第 1 章

HTML5 概述

1

这是一本关于 HTML5 编程的书。不过在学习之前，有必要先了解一下背景知识，什么是 HTML5？它经历了怎样的发展历程？HTML4 和 HTML5 有什么区别？

本章中，我们会集中讨论大家关注的一些实际问题。为什么是 HTML5？为什么它能掀起风潮？是什么设计理念使得 HTML5 真正具有革命性的进步？HTML5 如何在大幅改动的同时保持高度兼容？无插件范式意味着什么？HTML5 包含什么，不包含什么？HTML5 新增加了哪些特性，为什么能揭开整个 Web 开发新时代的序幕？下面我们一起来了解一下。

1.1 HTML5 发展史

HTML 的历史可以追溯到很久以前。1993 年 HTML 首次以因特网草案的形式发布。20 世纪 90 年代的人见证了 HTML 的大幅发展，从 2.0 版，到 3.2 版和 4.0 版（一年出了两个版本！），再到 1999 年的 4.01 版。随着 HTML 的发展，W3C（万维网联盟）掌握了对 HTML 规范的控制权。

然而，在快速发布了这四个版本之后，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。不过在此期间，HTML 体现了顽强的生命力，主要的网站内容还是基于 HTML 的。为能支持新的 Web 应用，同时克服

现有的缺点，HTML 迫切需要添加新功能，制定新规范。

致力于将 Web 平台提升到一个新的高度，一小组人在 2004 年成立了 WHATWG（Web Hypertext Application Technology Working Group，Web 超文本应用技术工作组）。他们创立了 HTML5 规范，同时开始专门针对 Web 应用开发新功能——这被 WHATWG 认为是 HTML 中最薄弱的环节。Web 2.0 这个新词也就正是在那个时候被发明的。Web2.0 实至名归，开创了 Web 的第二个时代。旧的静态网站逐渐让位于需要更多特性的动态网站和社交网站——这其中的新功能真的是数不胜数。

2006 年，W3C 又重新介入 HTML，并于 2008 年发布了 HTML5 的工作草案。2009 年，XHTML 2 工作组停止工作。又过了一年，也就到了现在。因为 HTML5 能解决非常实际的问题（随后可以看到），所以在规范还未定稿的情况下，各大浏览器厂家就已经按耐不住了，开始对旗下产品进行升级以支持 HTML5 的新功能。这样，得益于浏览器的实验性反馈，HTML5 规范也得到了持续地完善，HTML5 以这种方式迅速融入到了对 Web 平台的实质性改进中。

HTML 的过去和未来

“ 大家好，我是 Brian[①]，HTML 的铁杆老粉丝。

1995 年，我创建了第一个属于自己的个人主页。那时候的‘ 主页’ 就是用来介绍自己的。上面的照片通常不清晰，代码中用了很多<blink>标签，页面上会告诉大家我住在哪儿、读过什么书、正在做什么跟计算机相关的工作。我和我的那些所谓‘ 万维网开发者’ 不是在大学里读书

① Brian，本书作者之一。——译者注

就是在大学里工作。

那时候的 HTML 非常初级，没有任何工具可用。Web 应用几乎没有，顶多有少量的文本处理脚本。页面代码都是用大家各自喜欢的文本编辑器写出来的。页面的更新频率基本上是数周或者数月。

不知不觉，我们已经走过了漫长的 15 个年头。

今天，用户对其在线资料一天更新很多次已经是很平常的事了。当然，如果没有在线工具持续稳定的更新换代，也不会有今天这样的交互方式。

提醒各位读者，大家在看这本书的时候心里要明白，我们的示例虽然现在看起来非常简单，但潜力是巨大的。就像 20 世纪 90 年代中期那些率先使用`<img>`标签的人一样，他们又怎么会知道在十年以后，很多人都已经在线编辑和储存照片了；而我们要有这种前瞻性。

我们希望书中示例的基本思路能够激发读者无穷的创意，从而为 Web 的下个十年奠定新的基础。”

——Brian

1.2 关于 2022 年的那个神话

今天，我们看到的 HTML5 规范已经以工作草案的形式发布了——还不是最终版。那什么时候 HTML5 规范才能尘埃落定呢？现在就来了解一下几个关键时间点。第一个时间点是 2012 年，目标是发布候选推荐版。第二个时间点是 2022 年，目标是发布计划推荐版。哦！那等着吧，还早着呢！可能大家会这么想，然后就把书合上，扔到一边，等十年后再说。那就大错特错了，在

明白这两个时间点的真正意义之前，可别急着下结论。

第一个，也就是最近的 2012 年，可以说是最重要的时间点，因为这个时间点一到就意味着 HTML5 规范编写完成了。想象一下，这并不久远，也就两年后的事情。计划推荐版（普遍认为距今还有点远）的重要性在于届时将会有两个对 HTML5 的互通实现，意味着将有两个浏览器会完全支持整个 HTML5 规范的所有功能——这个远大的目标让 2022 年这个时间点看起来又似乎太近了。毕竟，现在连 HTML 4 都还没有实现这个目标呢[①]。

关键是现在浏览器厂家已经着手支持 HTML5 中很多优秀的新功能了。只要用户有需求，现在就可以利用这些新功能进行 Web 应用的开发。虽然一些细节方面的改造还会持续进行，相应的 Web 应用可能需要改动，不过，相对于使用 HTML5 为用户带来的体验来讲，这点付出不算什么。当然，如果用户的浏览器是 IE6 的话，很多新功能是不支持的，需要模拟——不过这也不能成为抛弃 HTML5 的理由，毕竟这些用户最终都会升级浏览器版本，很多可能会直接选用 IE9，而且微软承诺在 IE9 中持续增加对 HTML5 的支持。实际上，通过使用新的浏览器和改进的模拟技术意味着用户现在和不久的将来便可以使用很多 HTML5 功能了。

