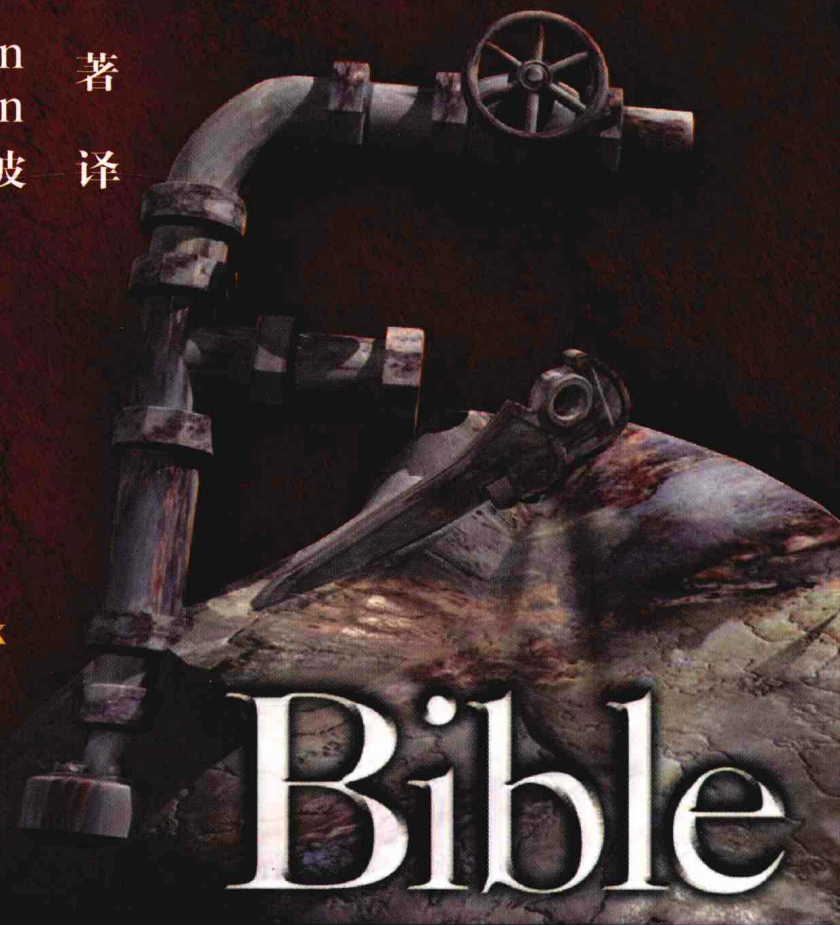
WILEY

本书由JavaScript创始人Brendan Eich作序

丰富 权威 实用

JavaScript 宝典（第6版）

[美] Danny Goodman 著
Michael Morrison
张文波 译



创建具有交互能力的现代网站

编写当今浏览器上运行的脚本

使用文档对象模型，包括Ajax

CD-ROM

本书配套光盘内容：
23个附赠章节
300个可运行的脚本
10个实际情况的JavaScript应用

Bible

人民邮电出版社
POSTS & TELECOM PRESS

JavaScript宝典（第6版）

[美] Danny Goodman 著
Michael Morrison
张文波 译

人民邮电出版社
北京

图书在版编目（CIP）数据

JavaScript宝典：第6版 /（美）古德曼（Goodman, D.），（美）莫里森（Morrison, M.）著；张文波译. —北京：人民邮电出版社，2009.6
ISBN 978-7-115-19338-4

Ⅰ. J… Ⅱ. ①古…②莫…③张… Ⅲ. JAVA语言－程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2008）第193120号

版 权 声 明

JavaScript 宝典（第 6 版）

◆ 著　[美] Danny Goodman　Michael Morrison
译　张文波
责任编辑　陈　昇

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

◆ 开本：787×1092　1/16
印张：47.25
字数：2 125 千字　2009 年 6 月第 1 版
印数：1－3 000 册　2009 年 6 月北京第 1 次印刷

