

WEB 专家工作室系列之四

Laura Lemay's Web Workshop JavaScript



CMP

(美) Michael Moncur 著

闻 芳 马昌超 刘景华 等译

JavaScript

实战解析

 机械工业出版社

 西蒙与舒斯特国际出版公司

本书介绍了 JavaScript 的程序设计技术。书中从最基本的 JavaScript 编程概念谈起,详细介绍了 JavaScript 中可使用的各种对象,讨论了如何用 JavaScript 操作 HTML 文档中的各个元素,从而获得更灵活、交互性更强的 Web 页,接着说明了多媒体的处理技术,最后介绍了 JavaScript 与 Java、CGI、SSI 等工具的结合技术以及 JavaScript 的未来。本书循序渐进,几乎覆盖了 JavaScript 编程的各个细节,并且包含大量新颖实用的示例,以及 JavaScript 中各种结构、对象和函数的详细说明,具有很好的参考价值。

本书叙述清晰,通俗易懂,使用方便,是使用 JavaScript 语言的用户极为有用的工具书,也是计算机应用人员和大中专院校师生必备的参考书。

Michael Moncur; Laura Lemay's Web Workshop JavaScript

Authorized translation from the English language edition published by Sams. net Publishing

Copyright 1996 by Sams. net Publishing

All rights reserved. For Sale in Mainland China only.

本书中文简体字版由机械工业出版社和美国西蒙与舒斯特国际出版公司合作出版,未经出版者书面许可,本书的任何部分不得以任何方式复制抄袭。

本书封面贴有 Prentice Hall 防伪标签,无标签者不得销售。

版权所有,翻印必究。

本书版权登记号:图字:01-97-0763

图书在版编目(CIP)数据

JavaScript 实战解析/(美) 芒克 (Moncur, M) 著; 闻芳等译. —北京:机械工业出版社, 1997. 7

(Web 专家工作室系列)

书名原文: Laura Lemay's Web Workshop JavaScript

ISBN 7-111-05827-5

I. J... I. ① 芒... ② 闻... II. JavaScript 语言-程序设计 N. TP312

中国版本图书馆 CIP 数据核字(97)第 13449 号

出 版 人: 马九荣(北京市百万庄南街 1 号 邮政编码 100037)

责任编辑: 刘小玲

北京第二外国语学院印刷厂印刷·新华书店北京发行所发行

1997 年 7 月第 1 版·第 1 次印刷

787mm×1092mm 1/16·18.5 印张·441 千字

0001-6000 册

定价:33.00 元

凡购本书,如有缺页、倒页、脱页,由本社发行部调换

译者序

JavaScript 是一种新的 Web 语言,也称为脚本语言。虽然该语言面市时间不长,但是由于它的特点,很快就得到了程序设计者的重视。用 JavaScript 编程非常容易,不需要任何开发工具或编译器。而且可以不经编译就可在浏览器上运行,源代码可以直接放到 Web 页中,故使广大的程序设计者爱不释手。

本书的侧重点是理论与实践相结合,循序渐进,在比较详细地介绍了 JavaScript 语言特点及其编程技巧的基础上,由浅入深地给出了大量的实例,从而使读者达到学以致用效果。本书取材新颖,内容丰富,是一本不可多得的学习 JavaScript 语言的好书。

参加本书翻译的还有谷晓鹏、汪再平、段健、赵志刚、杨浩、曾玉国、罗文林、易建设、刘湘、陈旺、金鑫、张忠荣和蒋才国等。

因译者水平有限,错误之处在所难免,敬请读者和同行批评指正。

译者

1997 年 5 月 4 日

引 言

World Wide Web 开始时像个简单的信息贮藏室,但它现在已发展了很多——它可以用于娱乐、教学、广告以及通信。在 Web 发展的同时,工具也有很大的发展。简单地记录工具(如 HTML)中嵌入了真正的编程语言——其中包括 JavaScript。

现在不要被“编程”吓住了。对很多人来说,这个词令人想到这样一幅景象:整晚盯着屏幕,试图记住哪些标点符号将产生我们需要的结果(有些人喜欢这种东西)。

尽管 JavaScript 是一种编程语言,但它很简单。事实上,如果以前没编过程序,它将是一个很好的入门。用 JavaScript 开始编程需要很少的知识——第 1 章“创建简单的 JavaScript 程序”中我们要将写下第 1 个程序。

如果读者以前用其他语言编过程序,就会发现 JavaScript 是非常有趣的——它固定的规则很少,句法很容易掌握,用 JavaScript 编程实在是一种乐趣,因为用 JavaScript 只要一两个小时就能写出真正有用的应用程序,而用其他语言,在看到程序结果前通常要花上几天的时间。

JavaScript 可以做很多事情来使 Web 页更生动,甚至可以创建多页的、完整的应用程序。本书中包括了所有这些内容——商业应用程序,还有像滚动消息这样的小玩意,甚至游戏。

当然,每种语言都有自己的局限,JavaScript 也不例外。本书中指出了它的缺陷所在,同时也介绍了一些可用来弥补这些缺陷的工具——CGI,SSI 和 Java。

