高级应用与实践

JavaScript

Broadview
www.broadview.com.cn

本书是具有十多年专业**软件开发**、
架构设计经验的**夏天**对JavaScript的真知灼见！

JavaScript 高级应用与实践

夏　天　编著
涂传滨　审校

- 一样的**JavaScript**，不一样的深入解析，
使**Web**的开发如同基于XPath的文档开发一样便捷！
- 光盘赠送300MB本书所有例程代码。

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

JavaScript
高级应用与实践

夏 天 编著
涂传滨 审校

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

本书的内容涵盖大量提升 JavaScript 性能的一些技巧、鲜为人知的写法，以及 JavaScript 在 Web 项目中某些智能的、经典的设计。本书在讲解过程中会有大量的例程和各种写法运行耗时的性能比较，给你展示了如何优化 JavaScript 的性能、一些非常独特而又能提升 JavaScript 性能的技巧（涵盖了当前网络中流行的 JavaScript 框架的大多数技巧）以及每天海量交易项目的 JavaScript 相关经验。其中，很多经验性的思想描述和方法可以应用于各种语言的编程。另外，本书还阐述了大量的 JavaScript 泛型设计，以及 JavaScript 在 JSON-RPC 和 WebDAV 中的应用并给出了相关的 Java 设计。

最后本书的所有例程代码将组合为一套功能强大的基于 CSS、XPath 选择器模型的 JavaScript 框架，使 Web 的开发就如同基于 XPath 的文档开发一样便捷。

本书适合于不同层次的 JavaScript 语言爱好者和技术人员学习和参考。

图书在版编目（CIP）数据

JavaScript 高级应用与实践 / 夏天编著. —北京：电子工业出版社，2008.3
ISBN 978-7-121-06123-3

Ⅰ. J… Ⅱ. 夏… Ⅲ. JAVA 语言－程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2008）第 025879 号

责任编辑：顾慧芳
印　　刷：北京东光印刷厂
装　　订：三河市皇庄路通装订厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：787×1092　1/16　印张：29.5　字数：770 千字
印　　次：2008 年 3 月第 1 次印刷
印　　数：5000 册　定价：59.80 元（含光盘 1 张）

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

自从 web2.0 的诞生以来，JavaScript 一直炙手可热，如火如荼，加之 AJAX 的大行其道，时至今日，JavaScript 技术已被推到极致。它被应用在各种 B/S 架构的项目中、网站开发中，甚至一些嵌入式设备中。

JavaScript 是一个很大的话题，本书不可能详细介绍其中的所有细节，例如 JavaScript 的 ready、闭包、OOP 等概念和原理。

本书的适用对象

本书面向所有希望了解和掌握 JavaScript 编程的开发人员。只要对 JavaScript 感兴趣，无论是做 JavaScript 嵌入式的开发人员，还是做.Net、Java 的开发人员，还是已有一定的工作经验的读者，都能从书中找到自己所需要的内容。

本书的程序都是基于 JavaScript XPath 和泛型思想实现的，因此对那些想提高自己 JavaScript 优化水平和编程技巧的读者来说，也是一本很好的参考书。同时本书也适合作为 FireFox 和 IE 编程的参考书。

本书假定读者具有一定的 JavaScript 基础，如果读者不熟悉，也可以参考本书光盘中收录的巨经典的参考资料，然后再阅读本书。

本书的主要内容

本书讲解了大量的 JavaScript 编程技巧和优化技巧，并实现功能强大的 CSS、XPath Selector。同时，本书还立足于项目经验，结合大量翔实的示例，总结性地讲解一些经典的设计，向读者展示了同类书籍和网络中所无法学到的编程技术和经验。

本书共分为 15 章，现简要介绍如下。

第 1 章　快速入门

本章介绍光盘的使用，以及本书的代码规范和约定，本书中常见的高性能特殊语法，与 IE 和 FireFox 下兼容的设计，和本书后面章节将用到的公共代码的讲解。

第 2 章　Function 扩展和性能

本章介绍 Function 的扩展和高性能优化，如通用 Super 的实现，以及其作用域的应用，如 apply 和 call 在 web 事件编程处理中的特殊应用，以及匿名函数的递归，并实现支持 gzip 的 AJAX 框架，并能结合强大的 XPath 进行区域提交等项目经验应用。另外还讲解了如何给对象增加[0 ~ n]运算符。

