TURING 图灵程序设计丛书

Apress®

Pro JavaScript Development

Coding, Capabilities, and Tooling

精通JavaScript开发

前端开发人员进阶首选

掌握JavaScript语言各种细节，轻松编写出易维护、易扩展的高质量代码

【英】Den Odell 著

邝健威 厉海洋 译

中国工信出版集团

人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

Pro JavaScript Development

Coding, Capabilities, and Tooling

精通JavaScript开发

【英】Den Odell 著

邝健威 厉海洋 译

人民邮电出版社

北京

图书在版编目（CIP）数据

精通JavaScript开发 /（英）奥德尔（Odell,D.）著；
邝健威，厉海洋译. -- 北京 ：人民邮电出版社，2015.9
（图灵程序设计丛书）
ISBN 978-7-115-40255-4

Ⅰ. ①精… Ⅱ. ①奥… ②邝… ③厉… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第201198号

内 容 提 要

本书是 JavaScript 实战指南，主要内容包括：JavaScript 性能、可靠性、稳定性和代码管理分析，面向对象代码的使用，测试和错误处理机制的构建，用 AMD 和 RequireJS 管理代码依赖，移动端、游戏和实时通信的 JavaScript 开发，等等。

本书的读者对象为具有一定经验的前端开发工程师。

◆ 著 [英] Den Odell
译 邝健威 厉海洋
责任编辑 朱 巍
责任印制 杨林杰
◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷
◆ 开本：800×1000 1/16
印张：26.25
字数：636千字 2015年9月第1版
印数：1－4 000册 2015年9月北京第1次印刷
著作权合同登记号 图字：01-2014-6315号

定价：79.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

版权声明

前　言

如果你对 JavaScript 已经有了一定了解并且打算继续学习成为一名专业的 JavaScript 开发者，那么本书就是为你而写的。在我看来，现代化的 JavaScript 开发包含三方面内容：编程技巧、语言本身提供的各种特性以及工具的使用。我认为这三者相互交织，所以并没有分三部分单独讲，而是在每章中都沿着这三个方面展开。我的目标就是让你学会利用现代化的编程技巧、语言特性和各种工具，编写出质量最高、最易维护、最易扩展和最高效的代码。

阅读本书后，你的编程能力将会得到提高；你将学到 JavaScript 语言的各种细节，比如对象、上下文、作用域、原型、继承，以及各大浏览器中新增的语言新特性。你将学到各种设计模式，了解如何对代码进行注释从而让整个项目团队都受益，以及如何提高代码运行时的性能。

你将接触到一些之前不太熟悉的语言特性，比如一些原生的 API，这些 API 有的能渲染和构造出在浏览器中运行的游戏，有的能支持免插件的视频聊天，还有的能对移动设备开发提供专门支持。

如今，开发者们正越来越多地利用起各种工具和自动化手段，来帮助他们改善开发流程，提高代码质量。在本书中，你将学会怎样检查代码质量，怎样从代码自动生成文档网站，怎样针对你的代码运行一系列的任务以改进每天的工作流，如何将代码打包发行，以及如何利用各大浏览器内置的开发工具，在运行时进行代码的调试和分析。

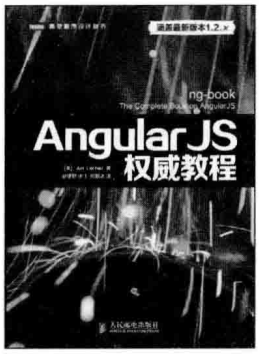
读完本书，你就应当具备成为一名专业 JavaScript 开发者所需的知识和经验了，将有能力开发出高质量、可维护、可扩展和高性能的应用。

让我们开始吧！

延 展 阅 读

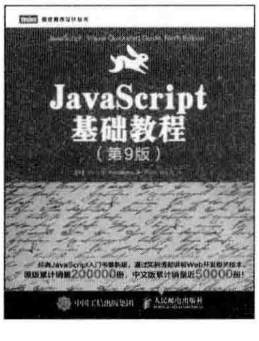
作为JavaScript技术经典名著，《JavaScript高级程序设计（第3版）》承继了之前版本全面深入、贴近实战的特点，在详细讲解了JavaScript语言的核心之后，条分缕析地为读者展示了现有规范及实现为开发Web应用提供的各种支持和特性。