我们知道 Web 是经常变化的,很难跟上它最新的特征。这本书将帮助我们吧 JavaScript 加到 Web 开发工具中,我们会很高兴掌握它的。

本书的读者对象

这本书是为那些想学习如何用 JavaScript 加强 Web 的人而作。不过这需要有下面几个条件:

- 熟悉 World Wide Web 和用 HTML 创建 Web 页的基本知识。
- 有 100%兼容 JavaScript 的浏览器——目前,这是指 Netscape 的最新版本。如果没有的话,可参看附录 C“在线 JavaScript 资源”,上面列出了一些可以下载 Netscape 的站点。
- 需要一台运行浏览器的计算机。Netscape 在 Windows, Windows 95, Macintosh 和 UNIX 上都是可用的。本书中的例子在上述平台中都可以工作。只有几个例子需要本书附带的 CD-ROM 中的其他文件。
- 要有一个文本编辑器——像 Windows 的 Notepad 这样简单的编辑器就可以工作得很好。
- 要看网上的例子,需要连到 Internet 上,还要有一个 Web 服务器提供商来显示页。然而本书中大多数例子在不上网的情况下也能工作得很好。

注意：尽管很容易混淆，但 JavaScript 与 Java 必竟是非常不同的语言。本书中尽管也介绍了 Java 以及如何把二者结合起来，但它主要还是写 JavaScript 的。

如何阅读本书

本书分为 6 个部分，大致是根据涉及问题的复杂性划分的：

- 第 1 部分“快速学习 JavaScript 编程”让我们很快开始 JavaScript 编程并介绍了几个基本概念。
- 第 2 部分“使用 JavaScript 对象和表格”介绍了对象和表格——建立 JavaScript 应用程序的基础。
- 第 3 部分“创建漂亮的 Web 页”介绍了一些更为复杂的 JavaScript 问题，主要集中在给 Web 页增加灵活的交互性特征。
- 第 4 部分“高级 JavaScript 概念”涉及更进一步的概念，包括图形、多媒体和查错。
- 第 5 部分“JavaScript 的替代产品及其前景”介绍了其他一些常用的 Web 语言以及它们如何在页中与 JavaScript 相结合。最后一章介绍了将来可能影响 JavaScript 编程的变化和其他问题。
- 第 6 部分包括 4 个附录。它给出了 JavaScript 精确的参考文献或在线资源的列表。

尽管这本书划分为几部分，可以一部分一部分读下去。但也可以不这样。第一章是要读者了解 JavaScript 的基本概念，但如果读者愿意，也可以将其跳过。在每一章末尾都有后续内容的列表，列出了可以继续学习的其他章节，不一定非按次序进行。

本书中有一些实例，展示了如何使用 JavaScript 的独特特性，并包含了一些实际应用程序。这些应用程序包括简单的小玩意、商业应用程序、游戏和图形。

关于软件的注意事项

JavaScript 像多数与 Web 有关的语言和产品一样，也在不断发展。随着不断普及，它也在不断地增加新的特性，不断地发生着变化。正是由于这一点，使用哪种浏览器将会影响到程序能否正常工作。

本书中的例子是用 Netscape 3.0 开发的，并在 3.0 最终的版本中得到检验。尽管要确保它必须在老版本的浏览器中工作，但我们还是建议读者下载一个最新版本的 Netscape 来确保每个脚本都能正常工作。

