe 元素增加 3 个属性 sandbox、seamless 与 srcdoc，用来提高页面安全性，防止不信任的 Web 页面执行某些操作。'

2. 废除的属性

HTML5 废除了 HTML4 中过时的属性，而采用其他属性或其他方案进行替代，具体说明如表 2.3 所示。

表 2.3　HTML5 废除的属性

HTML 4 属性	适 应 元 素	HTML5 替代方案
rev	link、a	rel
charset	link、a	在被链接的资源中使用 HTTP Content-type 头元素
shape、coords	a	使用 area 元素代替 a 元素
longdesc	img、iframe	使用 a 元素链接到较长描述
target	link	多余属性，被省略
nohref	area	多余属性，被省略
profile	head	多余属性，被省略
version	html	多余属性，被省略
name	img	id
scheme	meta	只为某个表单域使用 scheme
archive、classid、codebase、codetype、declare、standby	object	使用 data 与 type 属性类调用插件。需要使用这些属性来设置参数时，使用 param 属性
valuetype、type	param	使用 name 与 value 属性，不声明值的 MIME 类型
axis、abbr	td、th	使用以明确简洁的文字开头，后跟详述文字的形式。可以对更详细内容使用 title 属性，来使单元格的内容变得简短
scope	td	在被链接的资源中使用 HTTP Content-type 头元素
align	caption、input、legend、div、h1、h2、h3、h4、h5、h6、p	使用 CSS 样式表替代

续表

HTML 4 属性	适 应 元 素	HTML5 替代方案
alink、link、text、vlink、background、bgcolor	body	使用 CSS 样式表替代
align、bgcolor、border、cellpadding、cellspacing、frame、rules、width	table	使用 CSS 样式表替代
align、char、charoff、height、nowrap、valign	tbody、thead、tfoot	使用 CSS 样式表替代
align、bgcolor、char、charoff、height、nowrap、valign、width	td、th	使用 CSS 样式表替代
align、bgcolor、char、charoff、valign	tr	使用 CSS 样式表替代
align、char、charoff、valign、width	col、colgroup	使用 CSS 样式表替代
align、border、hspace、vspace	object	使用 CSS 样式表替代
clear	br	使用 CSS 样式表替代
compact、type	ol、ul、li	使用 CSS 样式表替代
compact	dl	使用 CSS 样式表替代
compact	menu	使用 CSS 样式表替代
width	pre	使用 CSS 样式表替代
align、hspace、vspace	img	使用 CSS 样式表替代
align、noshade、size、width	hr	使用 CSS 样式表替代
align、frameborder、scrolling、marginheight、marginwidth	iframe	使用 CSS 样式表替代
autosubmit	menu	

2.3.4 HTML5 全局属性

在 HTML5 中新增了全局属性的概念。所谓全局属性是指可以对任何元素都使用的属性。

1. contentEditable 属性

contentEditable 属性的主要功能是允许用户在线编辑元素中的内容。contentEditable 是一个布尔值属性，可以被指定为 true 或 false。此外，该属性还有个隐藏的 inherit（继承）状态，属性值为 true 时，元素被指定为允许编辑；属性值为 false 时，元素被指定为不允许编辑；未指定 true 或 false 时，则由 inherit 状态来决定，如果元素的父元素是可编辑的，则该元素就是可编辑的。

【示例】在本示例中为列表元素加上 contentEditable 属性后，该元素就变成可编辑的了，读者可自行在浏览器中修改列表内容。

```
<!DOCTYPE html>
<head>
<meta charset="UTF-8">
<title>conentEditalbe 属性示例</title>
</head>
<h2>可编辑列表</h2>
<ul contentEditable="true">
    <li>列表元素 1</li>
    <li>列表元素 2</li>
```

```
    <li>列表元素 3</li>
</ul>
```

这段代码运行后的结果如图 2.5 所示。

原始列表　　编辑列表项项目

图 2.5　可编辑列表

在编辑完元素中的内容后，如果想要保存这些内容，只能把该元素的 innerHTML 发送到服务器端进行保存，因为改变元素内容后该元素的 innerHTML 内容也会随之改变，目前还没有特别的 API 来保存编辑后元素中的内容。

contentEditable 属性支持的元素包括：defaults、A、ABBR、ACRONYM、ADDRESS、B、BDO、BIG、BLOCKQUOTE、BODY、BUTTON、CENTER、CITE、CODE、CUSTOM、DD、DEL、DFN、DIR、DIV、DL、DT、EM、FIELDSET、FONT、FORM、hn、I、INPUT type=button、INPUT type=password、INPUT type=radio、INPUT type=reset、INPUT type=submit、INPUT type=text、INS、ISINDEX、KBD 和 LABEL。