1.3　谁在开发 HTML5

我们都知道开发 HTML5 需要成立相应的组织，并且肯定需要有人来负责。这正是下面这三个重要组织的工作。

① HTML 4 最早于 1997 年成为 W3C 推荐标准，到现在 10 多年早已经过去了，仍然不存在两个完全支持这一规范的浏览器。——编者注

- WHATWG：由来自 Apple、Mozilla、Google、Opera 等浏览器厂商的人组成，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（Internet Engineering Task Force，因特网工程任务组）：这个任务组下辖 HTTP 等负责 Internet 协议的团队。HTML5 定义的一种新 API（WebSocket API）依赖于新的 WebSocket 协议，IETF 工作组正在开发这个协议。

1.4 新的认识

HTML5 是基于各种各样的理念（在 WHATWG 规范中有详述）进行设计的，这些设计理念体现了对可能性和可行性的新认识。

- 兼容性
- 实用性
- 互通性
- 通用访问性

1.4.1 兼容性和存在即合理

别担心，HTML5 并不是颠覆性的革新。相反，实际上 HTML5 的一个核心理念就是保持一切新特性平滑过渡。一旦浏览器不支持 HTML5 的某项功能，针对功能的备选行为就会悄悄进行。再说，互联网上有些 HTML 文档已经存在 20 多年了，因此，支持所有现存 HTML 文档是非常重要的。

HTML5 的研究者们还花费了大量的精力来研究通用行为。比如，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。例如，很多开发人员使用 `DIV id="header"`来标记页眉区域。HTML5 不就是要解决实际问题吗？那何不直接添加一个 `<header>`标签呢？

尽管 HTML5 标准的一些特性非常具有革命性，但是 HTML5 旨在进化而非革命。毕竟没有从头再来的必要。(就算有必要的话，也不应该是 HTML5，起码也要发明一个更好的！)

1.4.2　效率和用户优先

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，只是有些情况下还不够完美。

看看这个示例，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，肯定会有人反对这种不严格的语法，我们不去辩论对错，只去关心一个底线，那就是最终用户其实并不在乎代码怎么写。当然，我们并不提倡入门者一开始写代码就这么不严谨，毕竟归根结底，受害者还是最终用户，因为一旦由于开发人员的原因造成页面错误导致不能正常显示，那么被折磨的肯定是最终用户。

HTML5 也衍生出了 XHTML5（可通过 XML 工具生成有效的 HTML5 代码）。HTML 和 XHTML 两种版本的代码经过序列化应该可以生成近乎一样的 DOM 树。显然 XHTML 的验证规则严格得多，刚才示例中后两行代码是无法通过验证的。

1. 安全机制的设计

为保证 HTML5 足够安全，HTML5 在设计时就做了大量的工作。规范中的各个部分都有专门针对安全的章节，并且安全是被优先考虑的。HTML5 引入了一种新的基于来源的安全模型，该模型不仅易用，而且对各种不同的 API 都通用。这个安全模型可以让我们做一些以前做不到的事情，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。在这方面，我们肯定不会怀念过去的“ 好” 时光了。

2. 表现和内容分离

在清晰分离表现和内容方面，HTML5 迈出了巨大的步伐。HTML5 在所有可能的地方都努力进行了分离，也包括 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，但得益于先前提到的 HTML5 在兼容性方面的设计理念，那些功能仍然能用。表现和内容分离的概念也不是全新的，在 HTML 4 Transitional 和 XHTML 1.1 中就已经开始用了。Web 设计者把这个概念当做最佳实践使用了很久，不过现在清晰地分开表现和内容显得更为重要，否则会有如下弊端：

- 可访问性差；
- 不必要的复杂度（所有样式代码都放在页面中，代码可读性很差）；
- 文件变大（样式内容越多，文件越大），带来的后果就是页面载入变慢。

1.4.3　化繁为简

HTML5 要的就是简单、避免不必要的复杂性。HTML5 的口号是“ 简单至上，尽可能简化”。因此，HTML5 做了以下这些改进：

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

随后我们将详细讲解这些改进。

为了实现所有的这些简化操作，HTML5 规范已经变得非常大，因为它需要精确再精确。实际上要比以往任何版本的 HTML 规范都要精确。为了达到在 2022 年能够真正实现浏览器互通的目标，HTML5 规范制订了一系列定义明确的行为；任何歧义和含糊都可能延缓这一目标的实现。

另外，HTML5 规范比以往的任何版本都要详细，为的是避免造成误解。HTML5 规范的目标是完全、彻底地给出定义，特别是对 Web 应用。所以也难怪，整个规范超过了 900 页！

基于多种改进过的、强大的错误处理方案，HTML5 具备了良好的错误处理机制。非常有现实意义的一点是，HTML5 提倡重大错误的平缓恢复，再次把最终用户的利益放在了第一位。比如，如果页面中有错误的话，在以前可能会影响整个页面的显示，而 HTML5 不会出现这种情况，取而代之的是以标准方式显示“ broken”标记，这要归功于 HTML5 中精确定义的错误恢复机制。

1.4.4　通用访问

这个原则可以分成三个概念。

- 可访问性：出于对残障用户的考虑，HTML5 与 WAI（Web Accessibility Initiative，Web 可访问性倡议）和 ARIA（Accessible Rich Internet Applicaions，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：例如，新的`<ruby>`元素支持在东亚页面排版中会用到的 Ruby 注释。

1.5 无插件范式

过去，很多功能只能通过插件或者复杂的 hack（本地绘图 API、本地 socket 等）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（例如 Apple 的 iPad 就不支持 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（因为插件边界、剪裁和透明度问题）。

