

图书基本信息

书名：<<JavaScript语言精髓与编程实践>>

13位ISBN编号：9787121156403

10位ISBN编号：7121156407

出版时间：2012-3

出版时间：电子工业出版社

作者：周爱民

页数：456

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

内容概要

本书详细讲述JavaScript

作为一种混合式语言的各方面特性，包括过程式、面向对象、函数式和动态语言特性等，在动态函数式语言特性方面有着尤为细致的讲述。

本书的主要努力之一，就是分解出这些语言原子，并重现将它们混合在一起的过程与方法。

通过从复杂性到单一语言特性的还原过程，读者可了解到语言的本质，以及“层出不穷的语言特性”背后的真相。

本书主要的著述目的是基于一种形式上简单的语言来讲述“语言的本质及其应用”。

本书详细讲述了通过框架执行过程来构造一个JavaScript

扩展框架的方法，并完整地讲述了框架扩展中各种设计取舍，因此可以作为研究计算机程序设计语言时的参考，用以展示现实系统如何实现经典理论中的各种编程范型。

作者简介

周爱民, (Aimingoo), 国内软件开发界资深软件工程师、架构师, 技术作家。

有十余年的软件开发、项目管理、团队建设的经验。

著有《Delphi源代码分析》、《大道至简》和《JavaScript语言精髓与编程实践》等专著。

2001年, 主持完成的“极光数据处理仓库中心系统”被河南省信息产业厅授予省高新技术产品二等奖。

2003年, 被美国Borland公司授予“ Borland Delphi产品专家 ”称号。

2004年, 出版《Delphi源代码分析》, 被誉为“ Delphi领域精品著作 ”。

2005年, 发布《大道至简》电子版(第一版)。

2006年, 发起开源项目QomolangmaOpenProiect, 探讨语言系统基础技术。

2007年3月, 出版《大道至简》(第二版)。

2008年3月, 出版《JavaScript语言精髓与编程实践》第一版。

书籍目录

第1部分 语言基础

第1章 十年JavaScript

1.1 网页中的代码

1.1.1 新鲜的玩意儿

1.1.2 第一段在网页中的代码

1.1.3 最初的价值

1.2 用JavaScript来写浏览器上的应用

1.2.1 我要做一个聊天室

1.2.2 Flash的一席之地

1.2.3 RWC与RIA之争

1.3 没有框架与库的语言能怎样发展呢

1.3.1 做一个框架

1.3.2 重写框架的语言层

1.3.3 富浏览器端开发与AJAX

1.4 语言的进化

1.4.1 Qomo的重生

1.4.2 QoBean是对语言的重新组织

1.4.3 JavaScript作为一门语言的进化

1.5 为JavaScript正名

1.5.1 JavaScript

1.5.2 Core JavaScript

1.5.3 SpiderMonkey JavaScript

1.5.4 ECMAScript

1.5.5 JScript

1.5.6 总述

1.6 JavaScript的应用环境

1.6.1 宿主环境

1.6.2 外壳程序

1.6.3 运行期环境

第2章 JavaScript的语法

2.1 语法综述

2.1.1 标识符所绑定的语义

2.1.2 识别语法错误与运行错误

2.2 JavaScript的语法：变量声明

2.2.1 变量的数据类型

2.2.1.1 基本数据类型

2.2.1.2 值类型与引用类型

2.2.2 变量声明

2.2.3 变量与直接量

2.2.3.1 字符串直接量、转义符

2.2.3.2 数值直接量

2.2.4 函数声明

2.3 JavaScript的语法：表达式运算

2.3.1 一般表达式运算

2.3.2 逻辑运算

- 2.3.3 字符串运算
- 2.3.4 比较运算
 - 2.3.4.1 等值检测