目 录

译者序

引言

第 1 部分 快速学习 JavaScript 编程

第 1 章 创建简单的 JavaScript 程

序 1

1.1 什么是 JavaScript 1

1.2 JavaScript 与 Java 3

1.2.1 实例:将 JavaScript 与 HTML
结合到一起 3

1.2.2 简化的语言结构 4

1.2.3 Web 浏览器集成 4

1.2.4 支持平台 5

1.3 JavaScript 的用途 5

1.3.1 包含动态信息 5

1.3.2 使表格更有效 6

1.3.3 使页交互性更好 6

1.4 开始使用 JavaScript 6

1.4.1 所需软硬件 6

1.4.2 开发工具 6

1.4.3 实例:测试一个简单的
JavaScript 程序 7

1.4.4 接收 Web 页 7

1.4.5 执行脚本 7

1.5 JavaScript 中潜在的问题 8

1.5.1 有关安全性的考虑 8

1.5.2 用户源代码是可被访问的 8

1.6 克服 JavaScript 的局限 9

1.6.1 未能具备的特性 9

1.6.2 将 JavaScript 与其他语言相
结合 10

1.7 编写一个简单的 JavaScript 应用
程序 11

1.7.1 创建一个脚本 11

1.7.2 把脚本嵌入 HTML 页 12

1.7.3 实例:创建一个事件句柄 12

1.7.4 查看脚本输出 13

1.7.5 实例:在老版本的浏览器中
隐藏 JavaScript 13

1.8 本章小结 14

1.8.1 后续内容 14

1.8.2 问题与解答 14

第 2 章 使用较大的程序和变量 16

2.1 JavaScript 语言结构 16

2.1.1 语句 17

2.1.2 函数 17

2.1.3 变量 18

2.1.4 表达式 18

2.1.5 对象、属性和方法 18

2.1.6 实例:使用注释 18

2.2 程序与应用程序 19

2.3 数据类型和变量 19

2.3.1 整型数 20

2.3.2 浮点数 20

2.3.3 布尔值 21

2.3.4 字符串 21

2.3.5 特殊字符 21

2.3.6 实例:创建一个数组 22

2.3.7 实例:使用数字和文本 23

2.4 给变量命名和声明变量 23

2.4.1 JavaScript 变量命名规则 23

2.4.2 实例:给变量赋值 24

2.5 变量声明和使用范围 24

2.6 使用表达式和运算符 26

2.6.1 赋值运算符 26

2.6.2 算术运算符 27

2.6.3 实例:变量递增和递减 28

2.6.4 字符串运算符 28

2.6.5 逻辑运算符 29

2.6.6 位运算符 29

2.6.7 比较运算符	29	3.5.3 事件句柄是如何响应的	52
2.6.8 理解运算符优先级	30	3.6 本章小结	52
2.6.9 实例:在表达式中使用变量	30	3.6.1 后续内容	52
2.7 转换、计算变量值和表达式	31	3.6.2 问题与解答	53
2.7.1 parseInt 函数	31	第4章 使用内置对象和用户自定义对象	54
2.7.2 parseFloat 函数	31	4.1 面向对象编程的技术	54
2.7.3 eval 函数	32	4.1.1 实例:使用对象、属性和方法	54
2.8 本章小结	32	4.1.2 关键字 new	56
2.8.1 后续内容	32	4.1.3 关键字 this	56
2.8.2 问题与解答	32	4.1.4 关键字 with	56
第2部分 使用 JavaScript 对象和表格		4.1.5 创建一个一般对象	57
第3章 使用对象和事件	35	4.1.6 实例:利用对象属性进行循环	57
3.1 使用 JavaScript 对象	35	4.2 使用内置对象	57
3.1.1 实例:使用对象属性	36	4.2.1 使用 Array 对象	58
3.1.2 给属性赋值	36	4.2.2 实例:创建一个数组	58
3.2 函数和方法	37	4.2.3 使用 String 对象	59
3.2.1 实例:声明一个函数	37	4.2.4 使用 Date 对象	62
3.2.2 调用一个函数	38	4.2.5 Math 对象	63
3.2.3 返回一个值	39	4.2.6 navigator 对象	64
3.2.4 将函数与 HTML 集成在一起	39	4.3 定制对象	65
3.2.5 把任务划分为几个函数	41	4.4 本章小结	66
3.2.6 函数间的通信	41	4.4.1 后续内容	66
3.3 理解方法	42	4.4.2 问题和解答	66
3.3.1 实例:定义对象和方法	42	第5章 以对象的形式访问窗口元素	68
3.3.2 实例:创建对象实例	43	5.1 window 对象	69
3.3.3 把它放在一起	44	5.1.1 window 对象属性	69
3.4 条件和循环	45	5.1.2 实例:改变状态行	69
3.4.1 if...else 结构	45	5.1.3 window 对象方法	71
3.4.2 条件表达式	46	5.1.4 实例:打开和关闭窗口	72
3.4.3 实例:使用关键字 for	47	5.1.5 实例:使用提示、警告和确认对话框	73
3.4.4 实例:使用 while 循环	48	5.1.6 实例:用定时器更新页	75
3.4.5 for...in 结构	48	5.1.7 实例:使用窗口对象事件句柄	77
3.4.6 无限循环	49	5.2 location 对象	78
3.4.7 break 语句	49	5.3 document 对象	79
3.4.8 continue 语句	50	5.3.1 document 对象属性	79
3.4.9 实例:在函数中使用各种参数	50	5.3.2 document 对象方法	80
3.5 事件和事件句柄	50	5.4 history 对象	81
3.5.1 事件类型	51		
3.5.2 实例:创建一个事件句柄	51		

5.5 link 对象	82
5.6 anchor 对象	83
5.7 form 对象	83
5.8 本章小结	84
5.8.1 后续内容	84
5.8.2 问题与解答	84
第 6 章 使用交互式表格	85
6.1 创建一个兼容的 JavaScript 的 HTML 表格	85
6.1.1 理解表格的响应和方法	85
6.1.2 浏览表格元素	86
6.1.3 实例:创建一个 HTML 表 格	89
6.2 在 JavaScript 中使用 form 对象	91
6.2.1 form 对象属性	91
6.2.2 form 对象方法和事件	92
6.3 JavaScript form 对象系统	92
6.3.1 文本域和密码域	93
6.3.2 文本区	93
6.3.3 复选框	94
6.3.4 单选按钮	94
6.3.5 选择列表	95
6.3.6 隐藏域	96
6.3.7 按钮	96
6.3.8 文件上载域	96
6.4 用 JavaScript 自动控制表格	96
6.4.1 实例:增加自动计算总和的 功能	96
6.4.2 实例:自动控制送货地址	99
6.5 用事件句柄检查表格数据的合 法性	102
6.5.1 在哪儿检查确认	102
6.5.2 检查确认哪个域	103
6.5.3 实例:创建检查确认函数	103
6.5.4 实例:增加一个用于检查确认 的事件句柄	104
6.6 本章小结	107
6.6.1 后续内容	108
6.6.2 问题与解答	108
第 7 章 实例剖析 I	110
7.1 例 1:显示弹出消息	110
7.2 例 2:显示随机内容	112