书号：978-7-115-27579-0
定价：99.00 元



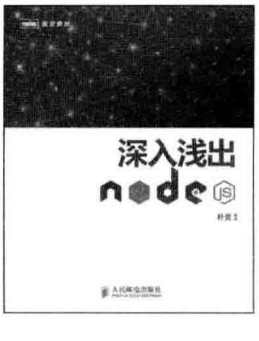
说到学习AngularJS，相信你早已厌倦了上网搜索、断续阅读的低效方式。本书堪称AngularJS领域的里程碑式著作，它以相当的篇幅涵盖了关于AngularJS的几乎所有内容，既是一部权威教程，又是一部参考指南。对于没有经验的人，本书平实、通俗的讲解，递进、严密的组织，可以让人毫无压力地登堂入室，迅速领悟新一代Web应用开发的精髓。如果你有相关经验，那本书对AngularJS概念和技术细节的全面剖析，以及引人入胜、切中肯綮的讲解，将帮助你彻底掌握这个框架，在自己职业技术修炼之路上更进一步。

书号：978-7-115-36647-4
定价：99.00 元



本书是经典JavaScript入门书，全球累计销量已超20万册。书中从JavaScript语言基础开始，分别讨论了图像、框架、浏览器窗口、表单、正则表达式、用户事件和cookie等，循序渐进地讲述了JavaScript及相关的CSS、DOM、Ajax、jQuery等技术。内容讲解透彻，图文并茂。

书号：978-7-115-38522-2
定价：69.00 元



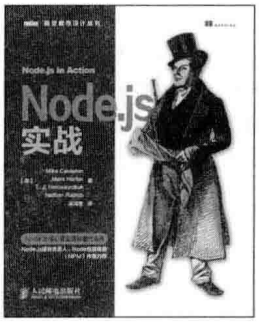
第一本深度讲解Node的图书
源码级别探寻Node的实现原理
阿里巴巴一线Node开发者最真实的经验

书号：978-7-115-33550-0
定价：69.00 元

延 展 阅 读

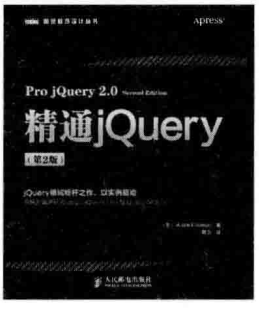
很多人对JavaScript这门语言的印象都是简单易学，很容易上手。JavaScript语言本身有很多复杂的概念，语言的使用者不必深入理解这些概念也可以编写出功能全面的应用。殊不知，这些复杂精妙的概念才是语言的精髓，即使是经验丰富的JavaScript开发人员，如果没有认真学习的话也无法真正理解它们。在本书中，我们要直面当前JavaScript开发者不求甚解的大趋势，深入理解语言内部的机制。

书号：978-7-115-38573-4
定价：49.00 元
注：中卷和下卷即将推出。



Node.js核心框架贡献者代表作，Node.js项目负责人、Node包管理器作者力荐！
本书向读者展示了如何构建产品级应用，对关键概念的介绍清晰明了，贴近实际的例子，涵盖从安装到部署的各个环节，是一部讲解与实践并重的优秀著作。通过学习本书，读者将深入异步编程、数据存储、输出模板、读写文件系统，掌握创建TCP/IP服务器和命令行工具等非HTTP程序的技术。本书同样非常适合熟悉Rails、Django或PHP开发的读者阅读学习。

书号：978-7-115-35246-0
定价：69.00 元



jQuery领域标杆之作，以实例驱动，系统全面讲解jQuery、jQuery UI以及jQuery Mobile。作为一款优秀的JavaScript框架，jQuery具有表达能力强、支持一次处理多个元素、能解决不同浏览器的兼容性问题等诸多优点，从而受到广大Web开发人员的追捧。本书是一本全面的jQuery手册，详尽介绍了jQuery库、jQuery UI和jQuery Mobile，能帮助具备一定Web开发基础知识的读者精通jQuery。

书号：978-7-115-36653-5
定价：149.00 元

由jQuery API网站维护者亲自撰写，第一版自2008上市以来，一版再版，累计重印14次，是国内首屈一指的jQuery经典著作！
注重理论与实践相结合，由浅入深、循序渐进，适合各层次的前端Web开发人员学习和参考。