 - 2.3.4.2 序列检测
- 2.3.5 赋值运算
- 2.3.6 函数调用
- 2.3.7 特殊作用的运算符
- 2.3.8 运算优先级
- 2.4 JavaScript的语法：语句
 - 2.4.1 表达式语句
 - 2.4.1.1 一般表达式语句
 - 2.4.1.2 赋值语句与隐式的变量声明
 - 2.4.1.3 (显式的) 变量声明语句
 - 2.4.1.4 函数调用语句
 - 2.4.2 分支语句
 - 2.4.2.1 条件分支语句 (if语句)
 - 2.4.2.2 多重分支语句 (switch语句)
 - 2.4.3 循环语句
 - 2.4.4 流程控制：一般子句
 - 2.4.4.1 标签声明
 - 2.4.4.2 break子句
 - 2.4.4.3 continue子句
 - 2.4.4.4 return子句
 - 2.4.5 流程控制：异常
- 2.5 面向对象编程的语法概要
 - 2.5.1 对象直接量声明与实例创建
 - 2.5.1.1 使用构造器创建对象实例
 - 2.5.1.2 对象直接量声明
 - 2.5.1.3 数组直接量声明
 - 2.5.1.4 正则表达式直接量声明
 - 2.5.1.5 【ES5】在对象直接量中使用属性读写器
 - 2.5.1.6 讨论：初始器与直接量的区别
 - 2.5.2 对象成员
 - 2.5.2.1 对象成员列举、存取和删除
 - 2.5.2.2 属性存取与方法调用
 - 2.5.2.3 对象及其成员的检查
 - 2.5.2.4 可列举性
 - 2.5.3 默认对象的指定
- 2.6 【ES5】严格模式下的语法限制
 - 2.6.1 语法限制
 - 2.6.2 严格模式的范围
- 2.7 运算符的二义性
 - 2.7.1 加号“+”的二义性
 - 2.7.2 括号“()”的二义性
 - 2.7.3 冒号“:”与标签的二义性
 - 2.7.4 大括号“{}”的二义性

2.7.5 逗号“,”的二义性

2.7.6 方括号“[]”的二义性

第2部分 语言特性及基本应用

第3章 JavaScript的非函数式语言特性

3.1 概述

3.1.1 命令式语言与结构化编程

3.1.2 结构化的疑难

3.1.3 “面向对象语言”是突破吗

3.1.4 更高层次的抽象：接口

3.1.5 再论语言的分类

3.1.6 JavaScript的语源

3.2 基本语法的结构化含义

3.2.1 基本逻辑与代码分块

3.2.2 模块化的层次：语法作用域

3.2.2.1 主要的语法作用域及其效果

3.2.2.2 语法作用域之间的相关性

3.2.3 执行流程及其变更

3.2.3.1 级别2：“break < label >”等语法

3.2.3.2 级别3：return子句

3.2.3.3 级别4：throw语句

3.2.3.4 执行流程变更的内涵

3.2.4 模块化的效果：变量作用域

3.2.4.1 级别1：表达式

3.2.4.2 级别2：语句

3.2.4.3 级别3：函数（局部）

3.2.4.4 级别4：全局

3.2.4.5 变量作用域中的次序问题

3.2.4.6 变量作用域与变量的生存周期

3.2.5 语句的副作用

3.3 JavaScript中的原型继承

3.3.1 空对象（null）与空的对象

3.3.2 原型继承的基本性质

3.3.3 空的对象是所有对象的基础

3.3.4 构造复制？

写时复制？

还是读遍历？

3.3.5 构造过程：从函数到构造器

3.3.6 预定义属性与方法

3.3.7 原型链的维护

3.3.7.1 两个原型链

3.3.7.2 constructor属性的维护

3.3.7.3 内部原型链的作用

3.3.7.4 【ES5】在SpiderMonkey与ES5中的原型链维护

3.3.8 原型继承的实质

3.3.8.1 原型修改

3.3.8.2 原型继承

3.3.8.3 原型继承的实质：从无到有

3.3.8.4 如何理解“继承来的成员”

