JavaScript Patterns

JavaScript 模式

O'REILLY® | YAHOO! PRESS
中国电力出版社

Stoyan Stefanov 著
陈新 译

JavaScript模式

Stoyan Stefanov 著
陈新 译

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Sebastopol • Tokyo

O'Reilly Media, Inc.授权中国电力出版社出版

中国电力出版社

图书在版编目（CIP）数据

JavaScript模式/（美）斯特凡洛夫（Stefanov, S.）著；陈新译. －北京：中国电力出版社，2012.4

书名原文：JavaScript Patterns

ISBN 978-7-5123-2923-2

I. J… II. ①斯… ②陈… III. ①Java语言－程序设计 IV. ①TP312

中国版本图书馆CIP数据核字（2012）第073567号

北京市版权局著作权合同登记

图字：01-2010-7922号

书　　名/ JavaScript模式
书　　号/ ISBN 978-7-5123-2923-2
责任编辑/ 刘炽
封面设计/ Karen Montgomery，张健
出版发行/ 中国电力出版社（http://www.cepp.sgcc.com.cn）
地　　址/ 北京市东城区北京站西街19号（邮政编码100005）
经　　销/ 全国新华书店
印　　刷/ 航远印刷有限公司印刷
开　　本/ 787毫米×980毫米　16开本　13.75印张　253千字
版　　次/ 2012年7月第一版　2012年7月第一次印刷
印　　数/ 0001－3000册
定　　价/ 38.00元（册）

目录

前言

模式是针对普遍问题的解决方案。更进一步地说，模式是解决一类特定问题的模板。

模式会有助于将问题分解为一个个像积木一样的小模块，并集中精力处理从各种烦琐的细节中提炼出来的该问题所特有的部分。

模式通过提供一个通用的词典来帮助我们更好地进行交流，因此学习并识别模式是很重要的。

目标读者

这本书不是一本入门级的书，而是适用于希望将自身的JavaScript技巧提高到一个新层次的专业的开发人员和程序员。

本书并未讨论一些基本的原理（例如循环、条件语句和闭包等）。如果发现需要复习这些主题，请参考推荐读物列表。

同时，一些主题（例如创建和提升对象）可能在本书中看起来太基础了，但是我认为它们对于充分发挥该语言的能力有很大作用，因此从模式的视角来讨论了这些主题。

如果正在寻找更好的实践和更强大的模式来编写更优、更好的可维护性和更强健的JavaScript代码，那么这本书正是您所要的。

本书使用的排版约定

本书使用如下排版约定：

斜体（Italic）

用来表示新术语、URL、Email地址、文件名、文件扩展名等。

等宽字体（Constant width）

用来表示程序列表，同时在段落中引用的程序元素（例如变量、函数名、数据库、数据类型、环境变量、声明和关键字等）也用该格式表示。

等宽粗体（Constant width bold）

用于表示需要用户逐字符输入的命令或其他文本。

等宽斜体（Constant width italic）

用于表示应该以用户提供的值或根据上、下文决定的值加以替换的文本。

如何使用代码范例

本书可以帮助您完成工作。一般来说，可以在自己的程序和文档中使用本书的代码。除非将重新编译代码中重要的部分，您是不需要联系我们来获得授权的。举例来说，使用书中几段程序来编程是不需要获得我们的授权的，但是销售或发布O'Reilly出版书籍中配套光盘中代码是需要授权的。通过引用本书中范例来回答问题是不需要授权的，而将本书中重要部分的范例代码整合到您产品的文档中是需要授权的。

我们感谢您在使用我们代码的时候给出引用说明，但这不是硬性规定。一个引用说明通常包括了标题、作者、出版社和ISBN。举例来说，本书的引用说明“JavaSc Patterns, by Stoyan Stefanov（O'Reilly）. Copyright 2010 Yahoo!, Inc.,9780596806750.”