书号：978-7-115-33055-0
定价：59.00 元

目　录

第1章 面向对象的JavaScript

如果你有过一些网站开发的经验，也许遇到过一些看不起JavaScript的程序员，他们认定JavaScript不是面向对象的，因而对其低看一等。作为JavaScript开发者，我们必须让大家，尤其是那些轻视JavaScript的人知道，JavaScript不但是一门货真价实的面向对象语言，而且在这方面功能还很强大。

其实那些程序员之所以小瞧JavaScript，是因为JavaScript并不遵循一些传统编程语言所确立的语法结构和编程惯例，这些语言包括C++、Java、PHP以及Objective-C。在我看来，这未必就不好，正是因为没有被强加一个严格而刻板的语法结构，所以只要方法得当，JavaScript就能释放出比传统语言更大的灵活性。

在本章中，我将介绍如何用JavaScript编写面向对象的代码，这些代码符合那些被其他语言所采纳的面向对象原则，而本章的重点在于介绍JavaScript是怎样更加灵活地完成这一任务的。另外还将介绍一些语言本身的内置对象，以及这些对象一些比较不为人知的方面。

注意 传统编程语言是指那些通过某种模板来定义和构造对象的语言，这些模板被称为类，因此而得名。

1.1 JavaScript 中的对象

JavaScript中的每个对象都是一个自成一体的实体，内部包含一个或多个相关的变量和函数，分别叫作属性和方法。对象的作用是将相关的概念或功能打成一包，这些概念或功能往往对应真实世界中的事物或者特定的软件功能。对象让代码变得更容易理解，从而也更容易阅读和编写。

1.1.1 定制对象

要创建你自己的对象并将其用于JavaScript代码中，最简单的方法就是用对象直接量标记法来定义变量。对象直接量是由一对花括号和其中的名值对组成的。通过将名值对封装在花括号内，我们就能将属性和方法赋给一个对象，具体形式如代码清单1-1所示。这里我们创建了一个对象来表示房子，该对象有两个属性和两个方法。一旦创建完成，我们就能使用点标记法对其属性和

方法进行读写了，这种标记法用一个圆点符号（.）将对象名和其属性或方法分隔开来。

代码清单1-1　使用对象直接量标记法来创建一个对象

```
var house = {
 rooms: 7,
 sharedEntrance: false,
 lock: function() {},
 unlock: function() {}
};

// 读取两个属性的值
alert(house.rooms); // 7
alert(house.sharedEntrance); // false

// 调用对象的lock方法
house.lock();

// 更新'rooms'属性的值
house.rooms = 8;

// 动态添加一个全新属性
house.floors = 2;

// 再次读取'rooms'属性的值，注意值发生了改变
alert(house.rooms); // 8
```

假设我们现在想创建一个对象来表示另外一种房产，比如公寓。公寓和独立住房很像，不过一般房间更少且分布在同一楼层上，而且公寓楼一般只有一个面向街道的公共入口。为了表示这样一个对象，我们用对象直接量来定义一个新的变量：

```
var apartment = {
 floors: 1,
 rooms: 4,
 sharedEntrance: true,
 lock: function() {},
 unlock: function() {}
};
```

公寓和独立住房在概念上很相似但属性值不同。如果继续用上面这种方法定义更多类型的住宅，很快就会陷入麻烦，因为如果同一个属性被所有这些对象所共享，就不能轻易改变这个属性的名字，同样在这些对象中新增一个属性或方法也很麻烦。理想情况下，我们希望创建一个模板来表示这些对象所共有的属性和方法，这样如果我们想改变一个属性的名字或者增加一个新方法，做起来就很容易了。JavaScript允许利用构造器来创建这样一个对象模板，而在传统语言中这种构造器或者模板被称为类。

1.1.2　类

类是对象的模板，用于创建共享一系列属性和方法的类似对象。在Java和Objective-C这类编

程语言中，有专门定义类的特定关键字和语法结构。而在JavaScript中，只需定义一个普通函数，我们就能获得一些与其他语言中的类相同的功能。JavaScript中所有函数的定义方式都是相同的，不同之处在于对象是如何通过这些函数被创建出来的。

