

<<JavaScript设计模式>>

图书基本信息

书名：<<JavaScript设计模式>>

13位ISBN编号：9787115191281

10位ISBN编号：711519128X

出版时间：2008

出版单位：人民邮电出版社

作者：Ross Harmes,Dustin Diaz

页数：249

译者：谢廷晟

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>

<<JavaScript设计模式>>

前言

设计模式对于程序员来说并不是一个陌生话题。在Erich Gamma等人合著的经典著作《设计模式》出版之后，十几年间陆续出现了许多这方面的专著。不过它们大都结合Java和C++等传统的面向对象语言进行讲解，而讲述设计模式在动态语言中的实现的书则较为罕见。在早期的JavaScript编程实践中，这种语言只被用于做点为网页涂脂抹粉的小差事；程序的规模很小，也很简单。那个时候恐怕没有人会想到把设计模式用到这种“玩具语言”编写的程序中。随着Ajax技术的兴起，Web应用的许多逻辑都从服务器端转移到客户端执行，客户端JavaScript程序的作用越来越重要，其规模和复杂程度也越来越大，人们也越来越多地把面向对象方法应用到JavaScript程序设计中。在此背景下，有许多人开始研究设计模式在JavaScript程序设计中的应用，网上也陆续出现了一些关于这个话题的零星讨论。但是到目前为止，系统地探讨面向对象的程序设计模式在JavaScript语言中的实现的书，只此一本。（Michael Mahemof所著的《Ajax设计模式》一书总结的是运用Ajax技术开发Web应用的各种设计模式，虽然也涉及大量JavaScript编程，但它与本书关注的焦点不同。本书讨论的是一些通用的面向对象设计模式在JavaScript中的实现，属于更基础性的东西，它们不仅仅适用于Web客户端编程。）JavaScript这种语言与Java等传统的面向对象语言有很大的不同。它的动态性、词法作用域和基于原型的继承机制等特点可能会让很多初次接触它的程序员都有点不习惯，而且由于语言设计上的一些不完善，许多在传统面向对象语言中只是举手之劳的事在JavaScript却不得不依靠hack手法来实现。这也许就是那些已经熟知设计模式在Java等语言中实现方式的程序员也需要本书的原因。本书第一部分着重讲述了面向对象技术在JavaScript中的实现方法。这对于对JavaScript只有过初步了解的人非常有用（当然，本书不适合对JavaScript一窍不通的读者。他们应该先找一本，JavaScript基础教材来看看，比如人民邮电出版社出版的《JavaScript基础教程》）。Java和C++编程老手们在学完这部分内容之后，想必应该能够在JavaScript程序设计中自行应用各种经典的设计模式了。不过不同的人可能会有一些不同的做法，因此继续看看本书第二部分，借鉴一下作者的方法也不无益处。对于那些从未学过设计模式的JavaScript程序员来说，本书的重要性更是毋庸置疑。不过，坦率地说，要想深入学习设计模式仅看本书是不够的。取代Gamma等人的《设计模式》并不是本书的目标。

<<JavaScript设计模式>>

内容概要

《JavaScript设计模式》共有两部分。

第一部分给出了实现具体设计模式所需要的面向对象特性的基础知识，主要包括接口、封装和信息隐藏、继承、单体模式等内容。

第二部分则专注于各种具体的设计模式及其在JavaScript语言中的应用，主要介绍了工厂模式、桥接模式、组合模式、门面模式等几种常见的模式。

为了让每一章中的示例都尽可能地贴近实际应用，书中同时列举了一些JavaScript程序员最常见的任务，然后运用设计模式使其解决方案变得更模块化、更高效并且更易维护，其中较为理论化的例子则用于阐明某些要点。

《JavaScript设计模式》适合各层次的Web前端开发人员阅读和参考，也适合有C++/Java/C#背景的服务器端程序员学习。

<<JavaScript设计模式>>

作者简介

Ross Harmes, 资深Web程序员, 有10多年编程经验。
现任Yahoo前端工程师。
他是开源图片博客软件Birch的开发者。
Blog地址为Http : //tecrhfoolery.com。