第 3 章　Object 扩展和性能及 Web 应用之神兵利器

本章介绍了大量的 Object 泛型设计，如完美的 bind、each、addClass、removeClass、toggleClass、mstgcls、remove，并实现了功能强大的 CSS、XPath Selector，是全书的核心，选择器同时还支持 78 种 HTML DOM 事件的快速简易绑定。

第 4 章　Array 扩展和独辟蹊径的应用

本章介绍大量 Array 的泛型设计，介绍了 Array 性能的优化和大量的编程技巧，并介绍了同类书籍和网络中没有的各种 Array 下标的应用。

第 5 章　String 的扩展

本章介绍大量 String 的泛型设计，并讲解了 JavaScript 的一些隐匿自动转换类型特性，以及一些性能优化技巧，和 Web 开发中常常遇到的编码、解码、replace 技巧，与身份证的升位和验证、email 和 IP 的验证，它们看上去都平淡无奇，却是有别于同类书籍中的介绍，它们都蕴藏着作者多年的项目经验。

第 6 章　For 的优化和性能提升

本章介绍 JavaScript 编程语言中各种大循环的优化技巧，使得大循环代码能如同 CPU 流水线那样的原理高效的运作。

第 7 章　window 对象的性能应用

本章基于前面第 3 章的 XPath 成果，释疑 this，介绍 alert 导致的 web 项目中 session 的无故丢失和解决方案，实现 JavaScript 环境的并发多线程模型，介绍 JavaScript 对象在应用中的共享技巧，以及冻结业务办理系统的实现，同时介绍 JavaScript 对象的自动释放和 Form 表单改变的智能检测与实现。另外，本章还介绍，基于 XPath 实现快速简便令 HTML DOM 对象支持 Resize 的功能，包括表格列宽度的拖动。

第 8 章　Form 开发经验之谈

本章结合作者多年的项目经验介绍了 N 个文件上传的封装，Form 中元素之间关系的智能动态计算，以及光标跟随长度提示，与项目中经常遇到的界面区域可操作性的智能控制。8.9 节中还介绍 AJAX 在 google 多语言翻译支持服务使用中的设计。

第 9 章　HTC

本章介绍 htc 在 IE 中 behavior 的 autocomplete、HTML DOM Resize 的应用，同时介绍了 htc 的不足之处。

第 10 章　HTA

本章介绍 JavaScript 在 HTA(HTML 小应用程序)中的应用，并给出功能强大的 search And Replace 示例，同时介绍了 JSE 技术。

第 11 章　JavaScript 封装其他语言能调用的 DCOM

本章介绍 JavaScript 如何封装文本实现的 DCOM 应用，并给出作者的一个开源项目的详尽示例，如 ASP、PHP、PB 等的调用示例，示例中给出 JavaScript 和 VBScript 共同封装的文本 DCOM，在 win32 平台中的使用，如在 ASP 中的应用。

第 12 章　JavaScript 其他优化

本章总结和完善前面章节中关于 JavaScript 优化的技巧和性能优化。

第 13 章　Web 开发中 Table 相关经典设计

本章介绍大数据表格的排序，XPath 在表格中的各种应用，如结合多线程在表格的快速排序应用，快速改变 TD、列风格，自定义多列关系表达式动态求值并填写到指定列上（如：{2} = 4 * Math.log ({5}) + {8}, {3} = {2} / {7}）。

第 14 章　支持级联调用的 JSON-RPC for Java 轻量级框架的实现及应用

本章设计了轻量级的 JSON-RPC for Java AJAX 框架，它优胜于 http://oss.metaparadigm.com/jsonrpc/的是，支持在 JavaScript 中对 Java 对象的级联调用，而不需要额外的写 JavaScript 代码。同时，本章还描述、实现了 JSON-RPC 在网站多语言中支持框架，读者朋友可以直接应用于自己的网站，令自己的网站支持多种语言。

第 15 章　Java Script 在 WebDAV 中的应用

本章介绍了 WebDAV 在 Office 文档在线编辑，以及微软操作系统中的资源管理器直接和 WebDAB Folder 中的资源相互复制、粘贴、剪切、拖放或编辑等 JavaScript 技术。

本书附光盘 1 张，提供了各章的源程序代码，共计大约 348M，其中示例工程文件大约 21.6M，“说明.txt”文件介绍了使用光盘的方法，以及光盘中各目录与本书的对应关系。