2. designMode 属性

designMode 属性用来指定整个页面是否可编辑，当页面可编辑时，页面中任何支持上文所述的 contentEditable 属性的元素都变成了可编辑状态。designMode 属性只能在 JavaScript 脚本里被编辑修改。该属性有两个值：on 与 off。属性被指定为 on 时，页面可编辑；被指定为 off 时，页面不可编辑。使用 JavaScript 脚本来指定 designMode 属性的用法如下所示。

```
document.designMode="on"
```

针对 designMode 属性，各浏览器的支持情况也各不相同。

- ☑ IE8：出于安全考虑，不允许使用 designMode 属性让页面进入编辑状态。
- ☑ IE9：允许使用 designMode 属性让页面进入编辑状态。
- ☑ Chrome 3 和 Safari：使用内嵌 frame 的方式，该内嵌 frame 是可编辑的。
- ☑ Firefox 和 Opera：允许使用 designMode 属性让页面进入编辑状态。

3. hidden 属性

在 HTML5 中，所有的元素都允许使用一个 hidden 属性。该属性类似于 input 元素中的 hidden 元素，功能是通知浏览器不渲染该元素，使该元素处于不可见状态。但是元素中的内容还是浏览器创建的，也就是说页面装载后允许使用 JavaScript 脚本将该属性取消，取消后该元素变为可见状态，同时元素中的内容也即时显示出来。hidden 属性是一个布尔值的属性，当设为 true 时，元素处于不可见状态；当设为 false 时，元素处于可见状态。

4. spellcheck 属性

spellcheck 属性是 HTML5 针对 input 元素（type=text）与 textarea 这两个文本输入框提供的一个

```
<img src="images/bg1.jpg" />
</body>
</html>
```

Note

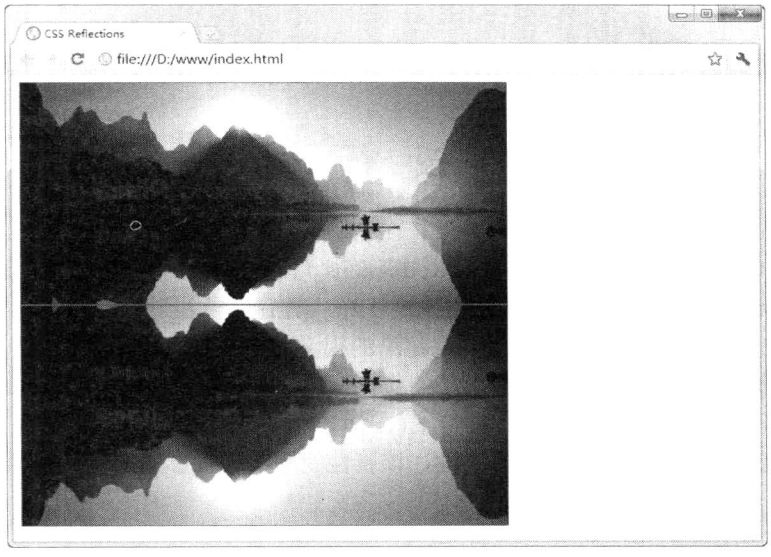


图 19.7　定义简单的倒影效果

在上面示例基础上，可以为倒影设置距离，向下偏移 10 像素，预览效果如图 19.8 所示。

```
<style type="text/css">
img {
    height:250px;
    -webkit-box-reflect:below 10px;
}
</style>
```

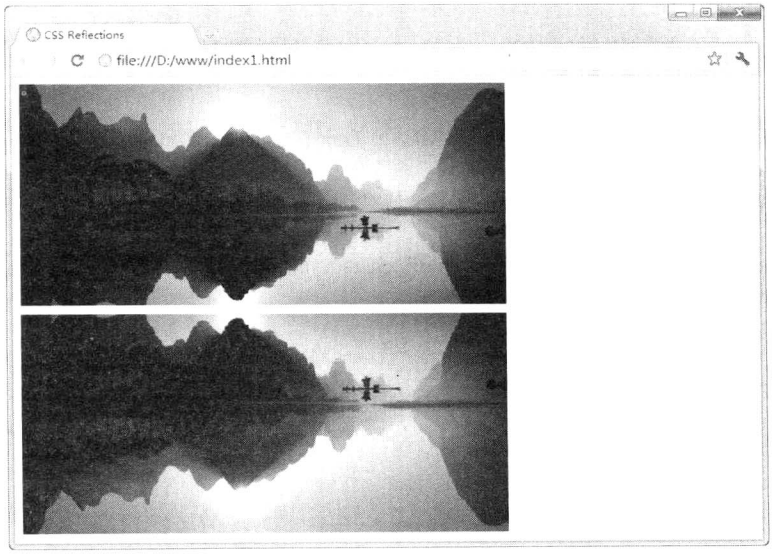


图 19.8　调整倒影距离后的效果

也可以设计渐变倒影，通过渐变遮罩逐渐盖住下面倒影，营造渐隐效果，样式代码如下，预览效果如图 19.9 所示。

Note

```
<style type="text/css">
img {
    height:250px;
    -webkit-box-reflect:below 5px -webkit-gradient(linear, left top, left bottom, from(transparent), color-stop(0.5, transparent), to(white));
}
</style>
```