Dustin Diaz, 资深Web程序员, 现任Google用户界面工程师。
新一代JavaScript框架DEDIChain (兼具jQuery和YUI的优势) 的开发者。
他还是一位中长跑健将, 800米跑曾经在全美国排名第13。
拥有西班牙语学士学位。
个人网站http : //dustindiaz.com。

<<JavaScript设计模式>>

书籍目录

第一部分 面向对象的JavaScript第1章 富有表现力的JavaScript1.1 JavaScript的灵活性1.2 弱类型语言1.3 函数是一等对象1.4 对象的易变性1.5 继承1.6 JavaScript中的设计模式1.7 小结第2章 接口2.1 什么是接口2.1.1 接口之利2.1.2 接口之弊2.2 其他面向对象语言处理接口的方式2.3 在JavaScript中模仿接口2.3.1 用注释描述接口2.3.2 用属性检查模仿接口2.3.3 用鸭式辨型模仿接口2.4 本书采用的接口实现方法2.5 Interface类2.5.1 Interface类的使用场合2.5.2 Interface类的用法2.5.3 示例：使用Interface类2.6 依赖于接口的设计模式2.7 小结第3章 封装和信息隐藏3.1 信息隐藏原则3.1.1 封装与信息隐藏3.1.2 接口扮演的角色3.2 创建对象的基本模式3.2.1 门户大开型对象3.2.2 用命名规范区别私用成员3.2.3 作用域、嵌套函数和闭包3.2.4 用闭包实现私用成员3.3 更多高级对象创建模式3.3.1 静态方法和属性3.3.2 常量3.3.3 单体和对象工厂3.4 封装之利3.5 封装之弊3.6 小结第4章 继承4.1 为什么需要继承4.2 类式继承4.2.1 原型链4.2.2 extend函数4.3 原型式继承4.3.1 对继承而来的成员的读和写的不对等性4.3.2 clone函数4.4 类式继承和原型式继承的对比4.5 继承与封装4.6 掺元类4.7 示例：就地编辑4.7.1 类式继承解决方案4.7.2 原型式继承解决方案4.7.3 掺元类解决方案4.8 继承的适用场合4.9 小结第5章 单体模式5.1 单体的基本结构5.2 划分命名空间5.3 用作特定网页专用代码的包装器的单体5.4 拥有私用成员的单体5.4.1 使用下划线表示法5.4.2 使用闭包5.4.3 两种技术的比较5.5 惰性实例化5.6 分支5.7 示例：用分支技术创建XHR对象5.8 单体模式的适用场合5.9 单体模式之利5.10 单体模式之弊5.11 小结第6章 方法的链式调用6.1 调用链的结构6.2 设计一个支持方法链式调用的JavaScript库6.3 使用回调从支持链式调用的方法获取数据6.4 小结第二部分 设计模式第7章 工厂模式7.1 简单工厂7.2 工厂模式7.3 工厂模式的适用场合7.3.1 动态实现7.3.2 节省设置开销7.3.3 用许多小型对象组成一个大对象7.4 示例：XHR工厂7.4.1 专用型连接对象7.4.2 在运行时选择连接对象7.5 示例：RSS阅读器7.6 工厂模式之利7.7 工厂模式之弊7.8 小结第8章 桥接模式8.1 示例：事件监听器8.2 桥接模式的其他例子8.3 用桥接模式联结多个类8.4 示例：构建XHR连接队列8.4.1 添加核心工具8.4.2 添加观察者系统8.4.3 开发队列的基本框架8.4.4 实现队列8.4.5 哪些地方用了桥接模式8.5 