3.4 JavaScript的对象系统

3.4.1 封装

3.4.2 多态

3.4.3 事件

3.4.4 类抄写？

或原型继承？

3.4.4.1 类抄写

3.4.4.2 原型继承存在的问题

3.4.4.3 如何选择继承的方式

3.4.5 JavaScript中的对象（构造器）

3.4.6 不能通过继承得到的效果

3.5 【ES5】可定制的对象属性

3.5.1 属性描述符

3.5.1.1（一般的）数据属性描述符

3.5.1.2（带读写器的）存取属性描述符

3.5.1.3 直接量形式的初始器是语法格式，而非描述符

3.5.2 定制对象属性

3.5.2.1 属性声明以及获取属性描述符

3.5.2.2 新的对象创建方法：Object.create()

3.5.3 属性状态维护

3.5.3.1 取属性列表

3.5.3.2 使用defineProperty来维护属性的性质

3.5.3.3 对于继承自原型的属性，修改其值的效果

3.5.3.4 重写原型继承来的属性的描述符

第4章 JavaScript的函数式语言特性

4.1 概述

4.1.1 从代码风格说起

4.1.2 为什么常见的语言不赞同连续求值

4.1.3 函数式语言的渊源

4.2 函数式语言中的函数

4.2.1 函数是运算元

4.2.2 在函数内保存数据

4.2.3 函数内的运算对函数外无副作用

4.3 从运算式语言到函数式语言

4.3.1 JavaScript中的几种连续运算

4.3.1.1 连续赋值

4.3.1.2 三元表达式的连用

4.3.1.3 一些运算连用

4.3.1.4 函数与方法的调用

4.3.2 运算式语言

4.3.2.1 运算的实质是值运算

4.3.2.2 有趣的运算：在IE和J2EE中

4.3.3 如何消灭掉语句

4.3.3.1 通过表达式消灭分支语句

- 4.3.3.2 通过函数递归消灭循环语句
 - 4.3.3.3 其他可以被消灭的语句
 - 4.4 函数：对运算式语言的补充和组织
 - 4.4.1 函数是必要的补充
 - 4.4.2 函数是代码的组织形式
 - 4.4.3 重新认识“函数”
 - 4.4.3.1 “函数” == “lambda”
 - 4.4.3.2 当运算符等义于某个函数时
 - 4.4.4 JavaScript语言中的函数式编程
 - 4.5 JavaScript中的函数
 - 4.5.1 可变参数与值参数传递
 - 4.5.2 非惰性求值
 - 4.5.3 函数是第一型
 - 4.5.4 函数是一个值
 - 4.5.5 可遍历的调用栈
 - 4.5.5.1 callee：我是谁
 - 4.5.5.2 caller：谁呼（叫）我
 - 4.6 闭包
 - 4.6.1 闭包与函数实例
 - 4.6.1.1 什么是闭包
 - 4.6.1.2 什么是函数实例与函数引用
 - 4.6.1.3（在被调用时，）每个函数实例至少拥有一个闭包
 - 4.6.2 闭包与调用对象
 - 4.6.2.1 “调用对象”的局部变量维护规则
 - 4.6.2.2 “全局对象”的变量维护规则
 - 4.6.2.3 函数闭包与“调用对象”的生存周期
 - 4.6.3 闭包相关的一些特性
 - 4.6.3.1 引用与泄漏
 - 4.6.3.2 函数实例拥有多个闭包的情况
 - 4.6.3.3 语句或语句块中的闭包问题
 - 4.6.3.4 闭包中的标识符（变量）特例
 - 4.6.3.5 函数对象的闭包及其效果
 - 4.6.4 闭包与可见性
 - 4.6.4.1 函数闭包带来的可见性效果
 - 4.6.4.2 对象闭包带来的可见性效果
 - 4.6.4.3 匿名函数的闭包与可见性效果
 - 4.7 【ES5】严格模式与闭包
 - 4.7.1 严格模式下的执行限制
 - 4.7.2 严格模式下的匿名函数递归问题
- 第5章 JavaScript的动态语言特性
- 5.1 概述
 - 5.1.1 动态数据类型的起源
 - 5.1.2 动态执行系统的起源
 - 5.1.2.1 编译系统、解释系统与编码
 - 5.1.2.2 动态执行
 - 5.1.3 脚本系统的起源
 - 5.1.4 脚本只是一种表面的表现形式