著作权合同登记号　图字：01-2008-0495 号
ISBN 978-7-115-19338-4/TP

定价：99.00 元（附光盘）

读者服务热线：(010)67132705　印装质量热线：(010)67129223
反盗版热线：(010)67171154

内容提要

JavaScript 脚本用于创建 Web 浏览器支持的交互式网页，是一种简单易学的 Web 编程语言。本书全面系统地介绍了客户端 JavaScript 脚本编程技术。从简单介绍 JavaScript 与 Web 开始，循序渐进地介绍了 Web 浏览器、基本的编程技巧和编程步骤。接着，深入探讨了浏览器文档对象模型的发展，详细介绍了 JavaScript 对象的有关知识，包括对象的属性、方法和事件处理程序等内容。最后，细致地讲述了 JavaScript 核心语言，包括 String，Math，Date 和 Array 等核心对象以及异常处理等内容。通过本书的学习，读者可以根据实际需要制作出自己的动态网页，全方位拓展自己的网页设计知识和基本技能。

本书内容全面、示例典型，适合各层次的网页设计人员学习和参考。

序 言

作为 JavaScript 的创建者，对于 JavaScript 的过去和将来，对于它能帮助读者做什么，我想说几句话。

JavaScript 的诞生，源自 HTML 的作者有直接在文档中编写脚本的需要。现在来看，这是显而易见的，但在 1995 年的春天，这是新兴事物，不亚于一种智慧的新提议（HTML 应当仅描述静态文档结构），还有一件重要事件（Java applet，人们宣称它是扩展网页和使网页更活泼的唯一实际方式）。回顾过去人们的相关看法，JavaScript 就在我的脑海中成型了：

- **“Java 方式的”语法**。一个朋友借给我一本书《The Complete HyperCard Handbook》作者是 Goodman，看了这本书，我刚刚知道，HyperTalk 的“自然语言”语法，随后又有了一件更吸引人的新鲜事物：Java applet 的脚本编程。如果脚本编程语言类似 Java，那么，那些从 JavaScript 转向 Java 的那些程序员，就会对这种语法上的相似性感到十分高兴。而坚持 Java 的类和类型声明，或者在一行结束时每个语句之后都加一个分号，就可以解决这个问题，对于多数人来说，脚本编程不过就是编写短小的代码片断，既快速又轻松。
- **HTML 元素的事件**。按钮应当有 `onClick` 事件处理器。文档从窗口中加载和卸载，所以，窗口应当有 `onLoad` 和 `onUnload` 处理器。用户和脚本提交表单，所以要有 `onSubmit` 处理器。尽管开始没有 HyperCard 消息那样灵活（它的处理器启示了 onEvent 命名约定），利用 JavaScript 事件，HTML 作者能够使用远程服务器控制用户交互，而且能够快速响应用户操作和浏览器的动作。通过采用 W3C DOM Level 2 事件处理建议方案，现代浏览器中的 JavaScript 对于事件有完全灵活的控制。
- **无类对象**。“自编程”语言检验了基于原型继承的概念。对于 JavaScript，我希望每个对象有一个原型（为了简单和有效），默认情况下，根据调用的函数使用新的操作符（为了与 Java 一致）。为了避免区别方法函数与构造函数，所有函数都接收对象，作为属性命名它们，在参数中调用。尽管直到 Navigator 3 还没有出现原型，但是，在版本 2 中，还是通过作为对象处理的引用文本而有所预示（Strong 对象原型，用户可以与方法相关联）。
- **生成的 HTML**。在 HTML 中嵌入 JavaScript，这就提出了这样的思考：让脚本“说”HTML，就像是代替脚本本身，“发射出”加载的文本和标记。这些可能性不仅仅是自动修改当前或最后修改的日期，还计算整个表树，所有重复的结构都会在一个脚本循环中上卷，而变化的内容则会作为表格方式，形成一个目录或者小型数据库。

首先，我认为，JavaScript 的很大作用是，验证 HTML 表单的输入。但是，在很久以前，我很惊讶地看到，有许多 Web 设计者借助于脚本生成的 HTML 和 JavaScript 对象，设计出引人注目的应用程序。很显然，从用户演示和反馈来说，Web 设计者一直寻求仅仅使用几个图片、HTML 和 JavaScript 快速而有效地建立重要应用的方式。最终，他们要求浏览器支持动态 HTML（一个有趣的网站：`http://www.javascript-games.org/`）。

许多 Web 作者都具备 JavaScript 的编写能力，而且，证明了脚本编程环境的关键优势，超越了传统的应用开发。HTML 和 JavaScript 语言不仅相当易于使用，而且开发也无需编程的专门技能就能处理所有像素和所有事件，就像在大型的传统应用中一样。