如果您不确定所使用的范例代码是否超出了上面给定的权限，可以随时通过电子邮件联系我们。我们的电子邮件地址是*permissions@oreilly.com*。

如何联系我们

请将关于本书的意见和问题发送给出版社：

美国：

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472

中国：

北京市西城区西直门南大街2号成铭大厦C座807室（100035）

奥莱利技术咨询（北京）有限公司

我们为本书提供了网页，该网页上面列出了勘误表、范例和任何其他附加的信息。您可以访问如下网页获得：

原文书：

http://www.oreilly.com/catalog/9780596806750

中文书：

http://www.oreilly.com.cn/index.php?func=book&isbn=978-7-5123-2923-2

要询问技术问题或对本书提出建议，请发送电子邮件至：

bookquestions@oreilly.com

要获得更多关于我们的书籍、会议、资源中心和O'Reilly网络的信息，请参见我们的网站：

http://www.oreilly.com
http://www.oreilly.com.cn

致谢

我一直都受惠于社区的一些难以置信的评论者，他们分享自己知识和能力，从而使得本书变得更好。他们的博客和推特具有让人敬畏的、敏锐的观察力，并包含了伟大的思想和模式。

- Dmitry Soshnikov (*http://dmitrysoshnikov.com, @DmitrySoshnikov*)
- Andrea Giammarchi (*http://webreflection.blogspot.com, @WebReflection*)
- Asen Bozhilov (*http://asenbozhilov.com, @abozhilov*)
- Juriy Zaytsev (*http://perfectionkills.com, @kangax*)
- Ryan Grove (*http://wonko.com, @yaypie*)
- Nicholas Zakas (*http://nczonline.net, @slicknet*)
- Remy Sharp (*http://remysharp.com, @rem*)
- Iliyan Peychev

感谢

本书中的一些模式是作者在实践和学习了一些流行的JavaScript库（例如jQuery和YUI）的基础上提出的，但大部分模式还是由JavaScript社区提出并描述的。因此，本书是众多开发者集体工作的成果。为了不打断关于本书历史和信誉的叙述，我们在本书的附属网站（*http://www.jspatterns.com/book/reading/*）上给出了参考文献和建议的附加读物列表。

如果我在参考文献中遗漏了某篇优秀的原始文献，请接受我的道歉并告知我，以便我可以将其增加到*http://www.jspatterns.com*的在线列表中。

推荐读物

本书不是一本入门读物，因此没有介绍一些基础的主题，例如循环和条件选择等。如果您需要获取关于本语言更多的细节，可以参考如下推荐读物：

- Object-Oriented JavaScript by yours truly (Packt Publishing)
- JavaScript: The Definitive Guide by David Flanagan (O'Reilly)
- JavaScript: The Good Parts by Douglas Crockford (O'Reilly)
- Pro JavaScript Design Patterns by Ross Hermes and Dustin Diaz (Apress)
- High Performance JavaScript by Nicholas Zakas (O'Reilly)
- Professional JavaScript for Web Developers by Nicholas Zakas (Wrox)

第1章

简介

JavaScript是一门基于Web的语言。它最初是作为在网页中操作一些特定类型元素（例如图像和表格字段）的方法，但现在该语言发展很快，远远超出了其预期。除了作为客户端浏览器所使用的脚本，现在JavaScript还可用于正在增加的更加多样化的平台编程。例如可以用于编写服务端代码（使用.NET或Node.js）、应用程序扩展（例如为Firefox和Photoshop编写扩展）、移动应用程序和命令行脚本。

JavaScript也是一门与众不同的语言。它没有类，并且函数是用于很多任务的顶层类对象。最初很多程序员认为该语言效率低下，但近年来该观念有所改变。有趣的是，例如Java和PHP这类语言开始增加闭包和匿名函数特性，而JavaScript程序员早就已经享用了这些特性。

JavaScript是一门动态性较强的语言，通过配置可以使其看上去和感觉起来像您过去习惯的其他语言一样。但接受JavaScript与其他语言的不同，并学习其特定的模式可能是更好的方法。

模式

广义上模式是指“重现事件或者对象的主题……它是一个可以用来产生其他事物的模板或者模型”（引自维基百科，*http://en.wikipedia.org/wiki/Pattern*）。

在软件开发过程中，模式是指一个通用问题的解决方案。一个模式不仅仅是一个可以用来复制粘贴的代码解决方案，更多地是提供了一个更好的实践经验、有用的抽象化表示和解决一类问题的模板。

学习和识别模式是非常重要的，理由如下：

- 通过学习模式，可以帮助我们使用经过实践证明有效的经验来编写代码，而无需做很多无用的工作。
- 模式提供了某种程度上的抽象。大脑在一定的时间内仅能记住一定数量的内容，因此当思考更复杂的问题时，使用模式可以让您集中精力去用已有的模式来解决该问题，而不需被一些低层次的细节所困扰。
- 模式可以改善开发者和开发团队之间的交流，通常开发者是采取远程交流而不是面对面进行交流。为一些编码技术或方法贴上标签并命名，可以很方便地确保双方能够了解对方确切想表达的意思。举例来说，如果需要表达“用大括号括上一个函数，并在刚刚定义的这个函数的结束位置放置另一个括号来调用该函数”这层意思，直接使用“立即函数”这样的称法可能更为简单明了。