虽然一些插件的安装率很高，但在控制严格的公司内部网络环境中经常会被封锁。此外，由于插件经常还会给用户带来烦人的广告，一些用户也会选择屏蔽此类插件。这样的话，一旦用户禁用了插件，就意味着依赖该插件显示的内容也无法表现出来了。

在已经设计好的页面中，要想把插件显示的内容与页面上其他元素集成也比较困难，因为会引起剪裁和透明度等问题。插件使用的是自带的渲染模型，与普通 Web 页面所使用的不一样，

所以当弹出菜单或者其他可视化元素与插件重叠时，会特别麻烦。此时，HTML5 却可以站出来，挥舞着它的原生功能魔棒，对这类问题笑而不语，它可以直接用 CSS 和 JavaScript 的方式控制页面布局。实际上这是 HTML5 的最大亮点，显示了先前任何 HTML 版本都不具备的强大能力。HTML5 不仅仅是提供新元素支持新功能，更重要的是添加了对脚本和布局之间的原生交互能力，基于此，我们可以实现以前不能实现的效果。

以 HTML5 中的 `canvas` 元素为例，有很多非常底层的事情以前是没办法做到的（比如在 HTML4 的页面中就难画出对角线），而有了 `canvas` 就可以很轻易地实现了。更为重要的是新 API 释放出来的潜能，以及通过寥寥几行 CSS 代码就能完成布局的能力。基于 HTML5 的各类 API 的优秀设计，我们可以轻松地对它们进行组合应用。比如，从 `video` 元素中抓取的帧可以显示在 `canvas` 里面，用户点击 `canvas` 即可播放这帧对应的视频文件。这只是一个使用原生方法实现插件功能的示例。其实，当工作不再基于黑盒后，开发反而会变得更简单。HTML5 的不同功能组合应用为 Web 开发注入了一股强大的新生力量，这也是我们为什么决定写一本关于 HTML5 编程的书，而不单单是介绍那些新元素的原因。

HTML5 包括什么，不包括什么

那么，HTML5 到底包括些什么？仔细阅读过规范的读者，可能会发现本书中讲解的很多功能其实在规范中是没有的。例如，Geolocation 和 Web Workers 就不在规范中。那为什么还要将它们纳入本书的讨论范围呢？炒作？当然不是！

很多 HTML5 的研究成果（如 Web Storage 和 Canvas 2D）起初都是 HTML5 规范的一部分，

后来移出来列入了单独的规范中，这么做是为了让 HTML5 规范更好地突出重点。在成为官方规范之前，先单列出来进行讨论和编辑不失为一个好办法。这样的话，即使存在争议，也不会影响到整个规范。

在讨论某个功能的时候，特定领域的专家可通过邮件列表的方式共同探讨，不会因为喋喋不休而引起激辩。业界仍然倾向于把原始功能集都视为 HTML5，其中包括 Geolocation。这样的话，HTML5 不仅涵盖了核心的标记元素，同时也可以包括很多很酷的新 API。写这本书的时候，下面这些功能也属于 HTML5：

- 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

可以看到本书中所讨论到的 API 大多都在上面的列表中。为什么选择这些 API 呢？我们挑选的都是基本成熟的功能，也就是说这些功能已经得到了不止一种浏览器的支持，其他功能（不太

成熟的）可能只在个别浏览器的某个 beta 版中可用，或者基本上只是个概念。

对于我们要讨论的 HTML5 功能，本书将提供各种浏览器的最新支持情况。不过，不管现在支持情况如何，不久的将来肯定会变，因为 HTML5 发展的速度非常快。不用担心，我们会推荐一些非常好的在线资源，用以查看当前（以及将来）浏览器的支持情况。www.caniuse.com网站按照浏览器的版本提供了详尽的 HTML5 功能支持情况。若用户通过浏览器访问 www.html5test.com的话，该网站会直接显示用户浏览器对 HTML5 规范的支持情况。

此外，本书的重点不是讨论使用某种模拟或者变通的方法让 HTML5 程序能够运行在旧浏览器上，而是着重关注 HTML5 规范本身，以及它的使用方法。也就是说，在本书中我们针对所讨论的每个 API 都会提供一些示例代码，开发人员可以直接拿来检测其可用性。因为检测用户代理的方式经常不可靠，所以我们使用特性检测。当然，还可以使用 Modernizr —— 一个 JavaScript 库，它提供了非常先进的 HTML5 和 CSS3 检测功能。我们强烈推荐使用 Modernizr，因为它是检测浏览器是否支持某些特性的最佳工具。

对于 HTML 多说几句

“ 大家好，我是 Frank[①]，我喜欢画画。

我见过的第一个 HTML canvas 演示是一款简单的绘图程序，用户界面模仿的是微软的画图程序。尽管那落后于数码绘图几十年，并且当时只有个别浏览器支持，不过它却让我对其表现能力充满了期待。

① Frank，本书作者之一。——译者注

当我开始数码绘图的时候，一般都使用安装在本地的桌面软件。不可否认，有些软件相当不错，但它们不具备 Web 应用的迷人特性。简而言之，这些软件都是离线的。想要共享数码作品，必须先从软件中将图像导出，然后上传至 Web。但是在 Web 的实时画布上协作讨论的话就不存在这种问题了。HTML5 应用可以省掉现在数码绘图流程中的导出环节，将整个创作过程都转移到线上，直至作品完成。

不能用 HTML5 实现的应用已经变得越来越少了。对于文本，Web 已然成为双向沟通的理想媒介。通过全 Web 的方式处理文本的应用程序比比皆是，而类似绘图、视频编辑、3D 建模等图形类程序才刚刚起步。

现在，我们已经开发出了许多功能强大的单机软件，用以创建和欣赏图片、音乐、电影等。更进一步，我们可以将这些软件移植到 Web 这个功能强大、无处不在的在线平台上。”