在当今的 Web 上，JavaScript 的首要任务是，维护 HTML 作者早期对脚本编程语言的信任。通过保持像素处理功能，具有图像的 HTML 已经“制造出”大量的 Web 设计者；通过保持事件处理功能，JavaScript 已经帮助成千上万的设计者成为程序员。Web 开发与应用开发“会聚”在一起的最佳例子可能是 Mozilla 浏览器，甚至一些自定义的工具和模块组件都完全使用 JavaScript、层叠样式表（CSS）、自定义的基于 XML 的标记语言和图像来实现。

JavaScript 也是一种常规的语言，它的用途与 HTML 和 XML 有所不同。它已经嵌入到服务器、编写工具、浏览器插件和其他种类的浏览器中（比如 3D 图形世界）。它的国际化的标准、ECMA-262（ISO 16262）已经发展到第 3 版。但是，与 Perl 甚至 Java 这样的语言相比，JavaScript 还很年轻。在这个语言第 4 版中，支持可选类型、类以及 ECMA 技术委员会提出的版本控制机制（参见 ECMA 技术委员会的 JS2 提议，文档在网站 `http://www.mozilla.org/js/language/js20` 中）。

显然，对于笔者来说，如果没有富有创造性的、诚恳耐心的开发人员团体，可能还不会使用 JavaScript。对此，笔者表示由衷的感谢。可以说，那些使用 Navigator 2 beta 版本的开发人员，那些通过电子邮件和网络新闻公布重要问题解决方法以及功能特征的人员，都可以说是这个语言的再生父母。开发人员的支持和反馈，使得 JavaScript 取得了巨大的成功。

本书汇集了具有专家和教师视角的数千开发人员的经验。Danny 不知道，他的 HyperCard 书对笔者有多少启发，在 1995 年，整个 JavaScript 的开发过程中，笔者就拥有这本书。他的活力、同情心，还有清晰的文笔，都在帮助笔者创建一种“普适一切的语言”。编写本书序使笔者感到极大的满足，这本书也会让读者感到心满意足。

对于想要学习 JavaScript 的每个人，强烈建议您阅读 Danny Goodman 的这本宝典，尤其是对于那些仅仅编写了一点点脚本或程序的 HTML 作者。拥有这本详实可靠的图书陪伴您，您的脚本编程之旅会更加轻松而有趣。

Brendan Eich
The Mozilla Organization（`http://www.mozilla.org`）

前　言

近 25 年以来，笔者编写了一些有助于学习或使用新技术的书籍。只要可能，笔者都会尽量从新的创作或编程环境的最开始出发，感受发展过程中经历的困难，与读者一起分享奋斗的结果。在本书第 6 版中，凝聚了 10 多年来在编写 JavaScript 脚本的工作中，以及主持新闻组过程中积累的知识和经验，在新闻组中，经常遇到各个层次的脚本编写人员的疑问、困难和挑战。笔者的目标是，使读者避免笔者所遇到的相同困难、挫折，以及避免笔者在脚本浏览器的多个版本中遇到的问题。

虽然本书的最早版本主要集中在当时的主流浏览器 Netscape Navigator 上，但目前浏览器的市场已经发生了许多变化。多年来，Microsoft 出品的 Internet Explorer 在此领域已经遥遥领先，近年来，其他支持业界标准的浏览器在用户计算机中也有所应用。所以，对于内容开发人员而言，就面临着艰难的选择：要求所设计的脚本内容，在符合标准的浏览器以及专有环境中，都能良好地发挥作用。本书之所以叫做“宝典（bible）”，是因为不仅说明了在标准和专有环境之间有分歧的细节是什么，还指出了如何编写适应不同情况的脚本，让它们能够应用到访问网站或 Web 应用的更广泛的浏览器中。通过本书的学习，读者将提高设计和编写与浏览器无关的优秀脚本的能力。笔者在本书中确实有所偏好，就是主要考虑业界标准，但专有特征也不能排除在外，这是因为，我们希望所编写的脚本能顺利运行于现在和将来尽可能多的浏览器上。

本书的组织和特点

与前两个版本一样，这个版本包含了更为丰富的信息，难以在一本书中完全体现出来。本书配套光盘提供了 23 个附赠章节。

在本书中提供了最常用的信息和引用，可以轻松学习 JavaScript 的基础知识。下面说明本书结构的一些具体内容。

第 1 部分　JavaScript 起步

本书第 1 部分的第 1 章中将 JavaScript 与 Java 进行了比较，并讨论了在万维网中的作用。自从 JavaScript 问世以来，Web 浏览器和脚本世界经历了巨大的变化，因此第 2 章集中阐述了脚本编写者面临的问题，这些脚本编写者必须在标准飞速发展的同时，为单平台和跨平台浏览器的用户开发应用程序。第 3 章开始谈到 JavaScript，在这里可以编写第一个应用脚本。