桥接模式的适用场合8.6 桥接模式之利8.7 桥接模式之弊8.8 小结第9章 组合模式9.1 组合对象的结构9.2 使用组合模式9.3 示例：表单验证9.3.1 汇合起来9.3.2 向FormItem添加操作9.3.3 向层次体系中添加类9.3.4 添加更多操作9.4 示例：图片库9.5 组合模式之利9.6 组合模式之弊9.7 小结第10章 门面模式10.1 一些你可能已经知道的门面元素10.2 JavaScript库的门面性质10.3 用作便利方法的门面元素10.4 示例：设置HTML元素的样式10.5 示例：设计一个事件工具10.6 实现门面模式的一般步骤10.7 门面模式的适用场合10.8 门面模式之利10.9 门面模式之弊10.10 小结第11章 适配器模式11.1 适配器的特点11.2 适配原有实现11.3 示例：适配两个库11.4 示例：适配电子邮件API11.4.1 用适配器包装Web邮件API11.4.2 从fooMail转向dedMail11.5 适配器模式的适用场合11.6 适配器模式之利11.7 适配器模式之弊11.8 小结第12章 装饰者模式12.1 装饰者的结构12.1.1 接口在装饰者模式中的角色12.1.2 装饰者模式与组合模式的比较12.2 装饰者修改其组件的方式12.2.1 在方法之后添加行为12.2.2 在方法之前添加行为12.2.3 替换方法12.2.4 添加新方法12.3 工厂的角色12.4 函数装饰者12.5 装饰者模式的适用场合12.6 示例：方法性能分析器12.7 装饰者模式之利12.8 装饰者模式之弊12.9 小结第13章 享元模式13.1 享元的结构13.2 示例：汽车登记13.2.1 内在状态和外在状态13.2.2 用工厂进行实例化13.2.3 封装在管理器中的外在状态13.3 管理外在状态13.4 示例：Web日历13.4.1 把日期对象转化为享元13.4.2 外在数据保存在哪里13.5 示例：工具提示对象13.5.1 未经优化的Tooltip类13.5.2 作为享元的Tooltip13.6 保存实例供以后重用13.7 享元模式的适用场合13.8 实现享元模式的一般步骤13.9 享元模式之利13.10 享元模式之弊13.11 小结第14章 代理模式14.1 代理的结构14.1.1 代理如何控制对本体的访问14.1.2 虚拟代理、远程代理和保护代理14.1.3 代理模式与装饰者模式的比较14.2 代理模式的适用场合14.3 示例：网页统计14.4 包装Web服务的通用模式14.5 示例：目录查找14.6 创建虚拟代理的通用模式14.7 代理模式之利14.8 代理模式之弊14.9 小结第15章 观察者模式15.1 示例：报纸的投送15.1.1 推与拉的比较15.1.2 模式的实践15.2 构建观察者API15.2.1 投送方法15.2.2 订阅方法15.2.3 退订方法15.3 现实生活中的观察者15.4 示例：动画15.5 事件监听器也是观察者15.6 观察者模式的适用场合15.7 观察者模式之利15.8 观察者模式之弊15.9 小结第16章 命令模式16.1 命令的结构16.1.1 用闭包创建命令对象16.1.2 客户、调用者和接收者16.1.3 在命令模式中使用接口16.2 命令对象的类型16.3 示例：菜单项16.3.1 菜单组合对象16.3.2 命令类16.3.3 汇合起来16.3.4 添加更多菜单项16.4 示例：取消操作和命令日志16.4.1 使用命令