——Frank

1.6 HTML5 的新功能

在讨论 HTML5 编程之前，让我们快速预览一下 HTML5 的新功能。

1.6.1 新的 DOCTYPE 和字符集

首先，根据 HTML5 设计准则的第 3 条——化繁为简，Web 页面的 `DOCTYPE` 被极大地简化了。以下面这段 HTML4 `DOCTYPE` 代码为例进行对比：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"□↵
"http://www.w3.org/TR/html4/loose.dtd">
```

谁能记得住？所以在新建页面的时候，我们往往只能通过复制粘贴的方式添加这么长的

DOCTYPE，同时脑子里还不确定复制的对不对。HTML5 干净利索地解决了这个问题：

```
<!DOCTYPE html>
```

现在的 DOCTYPE 好记多了。跟 DOCTYPE 一样，字符集的声明也被简化了。过去是这样的：

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

现在成了：

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

使用新的 DOCTYPE 后，浏览器默认以标准模式（standards mode）显示页面。例如，用 Firefox 打开一个 HTML5 页面，然后点击“工具➤页面信息”（Tools ➤ Page Info），会看到图 1-1 所示的画面。示例页面是以标准模式显示的。

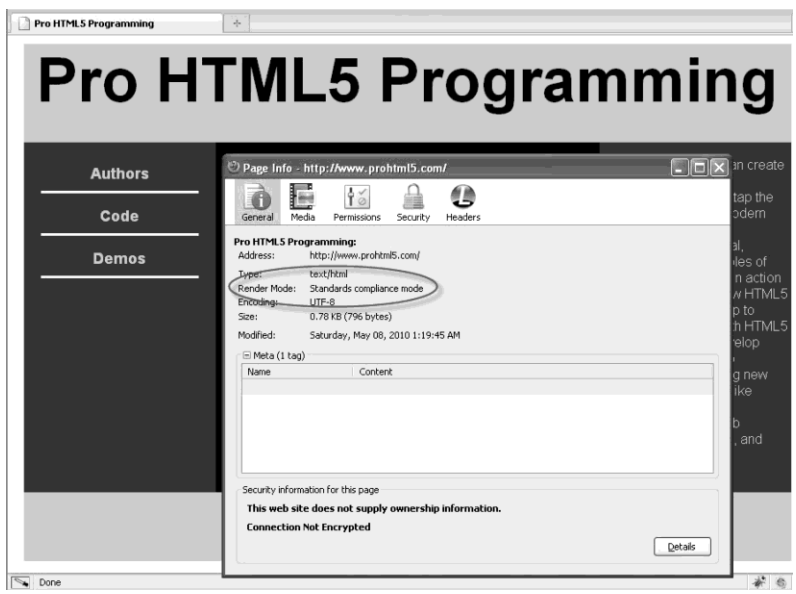

图 1-1　标准兼容模式下显示的页面

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，Web 页面有

多种显示模式，比如怪异模式（Quirks）、近标准模式（Almost Standards）以及标准模式（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。在怪异模式下，浏览器会尽量不中断页面显示，即使没有完全通过验证也会将其显示出来。HTML5 引入了新的标记元素和其他机制（随后会详细讨论），因此如果坚持使用已废弃的元素，那么页面将无法通过验证。

1.6.2 新元素和旧元素

HTML5 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。见表 1-1。

表1-1 HTML5的内容类型

内容类型	描 述
内嵌	向文档中添加其他类型的内容，例如audio、video、canvas和iframe等
流	在文档和应用的body中使用的元素，例如form、h1和small等
标题	段落标题，例如h1、h2和hgroup等
交互	与用户交互的内容，例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，例如script、style和title等
短语	文本和文本标记元素，例如mark、kbd、sub和sup等
片段	用于定义页面片段的元素，例如article、aside和title等

上述所有类型的元素都可以通过 CSS 来设定样式。此外，虽然其中一些元素，如 canvas、audio 和 video，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。我们在后续章节中详细讨论这类元素 API。

限于篇幅，本书讨论的内容无法涵盖所有新元素，不过片段类元素是全新的，我们会在下一

节讨论，而 canvas、audio 和 video 作为 HTML5 新增的元素也会在后续章节中详细讨论。

同样地，对于旧的标签元素，网上的资料已经很多了，我们不会把所有旧的标签元素都罗列出来。不过需要注意的是，HTML5 中移除了很多在行内设样式的标记元素，如 big、center、font 和 basefont 等，以鼓励开发人员使用 CSS。

1.6.3　语义化标记

片段类的内容类型包含许多 HTML5 元素。HTML5 定义了一种新的语义化标记来描述元素的内容。虽然语义化标记不会让你马上感受到有什么好处，但是它可以简化 HTML 页面设计，并且将来搜索引擎在抓取和索引网页的时候，也绝对会利用到这些元素的优势。

前面我们说过，HTML5 的宗旨之一就是存在即合理。Google 分析了上百万的页面，从中发现了 DIV 标签的通用 ID 名称重复量很大。例如，很多开发人员喜欢使用 DIV id="footer"来标记页脚内容，所以 HTML5 引入了一组新的片段类元素，在目前主流的浏览器中已经可以用了。表 1-2 列出了新增的语义化标记元素。

表1-2　HTML5中新的片段类元素

元 素 名	描　　述
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

上面所有的元素都能用 CSS 设定样式。之前说到了 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。代码清单 1-1 是一个 HTML5 页面的概貌，其中使用了新的 DOCTYPE、字符集和语义化标记元素——新的片段类元素。示例代码对应的源码在 code/intro 文件夹中。

代码清单 1-1 HTML5 示例页面

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" >
  <title>HTML5</title>
  <link rel="stylesheet" href="html5.css">

</head>

<body>

   <header>
     <h1>Header</h1>
     <h2>Subtitle</h2>
     <h4>HTML5 Rocks!</h4>
   </header>

     <div id="container">
        <nav>
           <h3>Nav</h3>
           <a href="http://www.example.com">Link 1</a>
           <a href="http://www.example.com">Link 2</a>
           <a href="http://www.example.com">Link 3</a>
        </nav>

            <section>
            <article>
              <header>
                <h1>Article Header</h1>
              </header>
              <p>Lorem ipsum dolor HTML5 nunc aut nunquam sit amet, consectetur
adipiscing

                  elit. Vivamus at est eros, vel fringilla urna.</p>
                <p>Per inceptos himenaeos. Quisque feugiat, justo at vehicula

                   pellentesque, turpis lorem dictum nunc.</p>
                <footer>
                  <h2>Article Footer</h2>
                </footer>
```

```
                </article>

                <article>
                  <header>
                    <h1>Article Header</h1>
                  </header>
                  <p>HTML5: "Lorem ipsum dolor nunc aut nunquam sit amet, consectetur
adipiscing

                     elit. Vivamus at est eros, vel fringilla urna. Pellentesque odio</p>

                  <footer>
                    <h2>Article Footer</h2>
                  </footer>
                </article>

            </section>

                <aside>
              <h3>Aside</h3>
              <p>HTML5: "Lorem ipsum dolor nunc aut nunquam sit amet, consectetur
adipiscing

                 elit. Vivamus at est eros, vel fringilla urna. Pellentesque odio
rhoncus</p>
            </aside>

                 <footer>
                 <h2>Footer</h2>
            </footer>
        </div>
      </body>

      </html>
```