7.3 例 3:可检查所填内容合法性的 登记表	114
----------------------------------	-----

第 3 部分 创建漂亮的 Web 页

第 8 章 用 JavaScript 改善 Web 页	117
8.1 开始:一个简单的 Web 页	117
8.2 创建导航条	119
8.2.1 给页命名	119
8.2.2 实例:创建导航条的数据结 构和 HTML	120
8.2.3 实例:创建导航条的函数	120
8.2.4 实例:把导航条加入 HTML 页	121
8.3 使用状态行	123
8.3.1 实例:创建友好的链接	123
8.3.2 实例:加入滚动的消息	125
8.3.3 实例:用文本框显示消息	128
8.4 本章小结	131
8.4.1 后续内容	131
8.4.2 问题与解答	132
第 9 章 使用框架,记忆块和其他 高级特性	133
9.1 高级浏览器特性	133
9.1.1 框架	133
9.1.2 目标链接	137
9.1.3 记忆块	138
9.1.4 Plug-Ins	139
9.2 在 JavaScript 中使用框架	139
9.2.1 在框架中使用的对象	139
9.2.2 在多框架窗口中使用 JavaScript	140
9.2.3 实例:创建框架文件	140
9.2.4 实例:修改导航条	141
9.2.5 实例:测试多窗口文件	142
9.2.6 实例:用 JavaScript 更新框 架	143
9.2.7 实例:用记忆块记忆用户的 偏好	145
9.3 本章小结	146
9.3.1 后续内容	147
9.3.2 问题与解答	147

第 10 章 使用多页面和大量数据	148	13.3.4 实例:测试插件	187
10.1 创建复杂的 Web 站点	148	13.4 本章小结	188
10.2 使用数据和数据库	150	13.4.1 后续内容	188
10.2.1 使用字符串数组	150	13.4.2 问题与解答	188
10.2.2 使用关联数组	151	第 14 章 调试 JavaScript 程序	190
10.2.3 创建用户的数据结构	151	14.1 JavaScript 编程中的常见错误	190
10.3 理解数据标记技术	152	14.1.1 语法错误	190
10.3.1 激活数据标记	153	14.1.2 赋值(=)与相等(==)混淆	190
10.3.2 数据标记命令	153	14.1.3 错误给对象命名	191
10.3.3 实例:使用多台服务器上的数据	153	14.1.4 错误使用标点和大写符号	192
10.4 在 JavaScript 中维护状态信息	155	14.1.5 常见的 HTML 错误	192
10.5 本章小结	157	14.1.6 错误使用对象	192
10.5.1 后续内容	157	14.1.7 时序错误	193
10.5.2 问题与解答	158	14.1.8 变量问题	193
第 11 章 实例剖析 I	159	14.1.9 保留字带来的问题	194
11.1 例 1:不滚动的状态行消息	159	14.1.10 平台问题	194
11.2 例 2:改善导航条	162	14.2 调试技术	194
11.3 例 3:保存用户偏好信息	165	14.2.1 实例:用 Alert 显示值	194
第 4 部分 高级 JavaScript 概念		14.2.2 实例:用临时窗口调试	195
第 12 章 在 JavaScript 中使用图形	169	14.3 调试 JavaScript 的工具	196
12.1 在 JavaScript 中使用动态图像	171	14.3.1 使用 JavaScript 命令行	196
12.1.1 images 数组	171	14.3.2 实例:交互式地测试 JavaScript 命令	197
12.1.2 预载入图像	171	14.3.3 在 URL 中使用命令行	198
12.1.3 实例:动态显示图像	172	14.3.4 JavaScript Trace Facility (JavaScript 跟踪设备)	199
12.1.4 实例:创建图形化的时钟	174	14.3.5 实例:安装 JavaScript Trce Facility	199
12.1.5 实例:轮换显示广告旗标	176	14.3.6 使用调试表格	199
12.2 JavaScript 和客户图像映像	177	14.3.7 使用跟踪函数	200
12.3 本章小结	179	14.4 处理缺陷和死机	200
12.3.1 后续内容	180	14.4.1 Netscape Navigator 2.0	200
12.3.2 问题与解答	180	14.4.2 Netscape Navigator 2.01	201
第 13 章 使用多媒体和插件	181	14.4.3 Netscape Navigator 2.02	202
13.1 在 JavaScript 中使用声音	181	14.4.4 Netscape Navigator 3.0 (Atlas)	202
13.1.1 配置声音播放器	182	14.4.5 Microsoft Internet Explorer	202
13.1.2 实例:用事件播放声音	182	14.5 本章小结	203
13.2 Netscape 插件概述	183	14.5.1 后续内容	203
13.3 插件和 JavaScript	184	14.5.2 问题与解答	203
13.3.1 与插件相关的对象	184		
13.3.2 实例:列出插件	185		
13.3.3 实例:列出 MIME 类型	186		

第 15 章 实例剖析	204
15.1 程序规划	204
15.2 完整的应用程序	205
15.3 程序是怎样工作的	211
15.3.1 游戏初始化	211
15.3.2 纸牌的放置	212
15.3.3 游戏结果计分	213
15.3.4 结束一个游戏	215