如何使用本书

本书提供了大量的实用代码和例程，在阅读本书时，读者最好能准备一台计算机，以便能随时尝试本书所提供的例程代码。如果你阅读中有什么疑问，请联系我：jcore.xt@gmail.com、jcore@sina.com、x...y...@tom.com。

为了能够正确地使用本书提供的示例程序，读者的计算机系统应该满足如下的基本要求：

- CPU：486 或更高
- 内存：64M 或以上
- 操作系统：正版 Windows 95/98/Me/XP sp2 或 Windows NT/2000/2003
- 编程环境：请见本书第 1 章 1.1.4 节

夏天

2008 年 1 月于成都

目 录

第 1 章 快速入门 ····· 1

1.1 阅读本书方式 ····· 1
1.1.1 融合注释，会说话的代码 ····· 1
1.1.2 交叉阅读 ····· 1
1.1.3 水到渠成 ····· 1
1.1.4 环境 ····· 1
1.2 规范和约定 ····· 2
1.2.1 注释约定 ····· 2
1.2.2 命名约定 ····· 2
1.2.3 辅助调试编码约定 ····· 3
1.3 本书公共代码（Jcore.js）导读 ····· 3
1.4 本书常见代码语法预阅 ····· 9
1.4.1 逗号运算符号 ····· 9
1.4.2 JavaScript 独特的逻辑或、逻辑与运算符号 ····· 10
1.5 不推荐的中文命名方式 ····· 10
1.6 本书的写作思想 ····· 11

第 2 章 Function 扩展和性能 ····· 12

2.1 关于 Function 的设计原则建议 ····· 14
2.2 arguments，length ····· 16
2.3 toString 方法和 valueOf 方法 ····· 17
2.4 Function 的 toString 还有个妙用 ····· 17
2.5 绑定技术 ····· 18
HTML DOM 对象事件动态行为绑定 ····· 22
2.6 JavaScript 中的 this( )是什么 ····· 27
2.7 绑定 NaN、Infinity、null、undefined 给对象，this 会是什么 ····· 28
2.8 Function 对象的 apply 和 call 的区别 29
2.9 不用 new 的时候应该注意什么 ····· 31
2.9.1 一般直接调用方式 ····· 32
2.9.2 new 调用 ····· 32
2.9.3 apply 方式 ····· 32
2.9.4 call 方式 ····· 32
2.9.5 本书推荐的巧妙方式 ····· 34
2.10 如何给你的 function 增加[ ]运算符号的支持 ····· 35
Web 项目中 HTML DOM 的应用 ····· 37
2.11 如何让你的 this 加、减、乘、除 ····· 38
2.12 回调函数之函数“类型” ····· 39
流行中的 AJAX 里的回调应用 ····· 40
2.13 打造方便的调试函数 ····· 48
2.14 匿名函数 ····· 49
2.15 匿名函数的递归调用 ····· 51
2.16 继承后如何在函数中访问 Super 的简化设计 ····· 52
2.17 静态函数、属性的访问问题 ····· 53
2.18 总结 ····· 56

第 3 章 Object 扩展和性能及 Web 应用之神兵利器 ····· 58

3.1 typeof 和 constructor 总结 ····· 59
3.2 高性能初始化 ····· 60
3.2.1 和普通 Object 初始化的性能比较 ····· 60
3.2.2 与 function 的比较 ····· 66
3.3 当心，这里的 bind 对我无用 ····· 69
3.4 完美之绑定(bind)将通用于 Object、Function 和其他对象，同时支持给 Object 增加 [ ] 运算符号 ····· 71
3.5 通用属性复制 ····· 76
3.6 “继承” ····· 78
3.7 多态 ····· 79

3.8 推荐的对象模型方法[优点：一个名称空间，更便于内存才清理和释放]……79
3.9 Json 引入……81
3.9.1 什么是 Json……81
3.9.2 Json 串形化有什么用途……82
3.9.3 我为 Json 修正了什么……82
3.9.4 Json 使用举例……90
3.10 toString……91
3.11 融合 HTML DOM 支持的完美通用对象迭代器模型设计之 each……91
3.12 通用迭代模式的元素属性的获取和设置……96
Style 外观样式对象的获取和设置设计…105
3.13 通用对象排序模型设计之继承 Array 的相关功能……108
3.14 isPrototypeOf 和 hasOwnProperty……112
3.14.1 isPrototypeOf……112
3.14.2 hasOwnProperty……113
3.15 给所有 JavaScript 对象扩展属性、方法……113
3.16 打造“完美中的完美”的 HTML DOM 对象 XPath 对象选择器……115
3.16.1 XPath 对象选择器架构说明……115
3.16.2 模式匹配函数定义说明……116
3.16.3 基本正则表达式模式详解……116
3.16.4 可扩展的转义及可扩展模块说明……119
3.16.5 选择后的对象集合如何支持 HTML 事件的动态绑定……125
3.16.6 实现代码及使用举例……126
3.17 不要被怪物吓倒：0.1234[“each”]、true[“each”]……152
3.18 泛型设计之保留四舍五入保留小数：toFixed……152
3.19 addClass、removeClass、toggleClass、mstgcls、remove 的扩展设计……153