第 2 部分　JavaScript 教程

第 2 部分适用于 JavaScript 初学者。共有 9 章内容，循序渐进地讲解浏览器原理、基本编程技巧和实用的 JavaScript 脚本，重点针对当今多数脚本浏览器支持的业界标准。每一章后的练习有助于加强理解刚学到的知识，并引导你使用新知识，附录 C 提供了答案。本部分的目标是，使读者掌握编写简单脚本页面的足够知识，并且有助于理解更深入的讨论和本书其他部分的例子。

第 3 部分　文档对象参考

第 3 部分是本书内容最多的一部分，它深入探讨了当今浏览器中实现的文档对象模型，包括现代 Ajax 应用使用的对象。在所有的参考章节中，兼容性图表显示了支持每个对象和对象特征的浏览器版本。尤其是，在第 15 章中包含了许多的参考资料，第 3 部分的其余章节多数都要涉及这一章的内容。在其他章节中，为了参考第 15 章的内容，会在页面边缘使用一个黑色标记，一眼便可以了解该章的位置。另外，还有一些帮助方式，比如在多数页面顶端有引导词，指示本页涉及的对象和对象特征。

第 4 部分　JavaScript 核心语言参考

第 4 部分中包括了 JavaScript 核心语言的参考内容。与第 3 部分的一些参考章节一样，本部分的多个章节中都显示了各个 JavaScript 语言条目的浏览器兼容性图表。页面顶端的引导词可以帮助读者快速找到特定的条目。

第 5 部分　附录

最后，本书的几个附录提供了有用的参考信息。这些资源包括：附录 A 中的 JavaScript 和浏览器对象快速参考；附录 B 中的 JavaScript 保留字列表；附录 C 提供了第 2 部分教程练习的答案；附录 D 列出了 Internet 资源。在附录 E 中，有使用本书附带光盘的有关内容，包括大量附赠章节和例子。

本书配套光盘

本书配套光盘就像一座丰富的信息金矿，提供了学习所需的所有内容。附赠章节包括：

- ✦ 动态 HTML、数据验证、插件程序和安全性。
- ✦ 开发和调试专业 Web 应用的技术。
- ✦ 10 个成熟完整而且实用的 JavaScript 应用程序。

光盘中有一个 Content 文件夹，在这里，可以找到 300 多个现成的 HTML 文档示例，它们是第 3 部分和第 4 部分文档对象和 JavaScript 词汇的示例。还包括所有附赠章节的例子。可以使用 JavaScript 浏览器来运行这些例子，但一定要在 Content 文件夹中使用 `index.html` 页，把它作为运行这些程序清单的路径。这里还提供了一些脱离上下文关系的小样例代码段，理解成熟的 HTML 文档（虽然它们可能非常简单）对于运用这些概念是至关重要的。笔者特意从本书第 2 部分忽略脚本程序清单，目的就是为了让读者自己输入这些脚本。相信可以从中学到许多东西，即使是模仿本书的清单，也会使你习惯于在文档中输入脚本。

本书第 13 章的程序清单文件为 `evaluator.html`。通过第 3 部分和第 4 部分的许多例子代码段，在交互平台的帮助下，可以尝试使用对象模型或语言特征，这个平台就是 The Evaluator，这是本书的特色之一。你可以看到实时的结果，并很快学会这些特征的工作方式。

学习 JavaScript 的必备知识

虽然本书不要求读者有大量的编程经验，但如果使用 HTML 创建过一些网页，就比较容易理解 JavaScript 如何与页面上的常用元素进行交互。有时，需要修改 HTML 标记语言来编写脚本。如果很熟悉那些标记，就可以轻松掌握 JavaScript 的增强功能。

幸运的是，读者无需知道服务器脚本编程，也不必了解如何从表单向服务器传递信息。这里的重点是客户端脚本编程，在增强 JavaScript 的 HTML 页面完全加载到浏览器之后，它的操作与服务器完全无关。

当前 HTML 标准的基本词汇应该是要掌握的基础知识的一部分。读者要熟悉一些最新的文档标记标准，比如 XHTML 和层叠样式表（Cascading Style Sheets，CSS）。但无论如何，都无需成为专家。在 Web 上搜索这些术语，会看到许多相关主题的教程。

