征服

JavaScript

高级程序设计与应用实例

施伟伟　冯梅　张蓓　编著

人民邮电出版社
POSTS & TELECOM PRESS

精通

JavaScript

高级程序设计与应用实例

征服 JavaScript 高级程序设计与应用实例

施伟伟　冯梅　张蓓　编著

人 民 邮 电 出 版 社
北　京

图书在版编目（CIP）数据

JavaScript 高级程序设计与应用实例 / 施伟伟，冯梅，张蓓编著. —北京：人民邮电出版社，2007.11
（征服系列）
ISBN 978-7-115-16773-6

Ⅰ. J… Ⅱ. ①施…②冯…③张… Ⅲ. JAVA 语言—程序设计 Ⅳ. TP312

中国版本图书馆 CIP 数据核字（2007）第 136462 号

内 容 提 要

本书通过大量的 JavaScript 应用实例帮助读者全面掌握 JavaScript 编程技术，全书共分 11 章，每章首先介绍基础知识，然后重点讲解相关的实例，让读者通过实践真正掌握 JavaScript 编程技术。第 1 章介绍 JavaScript 的基础知识，包括发展历史、相关标准、基本语法等；第 2 章介绍 JavaScript 的面向对象特性；第 3、4 章分别讲解 JavaScript 的字符串操作和浏览器编程的相关内容；第 5～7 章是本书的重点，分别讲解 DOM 的基础知识、事件处理和 DOM 样式编程；第 8 章介绍如何使用 JavaScript 进行 XML 编程；第 9 章也是本书的重点内容，讲解使用 JavaScript 与服务器端交互的各种技术；第 10 章介绍 JavaScript 与各种嵌入式对象的交互方法；第 11 章则介绍与 JavaScript 调试、优化相关的工具和技术。

本书面向的读者是具有初级和中级水平的 Web 开发人员，书中内容从目前 Web 开发中常见的应用场景出发，示例代码可以很容易地加入到实际应用项目中，具有很强的实用价值。

JavaScript 高级程序设计与应用实例

◆ 编　　著　施伟伟　冯　梅　张　蓓
责任编辑　李　岚

◆ 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
河北三河市海波印务有限公司印刷
新华书店总店北京发行所经销

◆ 开本：800×1000　1/16
印张：21.75
字数：476 千字　　2007 年 11 月第 1 版
印数：1－5 000 册　　2007 年 11 月河北第 1 次印刷

ISBN 978-7-115-16773-6/TP

定价：38.00 元

读者服务热线：(010)67132692　印装质量热线：(010)67129223

PREFACE

前　言

JavaScript 编程技术是 Web 开发人员的基本技能，也是实现 Ajax Web 应用的核心技术，在 Ajax 越来越流行的今天，精通 JavaScript 相关的开发技术显得尤为重要。JavaScript 编程方面有不少经典的书籍，其中绝大多数侧重于 JavaScript 相关技术原理的分析和讲解，而在项目中需要应用这些技术时，却并不是那么得心应手。要以最快的速度掌握一门技术，最佳的方式就是在实践中去应用它。

可惜的是，目前出版的 JavaScript 实例类图书无论在内容的广度和深度上，与相应的理论著作之间存在着较大的差距。在 Web 应用中 JavaScript 能够实现的功能绝不只是一些网页的特效，在 Nicholas C.Zakas 的《JavaScript 高级程序设计》一书中提到了多种 JavaScript 的应用场景，包括事件、数据验证、与服务器端通信等，而涉及这些 JavaScript 高级话题的应用实例，在相关书籍中却是非常的少见。本书编写的目的之一就是要填补 JavaScript 实例类书籍在这方面的空白，使读者能够更快速地掌握 JavaScript 的相关技术，并真正在实践中应用它们。