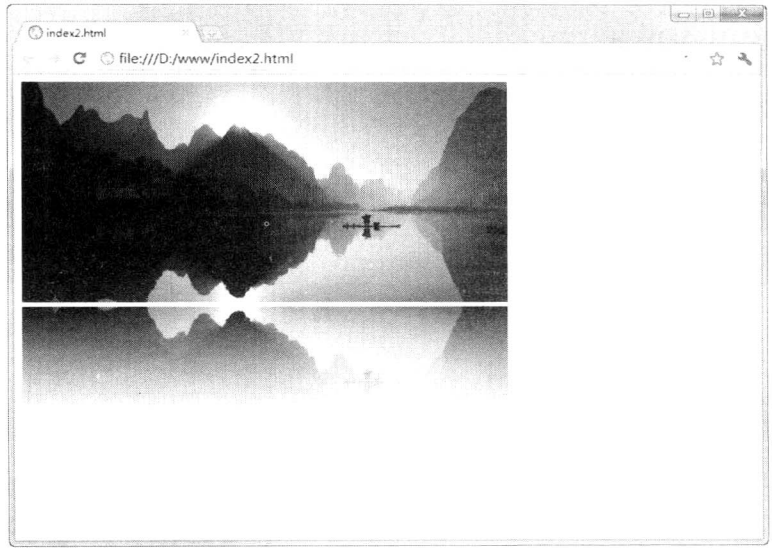


图 19.9　设置渐变倒影效果

【示例 2】除了图片可以设计倒影外，实际上网页上任何对象都可以设计倒影效果，本示例演示了如何为文本设置倒影，代码如下，预览如图 19.10 所示。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>CSS Reflections</title>
<style type="text/css">
img {
    height: 200px;
    position: absolute;
    right: 0;
    bottom: 0;
}
div {
    border: 1px solid #666;
    color: #666;
    -webkit-box-reflect: below 5px;
}
```

```
h1 { text-align: center; }
</style>
</head>
<body>
<div>
    <h1>《雨巷》--- 戴望舒</h1>
    <p>撑着油纸伞，独自<br />
        彷徨在悠长、悠长<br />
        又寂寥的雨巷<br />
        我希望逢着<br />
        一个丁香一样地<br />
        结着愁怨的姑娘 </p>
</div>
<img src="images/bg2.jpg" />
</body>
</html>
```

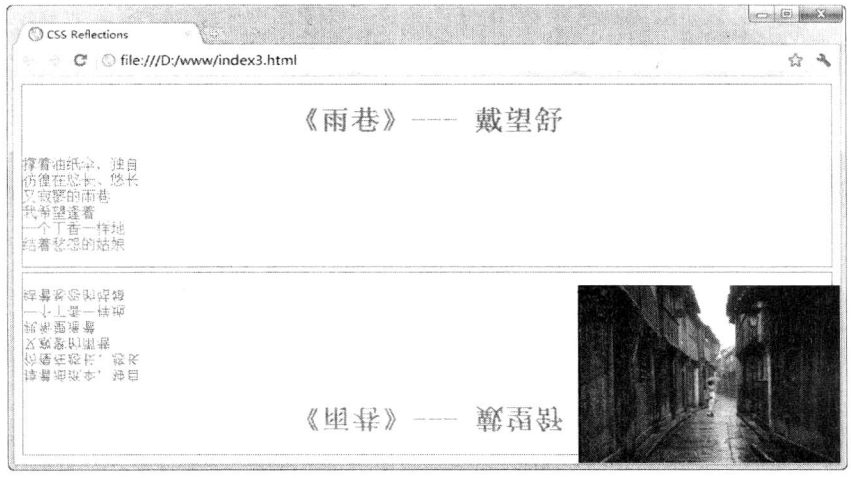


图 19.10　设置文本倒影效果

还可以为视频等多媒体界面设计倒影，这样能够设计出非常动感的视觉效果，这里不再一一举例进行演示。

有关此电子图书的说明

本人由于一些便利条件，可以帮您提供各种中文电子图书资料，且质量均为清晰的 PDF 图片格式，质量要高于网上大量传播的一些超星 PDG 的图书。方便阅读和携带。只要图书不是太新，文学、法律、计算机、人文、经济、医学、工业、学术等方面的图书，我都可以帮您找到电子版本。所以，当你想要看什么图书时，可以联系我。我的 QQ 是：85013855，大家可以在 QQ 上联系我。

此 PDF 文件为本人亲自制作，请各位爱书之人尊重个人劳动，敬请您不要修改此 PDF 文件。因为这些图书都是有版权的，请各位怜惜电子图书资源，不要随意传播，否则，这些资源更难以得到。