第 4 章 Array 扩展和独辟蹊径的应用……156
4.1 高性能初始化……156
4.2 数字下标……159
4.3 文本下标……160
4.4 “特殊”数字下标揭秘……161
4.5 对象下标揭秘……164
4.6 迭代器……166
4.7 扩展 Array 及应用……166
4.7.1 Max、Min 的扩展……167
4.7.2 indexOf、lastIndexOf 的扩展……170
4.7.3 some、every、filter、forEach、map 的扩展……172
4.7.4 克隆的实现……177
4.8 Array 函数功能增强……179
4.8.1 push 的增强，名为 ps……179
4.8.2 pop 的增强，名为 pp……181
4.8.3 shift 的增强，名为 sft……182
4.8.4 unshift 的增强，名为 usft……184
4.8.5 splice 的增强，名为 splc……185
4.9 给自己定义的对象增加[ ]运算符号…187

第 5 章 String 的扩展……188
5.1 如何将自定义对象直接赋予 HTML DOM 的 innerText、innerHTML、value 和其他属性……188
5.2 扩展自己的 append 方法……190
5.3 也谈 StringBuffer……191
5.3.1 StringBuffer 的实现……191
5.3.2 和 String 扩展的 append 性能比较 192
5.4 扩展自己的 trim、trimAll、trimL、trimR……195
5.5 兼容 Java 时 equals 的扩展……197
5.6 把 replace 玩到颠峰……198
5.6.1 稍微高级的玩法设计……198
5.6.2 最迷惑人的隐匿方式设计……201
5.7 转换汉字及双字节字符为 Unicode…203

5.7.1 Unicode 编码……203
5.7.2 Unicode 解码……204
5.8 加密我的 String……205
加、解密的实现……205
5.9 Web 常用验证功能集……209
5.9.1 身份证的验证和自动升位……209
5.9.2 E-mail 的验证……212
5.9.3 数字范围验证及在 Web 开发中的应用、输入限制的设计……214
5.9.4 Web 开发中 IP 地址的输入限制和验证……220
5.10 类似 Java 里的参数功能：toString 的{1 ……n}参数模式的合成……224
5.11 String 的 Left、Right、Mid 扩展……226

第 6 章 For 的优化和性能提升……228

6.1 使用 For…in 应该注意什么……228
6.2 do{……} while()和 for 的性能比较…229
6.3 匿名函数对象对性能的提升……230
6.4 倒着循环对性能的提升……231
6.5 最佳性能 for 的设计……231
6.6 本章 for 各种实现性能综合比较……233

第 7 章 window 对象的性能应用……237

7.1 alert，你让我的 session 去哪里了——如何让你的 session 永不过期的独有设计……237
7.2 神来之 this，前面章节释疑……245
7.3 如何冻结你的业务办理系统的操作界面[支持 iframe 和 frames 的多层嵌套]……247
7.4 通用“多线程”模型设计……256
7.5 execScript 实现你的 js 文件 import 功能……261
7.6 通用弹出窗口的对象共享应用设计，支持 N 层的弹出窗口及 Web 应用举例……262
7.7 createPopup 的问题……266
仿效 MSN 的消息提示……268
7.8 记得释放你的内存 onunload……269
表单修改状态在页面离开的时候进行提示保存……270
7.9 *、!、！[全角]在 getElementsByTagName 的特殊功效，和 document.all 的区别……272
7.10 局部刷新技术……274
7.11 处理好你组件的宽度……277
7.12 任意可见对象大小的鼠标可交互调整的支持……279