本书面向的读者是具有初级和中级水平的 Web 开发人员，书中提供大量的 JavaScript 应用实例帮助读者全面掌握 JavaScript 编程技术。这些实例考虑了目前 Web 开发中常见的应用场景，并且可以很容易地加入到实际应用项目中，具有很强的实用价值。除了一些浏览器特有的功能以外（例如 ActiveX 控件的相关示例），本书提供的示例均兼容各种主流浏览器。

全书共分 11 章，在大多数章节中，将首先对基础知识进行简要介绍，然后重点讲解相关的实例，使读者通过实践真正掌握 JavaScript 编程技术。

第 1 章对 JavaScript 的基础知识，包括 JavaScript 的发展历史、相关标准、基本语法等进行了介绍。

第 2 章对 JavaScript 的面向对象特性进行了介绍，读者可以从这一章中了解如何在 JavaScript 中实现基本的面向对象特性，如封装、继承、多态等。

第 3、4 章分别讲解了 JavaScript 的字符串操作和浏览器编程。

第 5～7 章是本书的重点内容，分别讲解了 DOM 的基础知识、事件处理和 DOM 样式编程。这几部分内容分别对应于 DOM2 标准中的 DOM 核心、DOM 事件和 DOM CSS。

第 8 章介绍了如何使用 JavaScript 进行 XML 编程。

第 9 章也是本书的重点内容，讲解了使用 JavaScript 与服务器端交互的各种技术，包括 Image 对象、隐藏框架、远程脚本和 XMLHttpRequest 对象等。

第 10 章介绍了 JavaScript 与各种嵌入式对象的交互方法，包括 ActiveX 控件、Flash 等。

第 11 章介绍了 JavaScript 调试、优化相关的工具和技术。

本书由施伟伟、冯梅、张蓓执笔编写，此外，参与本书编辑、修改和整理的还有陈宏伟、刘锋、徐红、冒小飞、周虹、范锡琴、岳文宇、朱琪、范宝龙、任献红和范宝琦等。在此对以上人员致以诚挚的谢意！

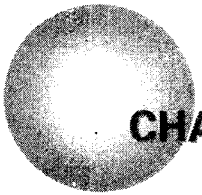
由于时间仓促且作者的水平有限，书中错误和不妥之处在所难免，敬请读者批评指正。如果有任何问题，可以发送 E-mail 到 shiweiwei97@gmail.com，作者将尽快予以答复。

编　者

2007 年 7 月

CONTENT

目　录



第 1 章　JavaScript 语言基础

JavaScript 语言是一门解释型的脚本语言，常在 Web 开发中用于增强网页与应用程序间的交互。网景公司（Netscape）首先在它的浏览器 Netscape Navigator 中实现了 JavaScript，随后其他的浏览器也相继推出了各自的 JavaScript 版本，并且与标准化组织 ECMA 一起制定了 ECMAScript 语言规范。目前，ECMAScript 第 3 版是 ECMAScript 的最新版本，也是 JavaScript 的工业标准，本章将基于该标准讲解 JavaScript 的基本语法和内置对象特性。

1.1　JavaScript 的发展历史

JavaScript 可以运行在多种宿主环境下，浏览器是其中最主要的一种，本书只讨论浏览器中的 JavaScript。实际上，JavaScript 的发展历史和浏览器的发展过程是密切相关的。

1.1.1　Netscape 时代

JavaScript 首次亮相于 1995 年。由于当时的浏览器缺乏相应的处理机制，所有的数据验证都必须在服务器端完成，因此用户常常会遇到少填写一、两个字段就不得不重新填写整个表单的情况。特别是当时的接入速率普遍为 28.8kB/s，漫长的等待让用户十分厌烦，希望在客户端进行数据校验的呼声越来越高。在这样的情况下，网景公司（Netscape）在其出品的浏览器软件 Netscape Navigator 2.0 中发布了 JavaScript 的 1.0 版本。推出 JavaScript 1.0 的主要目的正是为了实现在客户端验证用户输入的信息，从而减少不必要的等待时间。