假如以前从未编写过程序

不要被本书厚厚的篇幅吓倒。JavaScript 不是世界上最容易学习的语言，它和纯粹的编程语言也相距甚远，比如 Java 或 C。与开发功能完整的单一功能应用程序不同的是（比如在商店中购买的提高工作效率的程序），使用 JavaScript，可以通过编写小的程序代码段完成重要的任务。每个脚本浏览器中内置的 JavaScript 解释器为用户作了大量的技术工作。

从最基本的层面上说，编程主要就是编写一系列指令，让计算机按指令操作。我们的生活中无时无刻不在执行指令，即使是在无意识中。去朋友家的过程就是由一系列小指令构成的：走过 3 个街区，向左拐，再向右拐，就到了目的地。这些指令中，人们要作决策：如果交通指示灯为红色，就停止；如果是绿色，就行

进；如果是黄色，就把汽车加速器调至最低档。有时，必须几次重复一些操作，比如，不停地在该街区绕圈，直到找到停车位为止。计算机程序不仅包含主要的步骤序列，还预见到实现程序目标的过程中，要做什么决策或循环，比如，如何处理各种交通指示灯的情况，以及当某人抢走了停车位时该做什么。

学习编程最初的障碍是熟悉程序语言将文字和数字组织成指令的方式，和生活中的语言一样，这些规则叫做语法，就像我们说话使用的语言一样。因为计算机是不说话的电子机器，假如不使用它们能理解的特定语言与其通信，它们是不会理解的。当和另一个人说话时，如果用错句子的语法，别人还可理解，但计算机程序语言就不是这样。假如语法不正确（或至少在它可以纠正的语言范围内），计算机都会报告语法错误。

最好是将遇到的语法错误记下来作为学习经验，即使是有丰富经验的程序员也会犯语法错误。每遇到一个语法错误，每一次重写语句并解决错误，都会丰富自己的语言知识。

假如以前做过一些编程

有过去过程语言的编程经验，比如 BASIC，可能会是学习 JavaScript 的障碍，而不会有所帮助。虽然你已在语法上有精确的认识，但在程序如何适合于这个世界的整体概念方面，它们与 JavaScript 有本质的不同。一部分原因与脚本完成的典型任务有关（在网页内执行特殊的任务来响应用户动作），但更多的原因与面向对象编程的特性有关。

在典型的过程化程序中，程序员必须响应在屏幕上和后台发生的每件事情。程序首次运行时，大量的代码用于建立可视环境，也许屏幕包含几个文本输入域或可单击按钮。要确定用户单击哪个按钮，程序必须检测单击的坐标，并和屏幕上的所有按钮坐标比较，然后程序就执行对应于单击处的指令。

面向对象编程几乎和这个过程相反。一个按钮作为一个对象，是一个切实的东西。对象有属性，如标签、大小、对齐等。对象可能会包含脚本，同时，根据用户的行为，系统软件和浏览器一起把消息发送给对象用来触发脚本。比如，用户在文本输入域单击，系统/浏览器告诉该域某人进行了单击（使该域具有焦点），并让该域决定要做什么事情，这就是脚本开始之处。脚本连接到该域，包含一些指令，在用户激活域后执行这些指令。另一套指令控制当用户键入的内容、使用 Tab 键、在域外单击时所发生的事情，从而改变该域的内容。

人们编写的一些脚本可能在结构上类似过程化程序：它们包含按顺序执行的指令的简单列表。但当处理表单元素的数据时，这些指令使用 JavaScript 的面向对象特性。这个表单是一个对象；每个单选按钮或文本框也是对象，脚本作用于这些对象的属性以完成任务。

从过程化编程转到面向对象的编程是最困难的挑战。笔者在许多年前首次接触面向对象编程时，并没有立即领会它。但当茅塞顿开之后，很多事情变得豁然开朗。从那时起，面向对象几乎就成了笔者编程的唯一方法。

假如以前是 C 程序员

通过从 Java 借用的语法（依次从 C 和 C++派生而来），JavaScript 与 C 有许多相同的语法特点，熟悉 C 的程序员会觉得应用自如。操作符、条件结构和重复循环都遵循 C 的风格，在 JavaScript 可以不用像在 C 中那样关心数据类型。在 JavaScript 中，变量不局限于指某个特殊的数据类型。

有了这么多熟悉的 JavaScript 语法，这类程序员就可以把精力集中在文档对象模型的概念上，这对你来说可能是全新的。这里仍然需要具有 HTML（特别是表单元素）方面的良好基础，以发挥 JavaScript 的特长。