第 8 章 Form 开发经验之谈……284

8.1 通用获取输入对象元素 value 的设计……284
8.2 通用设置输入对象元素 value 的设计……287
8.3 异步支持设计……290
8.4 动态支持上传 N 个文件的封装设计 295
8.5 Web 开发中自定义输入对象组和表达式求值的设计……309
8.6 Web 开发中人性化输入值长度光标跟踪提示……311
8.7 如何在必输项没有输入值前其相关按钮、对象不可以操作……313
8.8 如何令表格中没有选择 radio、checkbox 输入对象的时候相应的按钮、相应的对象不可以操作……316
8.9 AJAX 让你的网站支持多语言——多语言的 google Translate AJAX 的封装……319

第 9 章 HTC……334

9.1 自定义 html 界面元素标签……337
9.2 behavior 的 autocomplete 下拉输入对象的设计……341
9.3 HTC 的 Bug……354
9.4 HTML DOM 的 resize 鼠标的支持设计……355

第 10 章 HTA ····· 357
10.1 什么是 HTA ····· 357
10.2 HTA 的特点 ····· 359
10.3 打造自己的功能强大的 searchAndReplace ····· 360
第 11 章 JavaScript 封装其他语言能调用的 DCOM ····· 370
11.1 JS 封装的 DCOM 的特点 ····· 371
11.2 JavaScript 封装的 DCOM 的应用范围 ····· 373
11.3 支持功能强大的 JavaScript 动态语言的设计 ····· 373
11.4 支持功能强大的 VBScript 动态语言的设计 ····· 374
11.5 JScript.Encode 解密的封装 ····· 374
11.6 Base64 编、解码的封装 ····· 380
11.7 繁、简体汉字相互转换的封装 ····· 382
第 12 章 JavaScript 其他优化 ····· 387
12.1 发布版本代码优化原则：能少则少 ····· 387
12.1.1 逗号的效益 ····· 387
12.1.2 return 的时候 ····· 388
12.1.3 new 的时候 ····· 388
12.1.4 多余分号的清除 ····· 388
12.1.5 多行注释的清除 ····· 389
12.1.6 单行注释的清除 ····· 389
12.1.7 运算符号前后多余空格的清除 ····· 389
12.2 if… else if 和 switch 的性能比较 ····· 389
12.3 Date 到 Number 的性能比较 ····· 390
12.4 !!是什么 ····· 392
12.5 不推荐使用 with 关键字 ····· 392
12.6 很少用的几个关键字 void, delete,instanceof, throw,finally, arguments ····· 393
12.7 void 用于函数的定义和调用 ····· 394
第 13 章 Web 开发中 Table 相关经典设计 ····· 395
13.1 快速多列组合排序的设计 ····· 395
13.2 快速改变列风格 ····· 398
13.3 根据规则快速改变 TD 风格 ····· 400
13.4 自定义多列关系表达式动态求值并填写到指定列上 ····· 403
13.5 模拟多线程在表格快速排序中的应用 ····· 407
第 14 章 支持级联调用的 JSON-RPC for Java 轻量级框架的实现及应用 ····· 411
14.1 配置 ····· 412
14.2 原理 ····· 412
14.3 框架实现及级联调用应用示例 ····· 414
14.4 可级联调用轻量级 JSON-RPC 框架在网站多语言智能转换中的应用 ··· 434
14.5 JSON-RPC 在 Web 项目中的应用 ··· 439
第 15 章 JavaScript 在 WebDAV 中的应用 ····· 442
15.1 什么是 WebDAV ····· 442
15.2 WebDAV 有哪些优点 ····· 443
15.3 WebDAV 配置 ····· 443
15.4 Office 等类型文件的在线编辑 ····· 450
15.5 WebDAV folder 在上传文件、目录等资源上的引用 ····· 452
附录 A ····· 454
附录 B ····· 459
参考文献 ····· 460

第1章　快速入门

1.1　阅读本书方式

1.1.1　融合注释，会说话的代码

本书的许多讲解内容都融合到代码的注释中，因此请读者仔细阅读和理解注释中的描述。

1.1.2　交叉阅读

本书交叉的内容很多，如果你遇到“请参考某某相关章节”类似的语言，你可以先跳到相应的章节，将当前你想了解的东西看明白了。这样，在反复的理解和阅读之中，就能很快掌握和找到你参考的“坐标”。

1.1.3　水到渠成

如果你遇到不理解的地方，请不要着急；继续阅读以后的章节，慢慢地你就会明白的。

1.1.4　环境

本书的代码都在 Windows XP sp2 的 IE6、IE7 上测试通过，还附带讲解一些 Firefox 和 IE 的区别。如果有涉及服务端的一些代码和术语，都是指 J2EE、JAVA JDK 1.6、TomCat 6.0 的环境，所有代码字符集都是 UTF-8。