5.2 动态执行

5.2.1 动态执行与闭包

5.2.1.1 eval使用全局闭包

5.2.1.2 eval使用当前函数的闭包

5.2.2 动态执行过程中的语句、表达式与值

5.2.3 奇特的、甚至是负面的影响

5.3 动态方法调用 (call、apply与bind)

5.3.1 动态方法调用中指定this对象

5.3.2 丢失的this引用

5.3.3 栈的可见与修改

5.3.4 兼容性：低版本中的call()与apply()

5.3.5 【ES5】兼容性：ES5中的call()、apply()

5.3.6 【ES5】bind()方法与函数的延迟调用

5.4 重写

5.4.1 原型重写

5.4.2 构造器重写

5.4.2.1 语法声明与语句含义不一致的问题

5.4.2.2 对象检测的麻烦

5.4.2.3 构造器的原型 (prototype属性) 不受重写影响

5.4.2.4 “内部对象系统” 不受影响

5.4.2.5 让用户对象系统影响内部对象系统

5.4.2.6 构造器重写对直接量声明的影响

5.4.2.7 构造绑定

5.4.2.8 内置构造器重写的概述

5.4.3 对象成员的重写

5.4.3.1 成员重写的检测

5.4.3.2 成员重写的删除

5.4.4 宿主对重写的限制

5.4.5 引擎对重写的限制

5.4.5.1 this的重写

5.4.5.2 语句语法中的重写

5.4.5.3 结构化异常处理中的重写

5.5 包装类：面向对象的妥协

5.5.1 显式包装元数据

5.5.2 隐式包装的过程与检测方法

5.5.3 包装值类型数据的必要性与问题

5.5.4 其他直接量与相应的构造器

5.5.4.1 函数特例

5.5.4.2 正则表达式特例

5.6 关联数组：对象与数组的动态特性

5.6.1 关联数组是对象系统的基础

5.6.2 用关联数组实现的索引数组

5.6.3 干净的对象

5.7 类型转换

5.7.1 宿主环境下的特殊类型系统

5.7.2 值运算：类型转换的基础

5.7.3 隐式转换

- 5.7.3.1 运算导致的类型转换
- 5.7.3.2 语句（语义）导致的类型转换
- 5.7.4 值类型之间的转换
 - 5.7.4.1 undefined的转换
 - 5.7.4.2 number的转换
 - 5.7.4.3 boolean的转换
 - 5.7.4.4 string的转换
 - 5.7.4.5 值类型数据的显式转换
- 5.7.5 从引用到值：深入探究valueOf()方法
- 5.7.6 到字符串类型的显式转换
 - 5.7.6.1 重写toString()方法
 - 5.7.6.2 从数值到字符串的显式转换
 - 5.7.6.3 其他类型的显式转换
 - 5.7.6.4 序列化

第3部分 编程实践

第6章 元语言：QoBean核心技术与实现

- 6.1 QoBean语言层的基本特性
 - 6.1.1 QoBean语言层概要
 - 6.1.1.1 如何使用QoBean
 - 6.1.1.2 QoBean中的面向对象（OOP）
 - 6.1.1.3 QoBean中的接口（Interface）
 - 6.1.1.4 QoBean中的切面（Aspect）
 - 6.1.2 Qomo的体系结构及其与QoBean的关系
- 6.2 QoBean的元语言特性
 - 6.2.1 QoBean如何理解元语言
 - 6.2.2 算法与数据结构
 - 6.2.2.1 引用类型与值类型的数据
 - 6.2.2.2 函数调用
 - 6.2.2.3 源起
 - 6.2.2.4 小结
 - 6.2.3 代码组织形式
 - 6.2.3.1 块，以及基于块的编织
 - 6.2.3.2 更强的编织
 - 6.2.3.3 逻辑代码块：局部、全局，以及闭包
 - 6.2.3.4 逻辑的属主
 - 6.2.4 对“如何组织对象”的补充
 - 6.2.4.1 原子，与原子联结的友类、友函数
 - 6.2.4.2 对象唯一化
 - 6.2.5 综述
- 6.3 基于元语言实现的语言特性
 - 6.3.1 基于元语言的类继承框架
 - 6.3.1.1 类注册过程
 - 6.3.1.2 示例：实现MetaClass与MetaObject的约定
 - 6.3.1.3 完整的Qomo语法实现
 - 6.3.1.4 类类型树的建立
 - 6.3.2 多投事件
 - 6.3.3 其他语言特性的实现