第 5 部分 JavaScript 的替代产品 及其前景

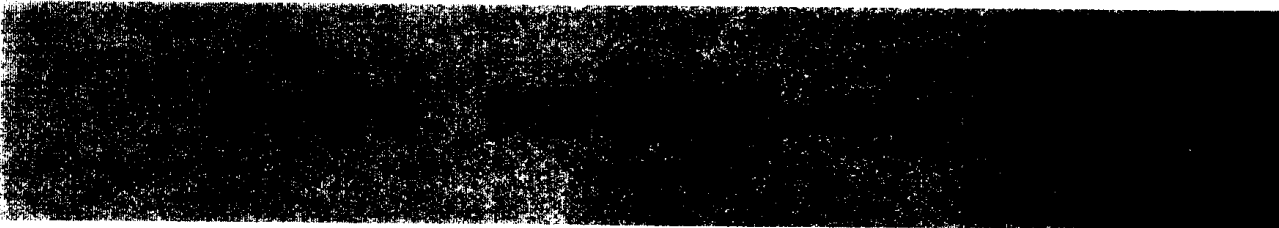
第 16 章 JavaScript 和 Java 的集成	217
16.1 Java 概览	217
16.1.1 Java 是如何工作的	218
16.1.2 Java 语言	218
16.1.3 Java 对象和类	218
16.1.4 Java 和 HTML 的集成	218
16.1.5 生成自己的 Java 小应用程序	219
16.1.6 实例:安装 Java 开发工具	219
16.1.7 实例:生成一个简单的 Java 小应用程序	220
16.1.8 实例:编译和演示一个 Java 小应用程序	220
16.2 在 JavaScript 中使用 Java 类	222
16.2.1 调动 Java 方法	223
16.2.2 applet 对象	223
16.2.3 使 Java 小应用程序可访问	223
16.2.4 实例:控制 Java 小应用程序	223
16.3 由 Java 调用 JavaScript 函数	225
16.3.1 JavaScript 程序员的操作	225
16.3.2 Java 程序员的操作	225
16.3.3 访问 JavaScript 对象	226
16.3.4 调用 JavaScript 函数	226
16.4 本章小结	226
16.4.1 后续内容	226
16.4.2 问题与解答	226

第 17 章 JavaScript、CGI 和 SSI 的结合	228
17.1 为工作选择恰当的工具	228
17.2 CGI 基础	230
17.2.1 获取和传递请求	231
17.2.2 用表格形式定义请求	231
17.2.3 名称和数值对	231
17.2.4 使用 URL 编码和解码	231
17.2.5 环境变量	232
17.2.6 CGI 程序怎样产生输出	232
17.3 SSI:在 Web 页中的 CGI	233
17.3.1 SSI 指令	233
17.3.2 使 SSI 可用	233
17.4 CGI 和 SSI 可以使用的语言	233
17.4.1 Perl 语言	234
17.4.2 C 和 C++ 语言	234
17.4.3 其他语言	234
17.4.4 实例:创建一个简单的 CGI	234
17.4.5 实例:安装 CGI 脚本	235
17.4.6 实例:用 SSI 创建 JavaScript 函数	235
17.5 本章小结	236
17.5.1 后续内容	236
17.5.2 问题与解答	236
第 18 章 使用 Active X 和 Microsoft Internet Explorer 3.0	238
18.1 Internet Explorer 和 JavaScript	239
18.2 使用 Active X 控制	240
18.2.1 Active X 控制是如何工作的	241
18.2.2 实例:使用已存在的 Active X 控制	241
18.2.3 一个 Active X 控制和特性的例子	242
18.2.4 实例:使用 Active X 控制台	243
18.2.5 开发 Active X 控制	244
18.3 介绍 VBScript	245

18.4	VBScript 与 JavaScript	246	20.5	JavaScript 的替代产品	258
18.4.1	标点和符号语法	246	20.5.1	Java	258
18.4.2	变量	246	20.5.2	CGI	258
18.4.3	VBScript 和表格	247	20.5.3	SSI	259
18.4.4	面向对象性质	247	20.5.4	LiveWire:用于服务器的 JavaScript	259
18.4.5	Active X 和 VBScript	247	20.5.5	Shockwave	259
18.5	本章小结	247	20.5.6	Active X	259
18.5.1	后续内容	247	20.5.7	VBScript	260
18.5.2	问题与解答	247	20.5.8	HTML 的增强功能	260
第 19 章	实例剖析 IV	249	20.6	本章小结	260
19.1	例 1:控制一个 Java 小应用 程序	249	20.6.1	后续内容	260
19.2	例 2:动态创建 JavaScript	252	20.6.2	问题与解答	261
19.3	例 3:使用 Active X 控制	254	20.6.3	结束语	261
第 20 章	JavaScript 前景	256			
20.1	计划中的新性质	256			
20.2	JavaScript 的希望列表	257			
20.2.1	语言特性	257			
20.2.2	交互表格	257			
20.2.3	图形和多媒体	257			
20.2.4	用户界面和框架	257			
20.2.5	Java 和插件	257			
20.3	JavaScript 的新应用程序	257			
20.4	JavaScript 开发环境	258			