没有样式的页面看起来有些枯燥乏味。代码清单1-2是一些可以用来设置内容样式的 CSS 代码。需要注意的是，这份样式表使用了 CSS3的一些新特性，比如圆角（`border-radius`）和旋转变换（`transform:rotate();`）。CSS3同 HTML5一样也正在开发过程中，并且为了便于浏览器逐步支持，也采用了模块化的方式发布子规范（例如变换（transformation）、动画（animation）和过渡（transition）分别对应不同的子规范）。

CSS3 的规范很可能还会变动，CSS3 中的功能也处于实验期，因此为了避免命名空间冲突，

这些功能都会加上浏览器厂商的前缀。要显示圆角、渐变（gradients）、阴影（shadows）和变形（transformations）的话，需要在声明的部分加上前缀：-moz-（Mozilla 浏览器）、o-（Opera 浏览器）和-webkit-（Safari 和 Chrome 等基于 WebKit 核心的浏览器）。

代码清单 1-2 HTML5 页面对应的 CSS 文件

```
body {
        background-color:#CCCCCC;
        font-family:Geneva,Arial,Helvetica,sans-serif;
        margin: 0px auto;
        max-width:900px;
        border:solid;
        border-color:#FFFFFF;
}

header {
        background-color: #F47D31;
        display:block;
        color:#FFFFFF;
        text-align:center;
}

header h2 {
        margin: 0px;
}

h1 {
        font-size: 72px;
        margin: 0px;
}

h2 {
        font-size: 24px;
        margin: 0px;
        text-align:center;
        color: #F47D31;
}

h3 {
        font-size: 18px;
        margin: 0px;
        text-align:center;
        color: #F47D31;
}

h4 {
        color: #F47D31;
```

```
        background-color: #fff;
        -webkit-box-shadow: 2px 2px 20px #888;
        -webkit-transform: rotate(-45deg);
        -moz-box-shadow: 2px 2px 20px #888;
        -moz-transform: rotate(-45deg);
        position: absolute;
        padding: 0px 150px;
        top: 50px;
        left: -120px;
        text-align:center;
}

nav {
        display:block;
        width:25%;
        float:left;
}

nav a:link, nav a:visited {
        display: block;
        border-bottom: 3px solid #fff;
        padding: 10px;
        text-decoration: none;
        font-weight: bold;
        margin: 5px;
}

nav a:hover {
        color: white;
        background-color: #F47D31;
}

nav h3 {
        margin: 15px;
        color: white;
}

#container {
        background-color: #888;
}

section {
        display:block;
        width:50%;
        float:left;
}

article {
        background-color: #eee;
        display:block;
        margin: 10px;
        padding: 10px;
        -webkit-border-radius: 10px;
```

```
        -moz-border-radius: 10px;
        border-radius: 10px;
        -webkit-box-shadow: 2px 2px 20px #888;
        -webkit-transform: rotate(-10deg);
        -moz-box-shadow: 2px 2px 20px #888;
        -moz-transform: rotate(-10deg);
}

article header {
        -webkit-border-radius: 10px;
        -moz-border-radius: 10px;
        border-radius: 10px;
        padding: 5px;
}

article footer {
        -webkit-border-radius: 10px;
        -moz-border-radius: 10px;
        border-radius: 10px;
        padding: 5px;
}

article h1 {
        font-size: 18px;
}

aside {
        display:block;
        width:25%;
        float:left;
}

aside h3 {
        margin: 15px;
        color: white;
}

aside p {
        margin: 15px;
        color: white;
        font-weight: bold;
        font-style: italic;
}

footer {
        clear: both;
        display: block;
        background-color: #F47D31;
        color:#FFFFFF;
        text-align:center;
        padding: 15px;
}
```

```
footer h2 {
        font-size: 14px;
        color: white;
}

/* links */
a {
        color: #F47D31;
}

a:hover {
        text-decoration: underline;
}
```

图 1-2 是代码清单 1-1 中的页面应用了 CSS (包括部分 CSS3) 之后的显示效果。其实并不能把这个页面当成所谓的典型 HTML5 页面。因为计划赶不上变化，这个示例使用了很多新标签只是为了演示而已。