关于 JavaScript 的命名还有一段小插曲，它曾经用过的名字有 LiveWire、LiveScript，在正式发布前才改名为 JavaScript，这样做可能是为了与当时正如日中天的 Java 语言沾上点儿边。 当然不管怎样，JavaScript 1.0 的推出都获得了极大的成功，从此 JavaScript 成为了进行 Web 开发所必备的一项技术。随后 Netscape 于 1996 年在 Netscape Navigator 3.0 中发布了 JavaScript 1.1。

1.1.2　Microsoft 的加入

90 年代，微软公司（Microsoft）的 Internet Explorer（IE）是 Netscape Navigator 的主要

竞争对手，它依靠其操作系统的绝对优势向 Netscape Navigator 发起了强大的攻势。从 Windows 95 OSR2 开始，IE 3.0 就成为所有 Windows 操作系统的默认浏览器。微软分别在 IE 3.0 的早期和后期版本中实现了 JScript 1.0 和 JScript 2.0，相当于 Netscape 的 JavaScript 1.0 和 JavaScript 1.1；在随后的 IE 4.0 中还实现了 JScript 3.0，相当于 Netscape 的 JavaScript 1.3。在很长一段时期内，IE 的市场占有率一直占有绝对优势，因此微软的 JScript 也成为 JavaScript 的事实标准。

尽管如此，Netscape 一直没有放弃与微软的竞争，在 1997～2002 年间，它先后发布了 Netscape Navigator 的 4.0～4.8 版本。1998 年，Netscape 还宣布 Netscape Navigator 免费，并且公布了 Netscape Communicator（包括 Netscape Navigator）的所有源代码，这也是 Mozilla 项目的开始。同年，Netscape 公司被 AOL 收购，之后 AOL 又被时代华纳公司收购，2003 年 7 月时代华纳宣布解散 Netscape 公司，从此 Netscape 公司成为历史，但是即便如此，在此之后 AOL 还是发布了 Netscape Navigator 7.2～8.x 版本，开发方面由加拿大的 Mercurial Communications 公司完成。

1.1.3 Firefox 的异军突起

微软在赢得浏览器大战胜利之后，IE 在标准化方面却少有进展，对于 DOM 和 CSS 标准的支持一直没有改善，直至 IE 7.0 推出，IE 对于 DOM 的支持还是 Level 1 和少数的 Level 2，对 CSS 的支持仍然存在着不少的缺陷，以致于开发人员不得不使用各种特殊技巧（Hack）来规避它们。

与 IE 形成鲜明对比的是 Mozilla Firefox。标签页浏览方式、良好的安全性和可扩展性、对标准的良好支持程度，使得 Firefox 赢得了众多用户（特别是 Web 开发人员）的青睐。Firefox 是基于 Gecko 引擎的浏览器，其内部使用的是名为 Spidermonkey 的 JavaScript 解释引擎，它是基于 C 语言的 JavaScript 实现的，也是目前性能最好的 JavaScript 引擎。目前，Firefox 最新的稳定版本是 Firefox 2.0，它实现的 JavaScript 版本是 1.7。在未来的 Firefox 3.0 中，将会发布 JavaScript 2.0。Firefox 是当前对标准支持程度最高的浏览器，它支持全部的 DOM Level 1 和 Level 2，以及少数 Level 3；在 CSS 方面，它支持大部分的 CSS 2 和少部分的 CSS 3。

今天我们在使用 JavaScript 进行 Web 开发时，至少需要考虑兼容 Firefox 和 IE 两种浏览器。由于 Firefox 对标准的支持程度更高，通常是 Web 开发人员首选的开发平台。

1.1.4 ECMAScript

不同厂商所实现的 JavaScript 并不完全一致，这在很大程度上增加了开发人员的工作量，甚至有时候会导致一些概念上的混淆。因此在 1997 年，JavaScript 1.1 作为一个草案被提交到欧洲计算机制造商协会（ECMA），由 Netscape、微软、SUN、Borland 以及其他一些公司的程序员组成的 TC39 委员会根据这一草案最终制定出 ECMA-262 标准，该标准