第 6 部分 附 录

附录 A	JavaScript 结构和对象参 考	263
附录 B	JavaScript 语句、函数、操 作符和关键词参考	270
附录 C	在线 JavaScript 资源	275
附录 D	CD-ROM 上的内容	282



第 1 章 创建简单的 JavaScript 程序

本章要点：

- 了解 JavaScript 语言的基础知识。
- 了解 JavaScript 和 Java 的区别。
- 创建一些简单的 JavaScript 程序并把它们与 Web 页结合到一起。

本章实例：

- 将 JavaScript 与 HTML 结合到一起。
- 测试一个简单的 JavaScript 程序。
- 创建一个事件句柄。
- 在老版本的浏览器中隐藏 JavaScript。

我们即将开始一次深入 JavaScript 的旅行。JavaScript 是一种新兴的开发 Web 页的热门语言，它使得 Web 页具有更好的交互性，更加灵活，也可以给枯燥的沉闷的页增加几分活力和灵气。

对于那些与 HTML（一种 Web 语言）打过交道并且对编程有一些基本了解的读者来说，理解 JavaScript 应该不会有任何困难。

JavaScript 是一种简单而又灵活的语言。这一章将介绍 JavaScript 的历史、特性以及它的局限性。

1.1 什么是 JavaScript

要准确地解释什么是 JavaScript 必须从 Java 说起。Java 是由 Sun Microsystems 公司开发的一种新的 Web 编程语言。一个 Java 程序或者一个小应用程序可以由 HTML 页加载并且被 Java 解释器执行，而 Java 解释器是嵌入在浏览器之中的。

Java 是一种复杂的语言，和 C++ 很相似。它是面向对象的语言，并且有着强大的功能。其缺点是弄清楚它需要一些时间，并且需要较长的开发周期，JavaScript 正是由此产生的。

JavaScript 是一种新的 Web 语言，被称为脚本语言。脚本语言是一些简单的语言，可以用来给沉闷枯燥的 Web 页增加一些额外的特性。Java 主要是为程序员设计的，相比之下脚本语言使非程序员改进 Web 页变得更加容易。

JavaScript 最初是由 Netscape 公司开发的,其目的是为了在它开发的浏览器 Netscape Navigator 中使用。它具有便于使用的句法、灵活的变量类型及易于访问浏览器的某些特性。它可以不经编译就在浏览器上运行,源代码可以直接放到 Web 页中。

用 JavaScript 编程非常容易,不需要任何开发工具或编译器。用户可以使用创建 HTML 文档的编辑器来创建 JavaScript 程序,并可以直接在浏览器(目前指 Netscape 或 Microsoft Internet Explorer)中执行它。

JavaScript 最初叫做 LiveScript,是 Netscape 浏览器独有的工具。现在,JavaScript 已被 Java 的开发者—Sun 公司认可。作为一种脚本语言,JavaScript 成为对 Java 的一种补充。其他几个公司也已宣布支持 JavaScript。

尽管 Java 在使用上非常有效,读者可能会发现 JavaScript 有自己的独到之处。它可以直接在 Web 页中处理某些 Java 不能处理的 HTML 元素。它也非常简单易用,用户可能只需要几条 JavaScript 语句就可以实现一些功能。在这本书中,读者将会看到一些例子,它们显示了 JavaScript 的强大功能。

注意: JavaScript 仍在不断发展之中,在 JavaScript 成为最终版本之前可能仍然会有一些变动。如果读者要了解最新的变动,请访问 Netscape 公司的 JavaScript Web 主页,地址如下:<http://home.netscape.com/comprod/products/navigator/version-3.0/script/>

JavaScript 的历史

正如前文所示,JavaScript 的历史是从 Java 开始的。Java 最初由 Sun Microsystems 公司为了在实时嵌入式系统中使用而开发的。

Java 现在事实上已经成为高级 Internet 编程的标准,但我们仍然可能看到它有时在运行自己的蜂窝电话。

Java 被设计运行于虚拟机上。虚拟机是一种软件,它可以解释 Java 代码并响应这些代码,本质上就好像一台计算机。虚拟机被设计得十分简单,这样它可以很容易地在某种设备中被实现。这就使它在 Web 浏览器中很容易实现。

注意: 有几家公司,包括 Oracle 公司和 Apple 公司,已经提出了网络计算机的设计方案。网络计算机是对 PC 的一种简化,它被设计用于网上工作,并从网络上获得所需要的软件。大多数网络计算机都将使用 Java 或它的某种变体作为自己的操作系统。

最初只有 HotJava 支持 Java,而 HotJava 也正是 Sun 公司为支持 Java 而开发的一种实验性 Web 浏览器。Netscape 在认识到 Java 的巨大潜力后,把它和自己开发的 Web 浏览器——Netscape Navigator 结合到一起。由于 Navigator 是目前最流行的浏览器,故 Java 得到了很大的普及。

1995 年,Java 成为 Internet 上最热门的新词,但很少有人真正了解如何用它编程。

Netscape Communications 认识到人们需要有一种用于 Web 页的简单明了的语言,于是引入了 LiveScript,这是第一个 Web 脚本语言。