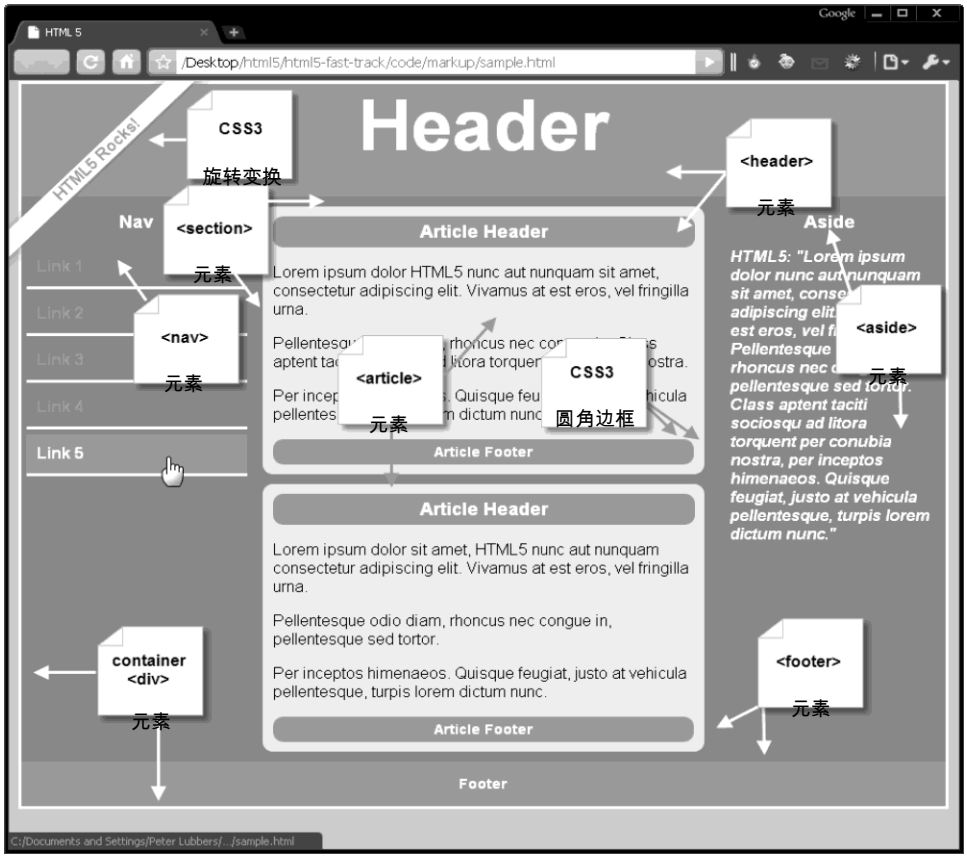

图 1-2 使用了所有新的语义化标记元素的 HTML5 页面

最后需要说明的是，看起来好像浏览器是因为识别了新的元素，所以显示出了对应的内容。其实不然，事实上这些元素很可能是先被重命名为了 `foo` 或者 `bar`，然后再应用样式，最后才显示出来的（当然，对于搜索引擎优化来说没有任何好处）。IE 是个特例，因为 IE 需要将这些元素都作为 DOM 的一部分，所以要想在 IE 中看到这些元素，必须用编程的方式把它们插入 DOM 中，然后再以块元素（block element）的形式显示出来。

1.6.4 使用 Selectors API 简化选取操作

除了语义化元素外，HTML5 还引入了一种用于查找页面 DOM 元素的快捷方式。表 1-3 列出了在 HTML5 出现之前，用来在页面中查找特定元素的函数。

表1-3　以前用来查找元素的JavaScript方法

函　数	描　述	示　例
`getElementById()`	根据指定的`id`特性值查找并返回元素	`<div id="foo">` `getElementById("foo");`
`getElementsByName()`	返回所有`name`特性为指定值的元素	`<input type="text" name="foo">` `getElementsByName("foo");`
`getElementsByTagName()`	返回所有标签名称与指定值相匹配的元素	`<input type="text">` `getElementsByTagName("input");`

有了新的 Selectors API 之后，可以用更精确的方式来指定希望获取的元素，而不必再用标准 DOM 的方式循环遍历。Selectors API 与现在 CSS 中使用的选择规则一样，通过它我们可以查找页面中的一个或多个元素。例如，CSS 已经可以基于嵌套（nesting）、兄弟节点（sibling）和子模式（child pattern）进行元素选择。CSS 的最新版除添加了更多对伪类(pseudo-classe)的支持（例如判断一个对象是否被启用、禁用或者被选择等），还支持对属性和层次的随意组合叠加。使用表 1-4 中的函数就能按照 CSS 规则来选取 DOM 中的元素。

表1-4　新**`QuerySelector`**方法

函　数	描　述	示　例	结　果
`querySelector()`	根据指定的选择规则，返回在页面中找到的第一个匹配元素	`querySelector("input.error");`	返回第一个CSS类名为“error”的文本输入框
`querySelectorAll()`	根据指定规则返回页面中所有相匹配的元素	`querySelectorAll("#results td");`	返回`id`值为`results`的元素下所有的单元格

可以为 Selectors API 函数同时指定多个选择规则，例如：

```
// 选择文档中类名为 highClass 或 lowClass 的第一个元素
```

```
var x = document.querySelector(".highClass", ".lowClass");
```

对于 querySelector()来说，选择的是满足规则中任意条件的第一个元素。对于querySelector- All()来说，页面中的元素只要满足规则中的任何一个条件，都会被返回。多条规则是用逗号分隔的。

以前在页面上跟踪用户操作很困难，但新的 Selectors API 提供了更为便捷的方法。比如，页面上有一个表格 ,我们想获取鼠标当前在哪个单元格上。从代码清单 1-3 中可以看到使用 Selectors API 实现有多简单。这份源代码也可以从 code/intro 路径下找到。