给出了 ECMAScript 脚本语言的定义。随后，国际标准化组织（ISO）和国际电工委员会（IEC）接纳 ECMAScript 为标准（ISO/IEC-1626），至此 ECMAScript 成为 JavaScript 的工业标准。

ECMAScript 从诞生至今共经历了 3 个版本。其中，ECMAScript 第 1 版对 JavaScript 1.1 的基本特性进行了标准化，并且增加了一些新特性，在 ECMAScript 第 1 版中没有对 switch 语句和正则表达式进行标准化。ECMAScript 第 2 版与第 1 版没有功能或者特性上的区别，只是增加了说明。与 ECMA 第 1 版和第 2 版相对应的浏览器实现是 Netscape 4.x 和 IE 4.0。ECMAScript 目前的最新版本是第 3 版，这个版本中增加了 switch 语句、异常处理和正则表达式等特性，相应的浏览器实现版本是 Mozilla、Netscape 6 和 IE 5.5+。ECMAScript 的第 4 版正在开发过程中，它对应的实现将是未来的 JavaScript 2.0。

这里需要说明的是，ECMAScript 虽然源自 JavaScript，但是作为一个标准，JavaScript 并不是它的惟一实现。例如 Adobe 公司的 ActionScript 也是符合 ECMAScript 标准的脚本语言。如果用面向对象编程的概念来作个比喻，ECMAScript 是接口（Interface），那么 JavaScript、ActionScript 等语言则是实体类（Concrete Class）。ECMAScript 对实现语言的共性进行抽象，描述了脚本语言的语法、数据类型和对象等基本要素，而如何具体实现这些要素，ECMAScript 是不会对其进行规定的。每个具体的实现语言都会增加一些 ECMAScript 未曾定义的特性，这一点在开发兼容各种浏览器的 Web 应用时需要特别注意，应该尽可能地使用 ECMAScript 中定义的方法和特性，而不要使用某种浏览器独有的特性。

Web 浏览器不是 ECMAScript 的惟一宿主环境，例如 Windows 操作系统可以通过 WScript 或者 CScript 来执行 JavaScript 脚本；Netscape 的应用服务器产品 Netscape Enterprise Server（NES）支持使用 JavaScript 进行服务器端开发，甚至与 Java 类进行交互。在本书中，主要考虑的宿主环境是 Web 浏览器，在这一领域最主要的两大 ECMAScript 解释引擎分别是微软的 JScript 和 Mozilla 的 Spidermonkey。根据相关的测试表明，Spidermonkey 是当今执行效率最高的 JavaScript 引擎，这也是在众多与 JavaScript 相关的性能测试报告中，Mozilla Firefox 的成绩会远远高于 IE 的主要原因。

1.2 JavaScript 语法基础

JavaScript 是一门解释型的、基于对象的脚本语言，它的基本语法与 Java、C 等高级语言类似。JavaScript 的运行环境通常是 Web 浏览器，但是并不局限于浏览器。JavaScript 可操作的对象一般划分为两大类：一类是由 ECMAScript 定义的 JavaScript 核心对象，如数组对象、日期时间对象等；另一类是 JavaScript 宿主环境定义的各种对象，例如 Web 浏览器中的 BOM 模型、Windows Script Host 中的 Shell 对象等。下面将分别在本节和 1.3 节中介绍 JavaScript 的基本语法和内置对象的特性。

1.2.1 语句

同许多编程语言一样，JavaScript 是采用文本方式编写而成的，并且由相关的语句集组成块（Block）和注释。JavaScript 程序是 JavaScript 语句的集合，每一条 JavaScript 语句由若干表达式组织在一起，完成一个任务。一条 JavaScript 语句可以是一行，也可以是多行。