LiveScript 的句法是以 Java 为基础的,但是更加简明扼要,也更易于掌握。同时,LiveScript 是一种直接解释型语言,而 Java 则必须经过编译。Netscape 公司将 LiveScript 嵌入 Netscape Navigator 的 β 版中。1995 年 6 月发布的 Netscape Navigator 2.0b1 版开始支持 LiveScript。

1995 年的早些时候,Netscape 公司与 Sun 公司达成协议。Sun 公司也认识到使用一种简单的脚本语言是一个好主意,于是他们正式认可了 LiveScript。这样,LiveScript 与 Java 的联系成为正式的,而 LiveScript 也改名为我们都很熟悉的 JavaScript。

在写这本书时,JavaScript 仍在开发之中并且不断地被完善。Netscape 在 1996 年底将 JavaScript 最终定稿。其他一些公司——例如最著名的 Microsoft 公司——正在加紧发布与其相竞争的产品。Microsoft Internet Explorer (MSIE)3.0(目前是 β 版)支持基本的 JavaScript 以及 VBScript,VBScript 正是 Microsoft 公司对 JavaScript 的响应。

1.2 JavaScript 与 Java

编写一个 Java 小应用程序的过程可能是非常复杂的——要写源代码、编译,还要使它在服务器上可用。JavaScript 为一些小项目提供了更为简单的方案。JavaScript 的源代码是直接由浏览器解释的。源代码可以直接嵌入 HTML 页中或放在一个单独的文件里。

尽管 JavaScript 并没有什么可以使 Java 黯然失色的特性,但它增加了一些便利之处,简化了编程,并且与浏览器结合得比较好。JavaScript 与 Java 主要的不同之处如下所示,这将在以后的章节中陆续加以解释。

- JavaScript 可以直接与 HTML 结合到一起。
- JavaScript 的语言结构比 Java 简单。
- JavaScript 的解释器是内置在 Web 浏览器中的。
- 可支持 JavaScript 的平台要多于支持 Java 的平台。

1.2.1 实例:将 JavaScript 与 HTML 结合到一起

Java 小应用程序是以字节码的方式被编译和存储在服务器上,而 JavaScript 程序是简单的 ASCII 文本文件。用户可以把它们保存在独立的文件中或直接把 JavaScript 函数嵌入 HTML 页中。

`<SCRIPT>` 标记是 Netscape 支持的 HTML 的扩展标记。使用 `<SCRIPT>` 标记使用户可以在页中嵌入一个或多个 JavaScript 函数。程序清单 1-1 是一个直接嵌入 HTML 中的非常小的 JavaScript 脚本。在这一章的后面部分将详细介绍这些标记的句法。

程序清单 1-1 (SIMPLE.HTM)HTML 文档中的一个小 JavaScript 程序

```
<HTML><HEAD>
<TITLE>Simple JavaScript Example
</HEAD>
<BODY>
HTML Text goes here.
<SCRIPT LANGUAGE="JavaScript">
document.write("Here is my output.")
</SCRIPT>
</BODY></HTML>
```

另一种调用函数的方式叫做事件句柄,它可以让用户指定当某个事件发生时 JavaScript 做出何种响应。例如,当页上的某个按钮被按下时可能会做出某种反应。这就为使用 JavaScript 提供了动态性更强的方式。

与<SCRIPT>标记不同,事件句柄是作为一种属性附加给 HTML 标记的。例如,下面的 HTML 码定义了一个与事件句柄的链接。

```
<A HREF="www.netscape.com" onClick="alert('This will take you to Netscape's home
page.');">
```

在本例中,事件名是 onClick。当用户单击这一链接时,onClick 事件就会发生。单击链接时要执行的 JavaScript 代码被括在双引号中。

1.2.2 简化的语言结构

JavaScript 的局限也使它对于编程者来说更加容易。JavaScript 语言的句法比 Java 的句法更加宽松,变量(用以存储数据的标记的名字,更易于使用。具体如下:

- JavaScript 语言是解释型的而非编译型的。改变一个脚本像改变一个 HTML 页一样简单。
- 用户仅仅通过访问 JavaScript 中已经存在的对象就可以完成一定的功能,而不需创建对象成类。
- 采用松散型变量。用户不需要在使用前声明变量类型,并且多数变量类型间的转换(像由数字型变量转换为字符串型变量)是自动进行的。
- 事件句柄使 JavaScript 函数在 HTML 文档的一部分有某些动作发生时被执行。例如,只要表格 form 的输入域被改变就检查输入域以确定输入值。

1.2.3 Web 浏览器集成

JavaScript 是一种面向对象的语言,这意味着它可以使用对象。对象是一种较为复杂的变量形式,它可以存储多个值也可以包含 JavaScript 代码,用户可以创建自己的对象来表示任何事物。

JavaScript 中也有一些对象可以使用户直接访问浏览器的某些特性。这些对象代表了浏览器和 Web 页中的某些元素,像窗口、文档、框架、表格、链接和链接点。

用户可以得到当前页的链接、链接点以及表格元素的有关信息,也可以通过 JavaScript 来控制浏览器。例如,可以在页上自己设置一个“back”按钮,按下这个按钮可以让用户返回到前一页,就像浏览器上自带的“←”的按钮。