本书主要讨论如下三种类型的模式：

- 设计模式。
- 编码模式。
- 反模式。

设计模式最初是在“Gang of Four”（该名称是根据本书四名作者的姓名而来的）这本书中定义的，该书早在1994年就出版了，当时的书名是“设计模式：可复用面向对象软件的基础”。这些设计模式的范例主要包括单件（singleton）、工厂方法（factory）、装饰（decorator）、观察者（observer）等。尽管设计模式是与语言无关的，但相对JavaScript的设计模式来说，过去设计模式主要是从强类型语言的视角进行研究的，例如C++语言和Java语言等。有时候将这些模式严格地应用于JavaScript这样弱类型动态语言是没有必要的。有时候这些模式是适用于处理强类型语言的一些本性和基于类的继承。在JavaScript语言中，可能有其他更好、更简单的选择。本书将在第7章讨论几种设计模式在JavaScript中实现的方法。

代码模式更为有趣，这是JavaScript特有的模式，它提供了关于该语言独特的很好的体验，例如various uses of functions。JavaScript的代码模式是本书的主题。

在本书中还会学到反模式。从名称上看，反模式有一些消极甚至是刺耳，但是事实并非如此。一个反模式并不是一个bug，或者是编码错误，它仅仅是常见的、引发的问题比解决的问题更多的一种方法。在代码中使用注释明确标记出反模式。

JavaScript：基本概念

首先快速地复习为后续章节提供背景支持的一些重要概念。

面向对象

JavaScript是一门面向对象的语言，很多程序员对此会很惊讶，因为他们之前都没看到JavaScript语言的这一特性。看到的任何一段JavaScript代码都很有可能是一个对象。只有五种基本类型不是对象：数值类型、字符串类型、布尔类型、空类型和未定义类型。其中前三个类型有对应的以基本类型封装形式体现的对象表示（将在接下来的一个章节进行讨论）。数值类型、字符串类型和布尔类型的值可以通过程序员或者位于幕后的JavaScript解析器来实现向对象的转换。

函数实际上也是对象，函数有属性和方法。

在任何一门语言中最简单的事情就是定义一个变量。在JavaScript中，一旦定义好了变量，同时也就已经正在处理对象了。首先，该变量会自动成为内置对象的一个属性，成为激活对象（如果该变量是一个全局变量，那么该变量会成为全局对象的一个属性）。第二，该变量实际上也是伪类，因为它拥有其自身的属性（称为attributes），该属性决定了该变量是否可以被修改、被删除和在一个for-in循环中进行枚举。这些属性在ECMAScript3中没有直接对外提供，但在第5版本的ECMAScript中，提供了一个特殊的描述符方法来操纵这些属性。

那么对象是什么东西呢？因为它们需要做很多事情，所以这些对象必须十分特殊。实际上对象是十分简单的。一个对象仅仅是一个容器，该容器包含了命名的属性、键－值对（大多数）的列表。这里面的属性可以是函数（函数对象），这种情形下我们称其为方法。

关于创建的对象的另外一件事情是可以在任意时间修改该对象（尽管ECMAScript5引入了API来防止突变）。可以对一个对象执行添加、删除和更新它的成员变量。如果关注隐私和访问，在模式方面也有对应的内容。

最后需要记住的是对象主要有两种类型：

原生的（Native）
: 在ECMAScript标准中有详细描述。

主机的（Host）
: 在主机环境中定义的（例如浏览器环境）。

原生的对象可以进一步分为内置对象（例如数组、日期对象等）和用户自定义对象（例如var o={};）等。

主机对象包含windows对象和所有的DOM对象。如果还不确定使用的是否是主机对象，可以尝试在不同的、无浏览器的环境下运行该代码，如果该代码能正确地运行，那么应该使用的是原生的对象。

没有类

可能在本书中将会多次看到这样的语句：在JavaScript中没有类。这对使用其他语言的老练的程序员可能是一个比较新颖的概念，因此需要多次重复来忘记类这个概念，并接受JavaScript只处理对象这样一个特点。

不使用类的做法可以使得编程更为简洁。在创建一个对象的时候，无需先拥有一个类。请考虑如下类Java的对象创建方式：

```
// 创建Java对象
HelloOO hello_oo=new HelloOO();
```

在需要创建一个简单的对象的时候，重复以上操作看起来是额外的开销。我们往往希望让对象更为简洁。

在JavaScript中可以在需要的时候创建一个空对象，然后开始为该对象添加感兴趣的成员变量。可以为该对象添加基本类型、函数和其他对象来作为该对象的属性。一个“空对象”实际上并不是完全空白的，它实际上是包含有一些内置的属性，但是没有其自身的属性。将在接下来的章节详细讨论该问题。

在“Gang of Four”这本书中的一条通用规则是：“尽量多使用对象的组合，而不是使用类的继承”。这句话的意思是通过已有的对象组合来获取新对象，是比通过很长的父－子继承链来创建新的对象更好的一种方法。在JavaScript中可以很简单地实践该建议，之所以这么简单是因为在JavaScript中没有类，因此使用对象的组合是唯一可以采取的方法。