运行本书所附光盘上的“Install.bat”，将在你的 C:\下创建 jcore 目录，并将 c:\jcore 目

录映射为虚拟盘符“X: ”。然后 install 会自动启动 tomcat 6.0，这时候你就可以在浏览器中输 http//127.0.0.1:8080:/jcore 看到本书的代码例程了。不过请确认你的本机 8080 端口没有被占用。如果有不明白的地方，你使用 Kmplayer 或“暴风影音”播放、观看光盘中的“光盘使用示例.mvb”或用 notepad.exe 打开并阅读“说明.txt”文件。

1.2 规范和约定

1.2.1 注释约定

本书中的 function 定义注释都约定用下面的形式，其好处是便于用我们自己写的自动生成的帮助手册的工具生成文档。除非是一般性测试，如不作最终的应用代码，那就不必依附下面的规范。而有关帮助生成工具，则请参考“第 11 章”中文件“AspLibInfo.hta”的实现。

注释规范如下：

```
// 功能描述：描述
// 返回信息：返回类型描述
// 使用指南：使用举例
// 应用范围：各种Web客户端和服务端开发，因为有的封装是可以用于封装DCOM的
```

1.2.2 命名约定

本书涉及的代码力求按下面的规范命名，如表 1-1 所示。除非是代码很少的地方，否则以代码量更少的方式出现。

表 1-1 命名约定

匈牙利命名法前缀	描　述	举　例
g_	全局变量前缀	var g_bDebug = true;
b	表示这个变量是逻辑、bool 类型，存储的是 true 或 false	var bHaveTitle = true;
sz 或 s	串，文本串对象，并且可以为空：“”或 null；这种对象规定不用前缀 o	var szName = “JavaScript”;
n	表示这个变量是数字类型	var nAge = 55;
o	表示这个变量是对象类型	var oDate = new Date(); // …. delete oDate;

fn	表示这个是函数的名称	function fnDisplayName()
		{
		alert(this.name)
		}

_	下画线开头的表示是自定义系统级的变量或对象，不需要其他人员定义的变量或对象	var __name = “test”;
a 或 arr，或单个 b 字母	表示是数组	var aElements = [];
i, j, x, y, z, n 等单个字母变量	用于循环语句中的下标变量	for(var I = 0; I < 1000; i++)

续表

匈牙利命名法前缀	描　述	举　例
$开头	表示是变量，注意函数名也是一种变量；另外还请注意，这个字符在其他一些浏览器中认为是非法的、不能用作命名开始的字母，如 Mozilla Firefox	var $nLen = 99;
r, oR	通常表示的是 TextRectangle、TextRange、controlRange、TR 等这样的对象	var oR = document.body.createTextRange(); // 为 body 元素创建一个 TextRange 对象
m	通常指正则表达式执行的结果	
oT, oTmp	通常指临时对象	

命名规则：前缀 + 名词、前缀 + 动名、前缀 + 动宾等。

约定的目的是为了代码的可读性和可维护性，当然也不是必须如此，因为有的地方大可不必一定要遵守约定。比如，有的参数类型本来就未定，只有在传入的时候、使用的时候才能确定，这时候我们就应该因地制宜、灵活处理。当然，原则还是力求使用最少的代码。

不过，本书的宗旨是："不是代码越少，性能就越高"，这一点请读者一定不要忘却。

1.2.3　辅助调试编码约定

如果需要加"try{…}catch(e){fnCatch(e)} "，那么约定需要给定义的函数加静态 name 属性。而这样的标准 try… catch 代码将在代码优化器中去除。一般情况下，如果你确实肯定你的代码不会发生错误，就算发生错误也不需要知道在哪个函数里发生的，那么不写 try …catch 也罢。因为在有的环境中，加了 try…catch 反而会降低性能。