6.4 基于元语言实现的DSL

6.4.1 DSL的基本设计

6.4.2 DSL的基本实现

6.4.3 DSL的基本应用

6.4.4 一些修补

6.4.5 基于严格模式的一些修补

第7章 一般性的动态函数式

语言技巧

7.1 消除代码的全局变量名占用

7.2 一次性的构造器

7.3 对象充当识别器

7.4 识别new运算进行的构造器调用

7.5 使用直接量及其包装类快速调用对象方法

7.6 三天前是星期几

7.7 使用对象的值含义来构造复杂对象

7.8 控制字符串替换过程的基本模式

7.9 实现二叉树

7.10 将函数封装为方法

7.11 使用with语句来替代函数参数传递

7.12 使用对象闭包来重置重写

7.13 构造函数参数

7.14 使用更复杂的表达式来消减if语句

7.15 利用钩子函数来扩展功能

7.16 安全的字符串

附录A 术语表

附录B 主要引擎的特性差异列表

附录C 附图

附录D 参考书目

附录E 本书各版次主要修改

章节摘录

版权页：插图：我最开始做的网页只用于展示信息，是一个个单纯的、静态的网页，并通过一些超链接连接起来。

当时网页开发的环境并不好（像现在的Dreamweaver这类程序，那时只能是梦想），因此我只能用记事本（notepad.exe）来写HTML。

当时显示这些htm文件的浏览器就是Netscape Navigator3。

我很快就遇到了麻烦，因为P & J的朋友说希望让浏览网页的用户们能做更多的事，例如搜索等操作。

我笑着说：“如果在电子公告板上，写段脚本就可以了；但在互联网上面，却要做很多的工作。

”事实上我并不知道要做多少的工作。

我随后查阅的资料表明：不但要在网页中放一些表单让浏览者提交信息，还要在网站的服务器上写些代码来响应这些提交信息。

我向那位先生摊开双手，说：“如果你真的想要这样做，那么我们可能需要三个月，或者更久。

因为我还必须学习一些新鲜的玩意儿才行。

”那时的触网者，对这些“新鲜的玩意儿”的了解还几乎是零。

因此，这个想法很自然地搁置了。

而我后来（1997年）被调到成都，终于有更多的机会接触Internet，而且浏览器环境已经换成了Internet Explorer 4.0。

那是一个美好的时代。

通过互联网络，大量的新东西很快被传递进来。

我终于有机会了解一些新的技术名词，例如css和JavaScript。

当时（1997年12月）HTML4.0的标准已经确定，浏览器的兼容性开始变得更好，Internet Explorer（以下简称IE）也越来越有取代Netscape Navigator（以下简称NN）而一统天下的趋势。

除了这些，我还对在Delphi中进行ISAPI-CGI和ISAPI Filter开发的技术也展开了深入的学习。

1.1.2 第一段在网页中的代码1998年，我被调回到河南郑州，成为一名专职程序员，任职于当时一家开发反病毒软件的公司，主要工作是用Delphi做Windows环境下的开发。

而当时我的个人兴趣之一，就是“做一个个人网站”。

那时大家都对“做主页”很感兴趣，我的老朋友傅贵就专门写了一套代码，以方便普通互联网用户将自己的主页放到“个人空间”里。

同时，他还为这些个人用户提供了公共的BBS程序和一些其他的服务器端代码。

但我并不满足于这些，我满脑子想的是做一个“自己的网站”。

编辑推荐

《JavaScript语言精髓与编程实践(第2版)》编辑推荐：以JavaScript视角看整个计算机语言的世界，小角度引来的大话题，难得的实践派写书风格，对语法认识细致入微，这是一本硬书，在技术原创书中，属于稀有“异数”之作。

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>