代码清单 1-3 使用 Selector API

```
<!DOCTYPE html>
<html>

<head>
  <meta charset="utf-8" />
  <title>Query Selector Demo</title>

  <style type="text/css">
    td {
      border-style: solid;
      border-width: 1px;
      font-size: 300%;
    }

    td:hover {
      background-color: cyan;
    }

    #hoverResult {
      color: green;
      font-size: 200%;
    }
  </style>
</head>

<body>
  <section>
    <!-- create a table with a 3 by 3 cell display -->
   <table>
      <tr>
```

```
          <td>A1</td> <td>A2</td> <td>A3</td>
        </tr>
        <tr>
          <td>B1</td> <td>B2</td> <td>B3</td>
        </tr>
        <tr>
          <td>C1</td> <td>C2</td> <td>C3</td>
        </tr>
      </table>

      <div>Focus the button, hover over the table cells, and hit Enter to identify them
using querySelector('td:hover').</div>
      <button type="button" id="findHover" autofocus>Find 'td:hover' target</button>
      <div id="hoverResult"></div>


        <script type="text/javascript">
          document.getElementById("findHover").onclick = function() {
            // 找到鼠标当前悬停的单元格
            var hovered = document.querySelector("td:hover");
            if (hovered)
                document.getElementById("hoverResult").innerHTML = hovered.innerHTML;
          }
        </script>
      </section>

    </body>
    </html>
```

从以上示例可以看到，仅用一行代码即可找到用户鼠标下面的元素：

```
var hovered = document.querySelector("td:hover");
```

提示　Selectors API不仅仅只是方便，在遍历DOM的时候，Selectors API通常会比以前的子节点搜索API更快。为了实现快速样式表，浏览器对选择器匹配进行了高度优化。

不难理解为什么 W3C 中的 Selectors API 标准规范会从 CSS 规范中单独分离出来，从上面的代码也可以看出来，Selectors API 在样式应用以外同样大有作为。虽然本书不会深入讲解 Selectors API 的全部细节，但是对于希望优化 DOM 操作方式的 Web 开发人员来说，建议使用新的 Selectors API 以便快速查询应用架构。

1.6.5 JavaScript 日志和调试

JavaScript 日志和浏览器内调试从技术上讲虽然不属于 HTML5 的功能，但在过去的几年里，相关工具的发展出现了质的飞跃。第一个可以用来分析 Web 页面及其所运行脚本的强大工具是一款名为 Firebug 的 Firefox 插件。

现在，相同的功能在其他浏览器的内嵌开发工具中也可以找到：Safari 的 Web Inspector、Google 的 Chrome 开发者工具(Developer Tools)、IE 的开发者工具(Developer Tools)，以及 Opera 的 Dragonfly。图 1-3 是 Google 的 Chrome 开发者工具截图，显示了大量与当前 Web 页面相关的信息（使用快捷键 Ctrl+Shift+J 可以看到），包括调试控制台、资源视图、存储视图等。

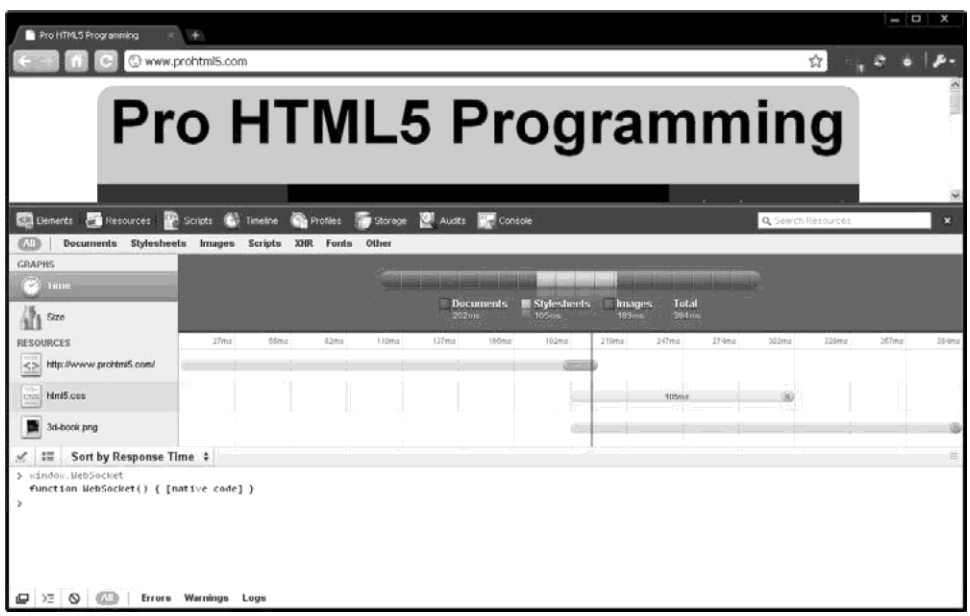

图 1-3 Chrome 的开发者工具

很多调试工具支持设置断点来暂停代码执行、分析程序状态以及查看变量的当前值。

`console.log` API 已经成为 JavaScript 开发人员记录日志的事实标准。为了便于开发人员查看记录到控制台的信息，很多浏览器提供了分栏窗格的视图。`console.log` API 要比 `alert()` 好用很多，因为它不会阻塞脚本的执行。

1.6.6　window.JSON

JSON 是一种相对来说比较新并且正在日益流行的数据交换格式。作为 JavsScript 语法的一个子集，它将数据表示为对象字面量。由于其语法简单和在 JavaScript 编程中与生俱来的兼容性，JSON 变成了 HTML5 应用内部数据交换的事实标准。典型的 JSON API 包含两个函数，`parse()` 和 `stringify()`（分别用于将字符串序列化成 DOM 对象和将 DOM 对象转换成字符串）。