假如以前是 Java 程序员

尽管名字相似，但这两个语言只在表面相同：循环和条件结构、类似 C 的“点”对象引用、用来分组语句的花括号、几个关键字和其他一些属性。它们在变量声明方面的差异很大，因为 JavaScript 是弱类型语言。变量可在一个语句中包含整数值，在下一个语句中包含字符串（这不能算是良好的程序风格）。Java 所指的方法，在 JavaScript 中叫做方法（和预定义的对象关联在一起时）或函数（脚本开发者定义的动作）。JavaScript 方法和函数不用预先声明数据类型，就可返回任何类型的值。

在编写 JavaScript 时，不能使用某些 Java 的概念，最主要的是一些面向对象的概念，如类、继承、实例化和消息传递。这些方面在编写脚本时都不是问题，同时，JavaScript 设计者知道，你已经养成了一些难于克服的习惯。比如，虽然 JavaScript 不要求在每个语句行后加分号，但假如在 JavaScript 源程序中键入分号，JavaScript 解释器也认可。

假如以前写过脚本（或宏）

对掌握 JavaScript 概念而言，用其他创作工具或商业程序的宏编写脚本的经验非常有用，也许最重要的概念就是组合少量的语句，对一些数据执行特殊任务。比如，可在 Microsoft Excel 中写一个宏，主要用于完成日常图形的数据转换，而这些图形来自于另一个计算机上的财务报表。该宏被构建到 Macro 菜单中，当新的图形到来时，选择该菜单条目就可运行它。

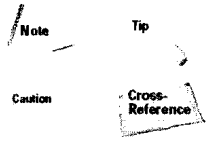
一些现代编程环境，比如 Visual Basic，与脚本环境类似。它们给程序员提供了一个界面编译器，显示用户能够交互的屏幕对象。程序员的主要工作是，编写一些代码，在用户与那些对象交互时，就执行这些代码。使用 JavaScript 的大量脚本编程工作就是采用这种模式。实际上，那些环境类似于脚本浏览器：它们提供了有限的预定义对象，而这些对象有确定的属性和行为集合。这使得学习和设计应用程序更加容易。

格式和命名约定

本书的脚本清单以等宽字体表示，这样可使它们与正文的其他部分区分开。由于页面宽度的限制，脚本清单可能会常常断行，在这些情况下，脚本的剩余部分就会在之后的行中出现，与清单的左边缘齐平，就像在打开自动换行功能的文本编辑器中一样。在文档中输入脚本清单时，假如这些断行引发了问题，建议在随书附带的光盘上找到相应的清单，查看输入脚本后应该是什么样子。

在本书的第 3 部分，在阅读对象模型或者语言功能（它们通常需要某个浏览器的指定最低级以上的版本）之前，不可能编写出多个页面。在需要特定的浏览器或浏览器版本时，要使其在文本中更易读。多数浏览器引用由缩写和版本号组成。比如：WinIE5 表示运行于 Windows 系统的 Internet Explorer 5；Netscape Navigator 4 表示运行于任何操作系统的 Netscape Navigator 4；Moz 表示现代 Mozilla 浏览器（从中派生了 Firefox、Netscape 6 或以后版本，以及 Camino）；Safari 表示 Apple 用于 Mac OS X 的自己的浏览器。如果浏览器的某个版本引入了一个功能，而且在后续版本中都支持，那么，就在这个版本号后面加一个“+”符号。例如，标记为 WinIE5.5+的功能，表示该功能至少需要 Windows 环境的 Internet Explorer 5.5，同时 WinIE7 和将来的 WinIE 版本也支持该功能。如果在现代浏览器的第 1 版中实现了某功能，那么，就在这个浏览器系列号的后面加上加号（+），比如 Moz+表示所有基于 Mozilla 的浏览器。有时，某功能或一些特殊行为只应用到一个浏览器。例如，某功能标记为 Netscape Navigator 4，表示只是在 Netscape Navigator 4.x 中有这个功能。减号（例如，WinIE-）表示浏览器不支持当前讨论的内容。

HTML 和代码清单都符合 XHTML 编码约定，即标记和属性名，以及不作为容器的自封闭（self-closing）的标记（例如，使用 XHTML `<br />`标记，而不是 HTML `<br>`标记）都使用小写形式。



这几个图标在本书中随处可见，用来标记重点内容或者告诉读者在哪里找到更详细的信息。

目　录

第 1 部分　JavaScript 起步

第 2 部分　JavaScript 教程

第 3 部分 文档对象参考