和 Java、C 语言类似，JavaScript 使用分号“;”表示一条语句的结束；而与 Java、C 语言不同的是，用分号结束一条语句并不是强制性的要求。例如下面的写法：

```
var a = 1;  // 以分号结尾的 JavaScript 语句
var b = 2   // 没有分号结尾的 JavaScript 语句
```

都是正确的，但是如果在 Java 或者 C 语言中不使用分号结束语句，那么就会导致编译错误。

JavaScript 解释器在语法检查方面相对比较宽松，但是我们还是建议开发人员在编写 JavaScript 代码时要尽量保持比较严谨的书写风格，最好使用分号结束语句。这样做有以下几点好处。

（1）代码便于阅读，不会导致歧义；

（2）在使用一些 JavaScript 代码压缩和代码混淆工具处理代码时，不会发生错误；

（3）保证代码在各种浏览器中均能正确执行，因为某些浏览器的 JavaScript 解释器要求语句必须以分号作为结束符，否则不能执行。

一组大括号“{}”内的 JavaScript 语句称为语句块，一个语句块内的语句可以被当作一条语句来处理。在 JavaScript 语言中一般的语句块并不代表新的范围，这与 C 语言是不同的。例如下面的 JavaScript 语句块：

```
{
    var v1 = 20;
    var v2 = 40;
}
```

其中变量 v1 和 v2 在大括号之外同样是有意义的。而在 C 语言中，类似的变量定义只在语句块的内部有效。

此外，在 JavaScript 语言中函数定义内部声明的变量只在其内部有效，如果在函数定义之外访问，那么就会返回 undefined。例如：

```
function test() {
    var v1 = 20;
    var v2 = 40;
}
```

如果在 test 函数之外访问变量 v1 或者 v2，会返回 undefined。

1.2.2 注释

JavaScript 语言支持 C 语言风格（/* … */）以及 Java、C++语言风格（// …）的注释方式。

使用双斜杠可以表示单行的注释，例如：

```
var v1 = 20;  // 这是一个单行注释
```

使用/*和*/的组合可以将多行文本内容作为注释，例如：

```
/*
    多行注释示例
    多行注释示例
*/
```

在使用/*和*/的组合添加注释时必须注意，注释的内容不能嵌套。例如下面的代码：

```
/*
    多行注释示例
    /*
    嵌套的注释
    */
    多行注释示例
*/
```

JavaScript 解释器将会把第一个/*与第一个*/当成一组，把它们内部的文本作为注释，而第一个*/之后的内容则会被解释为 JavaScript 语句，从而发生语法错误。鉴于这个原因，通常建议开发人员采用双斜杠的方式添加多行注释，如：

```
// 注释第 1 行
// 注释第 2 行
```

当然在使用双斜杠添加注释时也有需要注意的事项，那就是要正确区分正则表达式的定义。JavaScript 使用/expression/来表示一个正则表达式，例如/\d{2}/。这个正则表达式表示匹配由两个数字组成的字符串。如果有下面的 JavaScript 代码：

```
var re = /\d{2}///2
```

它表示的含义是将变量 re 定义为一个正则表达式，之后的 2 会被当作注释文本处理。如果省去上面代码中的一个斜杠，即：

```
var re = /\d{2}//2
```

那么，这里的 2 就不再是注释的内容，而代表了数字 2，这条语句的作用是将正则表达式/\d{2}/除以 2 的运算结果赋给变量 re。尽管这样的语句并没有任何实际含义，但是从语法角度上讲是正确的，因此在 JavaScript 代码中使用双斜杠添加注释时，需要特别注意这种情况，以免导致错误发生。

1.2.3 直接量

直接量是在程序中直接出现的数据值，JavaScript 支持以下几种形式的直接量。

```
20                  // 整数
0.5                 // 浮点数
"hello"             // 字符串
'hello'             // 字符串
true/false          // 布尔值
/abc/gi             // 正则表达式
null                // 空对象
```