1.3　本书公共代码（Jcore.js）导读

在后面各章节的演示代码中都会用到的一些公共的代码，就提到这里来加以注解，如下所示。

```
/*////////////////////////////////
 FileName: Jcore.js
 Ver     : 1.0
 Copyright (c) 2007 Summer
////////////////////////////////*/
// 防止多次引入，不过请注意，每个js文件里的防止多次引入的变量不能相同
if("undefined" ==  typeof g_bJcore)
{
  window.g_bJcore = true;

  // 表示是否开启调试标志
  var g_bDebugFlag = true;
  var _id = window.document.getElementById;
// 修正 Mozilla Firefox【下面简称MF】不支持和IE区别的一些功能
```

```
    // 另外，请注意：Jcore.js中函数定义的先后顺序不能随意更改，
    // 因为IE的函数定义中能访问后面定义的函数，但是MF中却必须严格按照定义的先后顺序出现，
    // 否则函数中引用其后面定义的函数就会发生未定义的错误
    if(/*@cc_on!@*/true)
    {
    fuction id(s)
    {
    return document.getElementByID(S);
    {
      // MF的document和window没有attachEvent和detachEvent方法
      document.attachEvent = function()
      {
        // 由于document在MF里没有事件绑定行为，因此转接到body上
        var a = Array.apply(null,arguments), element = document.body;
        element.addEventListener(a[0].replace(/^on/i, ''), a[1], false);
      };
      document.detachEvent = function()
      {
        var a = Array.apply(null,arguments), element = document.body;
        element.removeEventListener(a[0].replace(/^on/i, ''), a[1], false);
      };
      // 为什么这里的this就是绑定事件的对象呢？请参考本书“第3章”开篇
      window.attachEvent = Object.prototype.attachEvent = function()
      {
        var a = Array.apply(null,arguments), element = this.nodeName && this
|| window;
        element.addEventListener(a[0].replace(/^on/i, ''), a[1], false);
      };

      window.detachEvent = Object.prototype.detachEvent = function()
      {
        var a = Array.apply(null,arguments), element = this.nodeName && this
|| window;
        element.removeEventListener(a[0].replace(/^on/i, ''), a[1], false);
      };

      // MF的window也没有showModalDialog、showModelessDialog这两个方法
      // 写了下面的代码，MF中就可照样使用它们了：
      // window.showModalDialog("kkk.txt", [window], "dialogLeft:200px;
dialogTop:300px;dialogWidth:400px;dialogHeight:500px;")
      window.showModalDialog = window.showModelessDialog = function()
      {
        var a = Array.apply(null,arguments), oWin, s =
"titlebar=0,status=1,scrollbars=1,menubar=0,resizable=1,toolbar=0,location=0";
        if(a[2])
        {
          // 左边和顶端距离，这里把dialogLeft等单词中的dialog去除，是为了容错性更好
          var m = /Left\s*:\s*(\d+(?:(px|pt|pc|mm|cm|in|ex|em)))/im.exec(a[2]);
          if(m)s += ",left=" + m[1];
          m = /Top\s*:\s* (\d+)/im.exec(a[2]);
          if(m)s += ",top=" + m[1];
          // 宽度、高度
```

```
            m = /Height\s*:\s*(\d+)/im.exec(a[2]);
            if(m)s += ",height=" + m[1];
            m = /Width\s*:\s*(\d+)/im.exec(a[2]);
            if(m)s += ",width=" + m[1];
          }

          oWin = window.open(a[0], "_blank", s);
          oWin.dialogArguments = a[1];
          a = null;
          return oWin;
        }
    }

    // 功能描述：事件函数里获取第一个参数作为event对象，为了兼容不同的浏览器
    // 返回信息：返回[event对象，事件发生的html对象]
    // 使用指南：
    // 应用范围：各种Web客户端
    function getEO(event)
    {
      var e = event || getEO.arguments.callee.arguments[0] || window.event;
      return[e, e.srcElement || e.target];
    }

    // 功能描述：捕获程序中没有catch的错误消息，
    // 毕竟发生异常或错误给用户的体验很不好
    // 返回信息：返回：true/false
    // 使用指南：系统用
    this.onerror = function(sMsg, sUrl, sLine)
    {
      if(g_bDebugFlag)
      {
        // 初始化一个全局的变量用来存放错误消息，
        // 用全局的目的是为了能保存多个地方发生的多条错误或异常信息
        top.g_szLstErrMsg || (top.g_szLstErrMsg = "");
        top.g_szLstErrMsg = new StringBuffer("当前界面发生脚本错误，",
          "请通知本模块的开发人员或调整你的IE设置：",
          "错误消息：",sMsg,"\n\n",
          "发生的行：",sLine,"\n\n",
          "发生的URL：",sUrl,"\n\n").toString();
        alert(top.g_szLstErrMsg);
        top.g_szLstErrMsg = "";