原型（Prototypes）

JavaScript没有继承，尽管这是重用代码的一种方式（后面将通过一整章来介绍代码重用）。可以使用多种方法来实现继承，这里通常使用原型。原型是一个对象（别惊讶），并且创建的每一个都会自动获取一个Prototypes属性，该属性指向一个新的空对象。该对象几乎等同于采用对象字面量或Obect()创建的对象，区别在于它的constructor属性指向了所创建的函数，而不是指向内置的Object()函数。可以为该空对

象增加成员变量，以后其他对象也可以从该对象继承并像使用自己的属性一样使用该对象的属性。

本书将在后续章节详细讨论继承的细节，但现在请各位记住原型就是一个对象（不是一个类，也不是其他特殊的元素），每一个函数都有Prototype属性。

环境

JavaScript需要运行环境来执行。通常JavaScript是在浏览器中执行的，但是这不是唯一的运行环境。本书中介绍的模式大部分是和核心JavaScript（ECMAScript）相关的，因此它们是与环境无关的。除了以下两种情况以外：

- 第8章中特定针对浏览器的模式。
- 其他示范模式的实际应用程序范例。

环境会提供自身的主机对象，该对象在ECMAScript标准中没有定义，可能会带来没有特别提到的和不确定的行为。

ECMAScript 5

核心的JavaScript编程语言（不包含DOM、BOM和额外的主机对象）是基于ECMAScript标准（缩写是ES）。该标准的第3版是在1999年被官方所接受，并且是当前浏览器所使用的标准。该标准的第4版被放弃了，第5版在2009年12月得到通过，这相比第3版过去了10年。

第5版为ECMAScript增加了一些新的内置对象、方法和属性，但是最重要的是增加了所谓的strict模式，该模式实际上可以从JavaScript语言中移除某些特性，使得程序更为简洁和不容易出错。例如with语句的用法在这几年中有很多争议。现在在ES5的strict模式中将会引发错误，尽管在非strict模式中能正常运行。strict模式是通过一个普通的字符串来触发的，在该语言的较早的实现方式中将会简单地忽略该代码。这就意味着通过使用strict模式，可以实现向后兼容性，因为在之前不能理解该代码的浏览器中，它不会引起错误。

在一个作用域（可以是函数作用域、全局作用域或者在将字符串的起始位置传递给eval()）中，可以使用如下字符串：

```
Function my(){
    "use strict";
    // 函数的其余部分……
}
```

这就意味着函数中的代码是在ECMAScript语言的strict子集中运行。对于之前的浏览器，这仅仅是一个字符串，并没有分配给任何变量，因此不会被使用，进而不会导致错误。

在本语言的未来计划中将只允许使用strict模式。考虑到ES5是一个过渡版本，本语言现在鼓励开发者使用strict模式编写代码，但不强制要求。

本书将不探讨ES5增加的特有的模式，因为在本书编写的时候并没有任何浏览器实现了ES5。但本书中的范例通过以下方式推广使用新标准：

- 确保本书中提供的范例代码不会在strict模式中引起错误。
- 避免使用和指出类似`arguments.callee`之类的构造函数。
- 调用在ES5中有等价替代的ES3模式，例如`Object.create()`。

JSLint

JavaScript是一门解析型语言，没有静态编译时检查。因此有可能仅仅是因为几个拼写错误，配置一个有问题程序，而又没有意识到该问题的存在。在这种情况下JSLint就非常有用了。

JSLint（*http://jslint.com*）是Douglas Crockford编写的一个JavaScript代码质量检查工具，该工具可以检查代码，并对潜在的问题提出警告。强烈推荐使用JSLint运行您的代码。在刚开始使用的时候，该工具给出的警告可能会影响到心情，但是将很快从中发现错误，并适应这个专业JavaScript程序员都应具备的一个很重要的习惯。如果JSLint没有对代码给出任何警告，那么将会对代码更为自信，知道自己没有在匆忙中有所遗漏或语法错误。

从下一章开始将会多次提到JSLint。本书中所有示范代码都成功通过了JSLint检查（使用在本书编写期间的默认设置），除了偶尔有一些代码是用于展示反模式的（这些代码会清楚地标示出来）。

在默认设置中，JSLint希望用户最好在代码中使用strict模式。

Console

全书都使用了`Console`对象。该对象不是JavaScript语言的一部分，而是指当今大多数浏览器都提供的一个运行环境。在Firefox浏览器中，使用的是Firebug这个扩展工具。Firebug的控制台提供了一个图形化界面，可以方便用户快速地输入和测试小段的