如果在旧的浏览器中使用 JSON，需要 JavaScript 库（有些可以从http://json.org 找到）。在JavaScript中执行解析和序列化效率往往不高，所以为了提高执行速度，现在新的浏览器原生扩展了对 JSON 的支持，可以直接通过 JavaScript 来调用 JSON 了。这种本地化的 JSON 对象被纳入了 ECMAScript 5 标准，成为了下一代 JavaScript 语言的一部分。它也是 ECMAScript 5 标准中首批被浏览器支持的功能之一。所有新的浏览器都支持 window.JSON，将来 JSON 必将大量应用于 HTML5 应用中。

1.6.7　DOM Level 3

事件处理是目前 Web 应用开发中最令人头疼的部分之一。除了 IE 以外，绝大多数浏览器都支持处理事件和元素的标准 API。早期 IE 实现的是与最终标准不同的事件模型，而 IE9 将会支持 DOM Level 2 和 DOM Level 3 的特性。如此，在所有支持 HTML5 的浏览器中，我们终于可以使

用相同的代码来实现 DOM 操作和事件处理了，包括非常重要的 `addEventListener()`和`dispatchEvent()`方法。

1.6.8 Monkeys、Squirrelfish 和其他 JavaScript 引擎

最新一轮的浏览器创新不仅仅是增加了新的标签和 API。最重要的变化之一是主流浏览器中 JavaScript/ECMAScript 引擎飞快的升级。新的 API 提供了很多上一代浏览器无法实现的功能，因而脚本引擎整体执行效率的提升，不论对现有的，还是使用了最新 HTML5 特性的 Web 应用都有好处。还记得浏览器在显示复杂图像、处理数据或者编辑长篇文章时，明显变得迟钝的情景吗？再好好想一想。

最近几年，浏览器厂商争相比拼，看谁能开发出更快的 JavaScript 引擎。过去的 JavaScript 纯粹是被解释执行，而最新的引擎则直接将脚本编译成原生机器代码，相比 2005 年前后的浏览器，速度的提升已经不在一个数量级上了。

大约从 2006 年 Adobe 将其 JIT 编译引擎和代号为 Tamarin 的 ECMAScript 虚拟机捐赠给 Mozilla 基金会开始，竞争的序幕就拉开了。尽管新版的 Mozilla 中 Tamarin 技术已经所剩无几，但 Tamarin 的捐赠促进了各家浏览器对新脚本引擎的研发，而这些引擎的名字就如同他们声称的性能一样有意思。

表1-5 Web浏览器的JavaScript引擎

浏 览 器	引擎名称	备 注
Apple Safari 5	Nitro（也被称作Squirrel Fish Extreme）	Safari 4中发布，在Safari 5中提升性能，包括字节码优化和上下文线程的本地编译器
Google Chrome 5	V8	自从Chrome 2开始，使用了新一代垃圾回收机制，

		可确保内存高度可扩展而不会发生中断
Microsoft Internet Explorer 9	Chakra	注重于后台编译和高效的类型系统，速度比IE8快10倍
Mozilla Firefox 4	JägerMonkey	从3.5版本优化而来，结合了快速解释和源自追踪树（trace tree）的本地编译
Opera 10.60	Carakan	它采用了基于寄存器的字节码和选择性本地编译的方式，声称效率比10.50版本提升了75%

总之，得益于浏览器厂商间的良性竞争，JavaScript 的执行性能越来越接近于本地桌面应用程序。

关于 HTML 再多说两句

“ 我的名字叫 Peter[①]，说起竞争和疯狂的速度，我非常喜欢跑步。

马拉松是一项伟大的运动，从中可以发现伟大的人。当跑到百公里赛或者 165 公里长跑的最后阶段时，你真的可以通过一种新的方式来认识一些志同道合的人。因为在这个时候，人们放下了自己的架子，为真正伟大友谊的诞生提供了机会。可以肯定，竞争的因素仍然存在，但更重要的是那份深切的情谊。哦，我有点儿跑题了。

有的比赛我没有时间参加（比如在写这本 HTML5 的书的时候），但我想知道比赛中我的朋友们表现如何，于是通常会上比赛网站去看。当然了，网站中的‘ 实时跟踪’ 功能往往不那么可靠。

几年前，我偶然间遇到一家为欧洲赛事建立的网站，这个网站的思路是完全正确的。他们为跑在前面的运动员安装了 GPS 定位器，然后在地图上显示出来（本书中我们将使用 Geolocation 和 WebSocket 来模拟实现）。尽管事实上执行起来比较麻烦（为了看到最新的数据，用户必须频

① Peter，本书作者之一。——译者注

繁点击‘刷新’），但是我从中仍然看到了难以置信的潜力。

现在，仅仅过了几年，HTML5 便通过 API 的方式为我们提供了建立这类实时比赛网站所需要的工具，例如位置服务应用所需的 Geolocation，以及用来支持实时更新的 WebSocket。在我看来，毋庸置疑，HTML5 冲破了终点线成为了赢家！

——Peter

1.7 小结

本章概述了 HTML5 的重要特性。

我们讨论了 HTML5 的开发历史和即将迎来的几个重要时间点，还讲述了 HTML5 时代的四个新设计准则：兼容性、实用性、互通性和通用访问性。每项设计准则都打开了一扇大门，同时也宣告了已经过时的惯例和约定的消亡。接着，我们介绍了 HTML5 令人意想不到的新的无插件范式，回答了每个人都挂在嘴边的问题——HTML5 规范到底包括什么，不包括什么，还回顾了 HTML5 的新特性，例如新的 `DOCTYPE` 和字符集声明，许多新的标记元素等，最后我们讨论了 JavaScript 引擎的竞争。

下一章起，我们开始探索 HTML5 编程，旅途的起点是 Canvas API。