      }
      return true;
    };

    // 功能描述：Array迭代器，本方法将在“第3章3.11节”详细讲解
    // 返回信息：返回当前Array，因此你可以连续多次each
    // 使用指南：
    [213,234].each(function(a){alert(a)}).each(function(a){alert (a)}).each
(function(a){alert(a)});
```

```
    // 应用范围：各种Web客户端和服务端开发
    Array.prototype.each = function(fn)
    {
      try{
        this._eachPos || (this._eachPos = 0);
        if(0 < this.length && fn.constructor && Function == fn.constructor)
        do{
           var aT = this[this._eachPos];
           // aT 保证不是null,undefined; aT.constructor保证这个属性存在
           // Array嵌套
           if(aT && aT.constructor && Array == aT.constructor)
              aT.each(fn);
           else
              fn.apply(aT, [aT]);
           this._eachPos++;
        }while(this._eachPos < this.length);
        this._eachPos = null;
      }catch(e){}
      return this;
    }

    // 功能描述：将给定的对象转换为有效的数组对象，通常用于转换arguments、
    // getElementsByName、getElementsByTagName、oSelectObject.options、oTable.
rows等的结果，当然，
    // 你自己写的支持[ ]运算符号的对象也可以，详细请见“第3章3.4节”；
    // 函数支持传入第二个参数，设置为true，否则不会对array对象进行深度合并转换；
    // 也就是说，默认是要深度转换的，这也带来了一些致命性的问题，HTML DOM对象是没有
    // constructor属性的，这就使得你转换options这样对象的时候必须指定不深度转换的标志；
    // 例如： _A(oSelectObject.options, true).each(function(){ 这里就可以做你的
迭代的内容 });
    // 返回信息：返回数组
    // 使用指南：var a = Array.apply(null, arguments);
    function _A(irb)
    {
      // _A(arguments)比较多，我们这样做就简化这种调用了，
      // 直接写_A( )就等于Array.apply(null, arguments)了
      if(0 == arguments.length)
        irb = _A.caller.arguments;
      if(!irb) return [];
      // 兼容自己给自定义对象写toArray方法
      if(irb.toArray)
         return irb.toArray();
      else
      {
        // 检查是否指定标志不进行深度处理
        var bNarr = !(undefined != arguments[1] && Boolean == arguments[1].
constructor && arguments[1]),
            rst = [], i = 0;
        if(irb.length)
        {
```

```
        var nL = irb.length, n = nL % 8, nOldN = n, i = 0;
        while(0 < n--)
        {
           if(bNarr && irb[i] && Array == (irb[i]['constructor'] || ''))
              rst = [].concat(rst, irb[i]);
           else rst.push(irb[i]);
           i++;
        }
        n = (nL - nOldN) / 8;
        while(0 < n--)
        {
           if(bNarr && irb[i] && Array == (irb[i]['constructor'] || ''))
              rst = [].concat(rst, irb[i]);
           else rst.push(irb[i]);i++;
           if(bNarr && irb[i] && Array == (irb[i]['constructor'] || ''))
              rst = [].concat(rst, irb[i]);
           else rst.push(irb[i]);i++;
           if(bNarr && irb[i] && Array == (irb[i]['constructor'] || ''))
              rst = [].concat(rst, irb[i]);
           else rst.push(irb[i]);i++;
           if(bNarr && irb[i] && Array == (irb[i]['constructor'] || ''))
              rst = [].concat(rst, irb[i]);
           else rst.push(irb[i]);i++;

           if(bNarr && irb[i] && Array == (irb[i]['constructor'] || ''))
              rst = [].concat(rst, irb[i]);
           else rst.push(irb[i]);i++;
           if(bNarr && irb[i] && Array == (irb[i]['constructor'] || ''))
              rst = [].concat(rst, irb[i]);
           else rst.push(irb[i]);i++;
           if(bNarr && irb[i] && Array == (irb[i]['constructor'] || ''))
              rst = [].concat(rst, irb[i]);
           else rst.push(irb[i]);i++;
           if(bNarr && irb[i] && Array == (irb[i]['constructor'] || ''))
              rst = [].concat(rst, irb[i]);
           else rst.push(irb[i]);i++;
        }
     }
     else if(undefined == irb.length)rst.push(irb);
     return rst;
  }
  return [];
}
// 功能描述：将传入参数设置到浏览器状态栏
// 返回信息：无返回
// 使用指南：Msg(999, ".88");// 浏览器状态栏显示999.88
// 应用范围：各种Web客户端和服务端开发
function Msg()
{
```