<<JavaScript设计模式>>

日志实现不可逆操作的取消16.4.2 用于崩溃恢复的命令日志16.5 命令模式的适用场合16.6 命令模式之利16.7 命令模式之弊16.8 小结第17章 职责链模式17.1 职责链的结构17.2 传递请求17.3 在现有层次体系中实现职责链17.4 事件委托17.5 职责链模式的适用场合17.6 图片库的进一步讨论17.6.1 用职责链提高组合对象的效率17.6.2 为图片添加标签17.7 职责链模式之利17.8 职责链模式之弊17.9 小结索引

<<JavaScript设计模式>>

章节摘录

在事件驱动的环境中，比如浏览器这种持续寻求用户关注的环境中，观察者模式〔又名发布者—订阅者（publisher-subscriber）模式〕是一种管理人与其任务之间的关系（确切地讲，是对象及其行为和状态之间的关系）的得力工具。

用JavaScript的话来说，这种模式的实质就是你可以对程序中某个对象的状态进行观察，并且在其发生改变时能够得到通知。

观察者模式中存在两个角色：观察者和被观察者。

本书一般倾向于称其为发布者和订阅者。

这种模式在JavaScript中有几种不同的实现方式，本章将对其中的一些实现方式进行考察。

不过我们首先要说明一下发布者和订阅者这两种角色。

下一节的例子以报业为例说明了观察者模式的工作方式。

15.1 示例：报纸的投送 在报纸行业中，发行和订阅的顺利进行有赖于一些关键性的角色和行为。

首先是读者。

他们都是订阅者（subscriber），是与你我一样的人。

我们消费数据并且根据读到的消息做出反应。

我们可以选择自己的居住地点，让报社把报纸送到自己家中。

这个活动中的另一个角色是发行方（publisher）。

他们负责出版诸如San Francisco Chronicle、New York Times和Sacramento Bee这样的报纸。

确定了各方的身份之后，我们就可以分析每一方的职责所在。

作为报纸的订阅者，我们有一些事要做。

数据到来的时候我们收到通知。

我们消费数据。

然后我们根据数据做出反应。

只要报纸到了订阅者手中，他们就可以自行处置。

有些人读完之后会将其扔在一边，有些人会向朋友或家人转述其中的新闻，甚至还有一些人会把报纸送回去。

总而言之，订阅者要从发行方接收数据。

发行方则要发送数据。

在本例中，发行方也是投送方（deliver）。

一般说来，一个发行方很可能有许多订阅者，同样，一个订阅者也很可能会订阅多家报社的报纸。

问题的关键在于，这是一种多对多的关系，需要一种高级的抽象策略，以便订阅者能够彼此独立地发生改变，而发行方能够接受任何有消费意向的订阅者。

15.1.1 推与拉的比较 对于报社来说，只为给几个订阅者投送报纸就满世界跑是不划算的。而纽约市的居民也不可能特意飞到旧金山去拿自己订的San Francisco Chronicle，要知道这份报纸可以直接投送到他们家门口。

订阅者要想拿到报纸的话有两种投送方式可选：推或拉。

在推环境中，发行方很可能会雇佣投送人员四处送报。

换句话说，他们把自己的报纸推出去，让订阅者收取。

在拉环境中，规模较小的本地报社可能会在订阅者家附近的街角提供自己的数据，供订阅者“拉”。那些成长型发行方没有足够的资源进行大规模投送，因此采用拉方案，让订阅者到当地的杂货店或自动售货机那里“拿”报，对于它们来说往往是个优化投送环节的好办法。

15.1.2 模式的实践 在JavaScript中有多种方法可以实现发布者—订阅者模式。

在展示那些示例之前，我们先确保各种角色的扮演者（对象）及其行为（方法）都已就绪。

<<JavaScript设计模式>>

媒体关注与评论

“本书道前人所未道，引导你从编写代码进化为设计代码。
书中绝大部分示例代码都来自YUI等实战项目，并进行了深入剖析。
强烈推荐。

”——Nicholas C.Zakas，著名JavaScript专家，Yar100前端工程师，畅销书《JavaScript高级程序设计》作者
“毫不夸张地说，这是我有生以来读到的最好的一本JavaScript图书。
作者讲授了大量独门的专家经验。

”——Mostafa Farghaly，埃及程序员

<<JavaScript设计模式>>

编辑推荐

从这里开始，真正掌握JavaScript的精髓。

Google和Yahoo专家联手揭秘世界顶尖公司的技术内幕 Amazon全五星誉图书 Web应用取代桌面程序的时代已经到来！

作为Web前端的核心技术，JavaScript的重要性不言而喻，它有望成为下一代统治性程序语言。但由于业界长期的误解和滥用，也有不少人仍然对此半信半疑。

那么，JavaScript到底能否当此大任呢？

本书中，Google和Yahoo公司的两位资深Web专家对此给出了掷地有声的肯定回答。作者针对常见的开发任务，从YUI等实战代码中取材，提供了专家级的解决方案，不仅透彻剖析了JavaScript中的面向对象编程。

而且深入探讨了如何用JavaScript实现以前只在服务器端应用的设计模式。

如何根据实际场景选择恰当的设计模式，开发出高质量的企业级代码。

本书充分证明：JavaScript不仅毫不逊色于其他高级语言，已经是一种成熟且强大的面向对象语言。

而且还拥有Java和C++等语言不具备的面向未来的特性，因此更加灵活、更富于表现力。

无论是前端工程师-还是服务器端程序员，通过本书都将使自己的JavaScript功力提升到前所未有的高度。

<<JavaScript设计模式>>

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>