这里所说的浏览器是指支持 JavaScript 的浏览器。Netscape Navigator 从 2.0b1 版本开始支持 JavaScript,现在,Netscape Navigator 的最新版本已经到了 3.0b6。Microsoft Internet Explorer 也将支持 JavaScript。

1.2.4 支持平台

由于 Netscape 在所有可用平台上均支持 JavaScript,因而 JavaScript 比 Java 获得了更广泛的支持。在 Java 得到更广泛的支持前,使用 JavaScript 编写一些简单的应用程序可以使我们在各种不同的平台上拥有更多的观众。到目前为止,完全支持 JavaScript 的浏览器只有 Netscape Navigator。下面这些平台均可提供对 JavaScript 的支持。

- Windows 3.1,使用 16 位版本的 Navigator。
- Windows 95 或 Windows NT,使用 32 位版本的 Navigator。
- Macintosh System 7 或更高版本,使用 Macintosh 版本。
- 大多数版本的 UNIX,使用 Navigator 的 X Window 版本。

注意: Microsoft 公司现在已经发行了它所开发的浏览器的 β 版本,即 Microsoft Internet Explorer 3.0。这一浏览器也支持 JavaScript,但仍有待进一步发展,目前在运行某些 JavaScript 程序时可能会碰到一些问题。

尽管 JavaScript 获得了广泛的支持,也可以确保 90% 的 Web 的访问者使用用 JavaScript 开发的脚本,但是不要忘记其余 10% 的访问者。要尽可能为不支持 JavaScript 的浏览器提供一个替代品。

警告: 尽管上面介绍的那些平台均支持 JavaScript,但是到目前为止,只有 Navigator 的 β 版完全支持 JavaScript。这就意味着在使用 JavaScript 时很容易出错,本书第 14 章“调试 JavaScript 程序”中将提供进一步的详细信息。

1.3 JavaScript 的用途

我们已经了解了关于 Java 为什么非常重要的一些技术细节,但是它究竟好在什么地方?我们能用它为我们的 Web 页做些什么?下面这一部分将介绍 JavaScript 的一些最重要的用途。

1.3.1 包含动态信息

JavaScript 可以使一些文本成为动态的,从而给 Web 页增添几分生机。举个很简单的例子,我们可以在页上根据时间显示一句恰当的问候——“早上好”或者“晚上好”。由于我们可以使用和控制 HTML,因而可以通过动态选择一些图画或选择使用 JavaScript 的链接而将上面的这个例子再加以扩展。这样,我们就可以根据使用者、时间或其他任何因素的不同展现出恰当的页。

1.3.2 使表格更有效

读者在很多 Web 页上都看到过表格,可能自己也使用过表格。用户在表格内输入数据,然后按下“Submit”按钮,于是服务器使用 CGI 程序响应这些信息。这是非常有效的,但交互性并不很好。用户在收到响应前,不得不等待数据被传送到服务器再传回来。

本章第 6 节: 使用 JavaScript 使表格更有效

注意: JavaScript 与 CGI 最大的区别在于 CGI 运行在服务器上,而 JavaScript 运行在用户自己的机器上(客户机上)。对于把数据传送给服务器再把服务器数据传回给用户,CGI 是最有效的。

JavaScript 可以对 Web 表格立刻检查输入的合法性。通常在“Submit”按钮没有被按下之前,用户仍然不能把数据发往服务器,但 JavaScript 可立即响应这些数据。例如,用户可以在网上填一张贷款申请表,只要一填完,用户马上可以得到关于所需付款方式的反馈。用户对申请表满意之后再将它提交服务器。

1.3.3 使页交互性更好

最后,JavaScript 可以给用户一些页的控制权,从而从普通页中去除了一些费力不讨好的东西。例如,我们可以设置一个背景 on/off 按钮来控制背景显示或设置一个按钮来触发表格内容的显示。

本章第 7 节: 使用 JavaScript 使页交互性更好

1.4 开始使用 JavaScript

开发 JavaScript 应用程序过程是非常简单的。下面几部分将介绍一些创建和执行 JavaScript 脚本所需要的工具,并对如何将一个脚本与 HTML 页结合到一起的过程加以讲解。最后,读者可以了解一些现在支持 JavaScript 的软硬件平台。

1.4.1 所需软硬件

为使用 JavaScript,目前我们唯一的需要(同时也是唯一的选择)就是 Netscape Navigator 2.0b 或更高版本。可用的硬件平台我们将在这一部分的后面加以讨论。然而,其他几家浏览器的制造商也计划在不久的将来支持 JavaScript。

1.4.2 开发工具

假如用户已经有了一个 Netscape 来观赏自己的作品,那么开发 JavaScript 脚本不再需要其他任何特别的工具。要创造这些脚本只需要一个文本编辑器或者一个 HTML 编辑器,就用我们用来开发 HTML 页的那个编辑器就可以。

注意: 迄今为止,没有哪个专用的 HTML 编辑器宣布支持 JavaScript,但我们仍然可以使用它们来创建 JavaScript 程序。本书附带的 CD-ROM 盘中就有几个有用的 HTML