注意 JavaScript语言中class一词一直都是一个保留词，也就是说你自己定义的变量不能使用这个名称。不过这个词从未派上任何用场，只是被保留起来以备后用。现在看来这个词也许会在即将发布的新版JavaScript，即ECMAScript6中发挥一些作用，这个新版本正在起草中。

现在来创建一个构造函数用作房子和公寓对象的模板。稍后再来添加属性和方法。

```
function Accommodation() {};
```

这看起来与我们在JavaScript中所能创建的其他任何函数并无二致。要想把这个函数用作模板来创建对象，需要使用new这个关键字，后面跟着对该函数的调用。

```
var house = new Accommodation();
var apartment = new Accommodation();
```

用关键字new创建的所有对象都被称为这个函数所示的结构的对象实例，创建对象的过程就是这个模板实例化的过程。同一模板的对象实例之间互无关联，这些对象实例是完全独立的变量，只不过共享同一个模板结构而已。虽然这种模板和传统编程语言中的类很相似，但二者并不完全相同。本章稍后将对构造函数进行更详细的讨论。

1. 找出对象的构造器

通过上面的方法用模板创建的对象还有一个额外的属性，叫作constructor，这个属性指向创建该对象时所使用的JavaScript构造函数。知道了这一点，我们就可以直接将对象的constructor属性和某个构造函数进行比较，以查看程序中的对象是否是由某个构造函数生成的。

```
house.constructor === Accommodation;     // true
apartment.constructor === Accommodation; // true
```

这种比较也可以使用关键字instanceof来完成，该关键字的作用就是检查对象是否是某个构造函数的实例。

```
house instanceof Accommodation;     // true
apartment instanceof Accommodation; // true
```

实际上，因为一个实例的constructor属性直接指向创建该实例的构造函数，我们理论上可以直接用这个属性加上关键字new来创建新的实例。这种用法不常见，但是可以了解一下。

```
var apartment = new house.constructor();
apartment instanceof Accommodation; // true
```

前面在定义“类”的时候使用的是空函数，里面没有构成对象模板所需的属性和方法。给一个“类”添加属性和方法有两种方式，通过其原型或者通过其作用域。我们下面将逐一介绍。

2. 通过原型添加属性和方法

JavaScript中的每个函数，即每个构造器，都有一个叫prototype的属性。这个属性指向一个对象。我们用关键字new来创建一个“类”的对象实例时，实例中所包含的属性和方法都来自prototype所指向的这个对象。我们可以将点标记法用于这个prototype对象，从而为一个模板的所有对象实例添加自定义的属性和方法。对于添加的每个属性，都要赋予一个默认值以避免有未定义值。代码清单1-2演示了如何利用关键字prototype来定义模板或者“类”的属性和方法。

代码清单1-2　使用prototype关键字和点标记法为构造器添加属性和方法

```
// 定义一个名为Accommodation的构造函数
function Accommodation() {}

// 为这个“类”添加属性
Accommodation.prototype.floors = 0;
Accommodation.prototype.rooms = 0;
Accommodation.prototype.sharedEntrance = false;

// 为这个“类”添加方法
Accommodation.prototype.lock = function() {};
Accommodation.prototype.unlock = function() {};

// 创建Accommodation“类”的对象实例
var house = new Accommodation();
var apartment = new Accommodation();

// 读取对象实例的属性值
alert(house.floors); // 0
alert(house.sharedEntrance); // false

// 将对象实例的属性改为正确值
house.floors = 2;
accommodation.sharedEntrance = true;

// 调用对象实例的方法
house.unlock();
apartment.lock();
```

prototype属性本身是一个对象，这个对象被关联在扮演“类”这个角色的函数身上。因此除了点标记法，我们还可以使用对象直接量标记法。代码清单1-3演示了如何操作。

代码清单1-3　通过对象直接量为构造函数添加属性和方法

```
// 定义一个名为Accommodation的构造函数
function Accommodation() {}

// 通过对象直接量为这个“类”添加属性和方法
Accommodation.prototype = {
 floors: 0,
 rooms: 0,
 sharedEntrance: false,
 lock: function() {},
```

```
 unlock: function() {}
};

// 创建Accommodation“类”的对象实例
var house = new Accommodation();
var apartment = new Accommodation();

// 读取对象实例的属性值
alert(house.floors); // 0
alert(house.sharedEntrance); // false

// 将对象实例的属性设为正确值
house.floors = 2;
accommodation.sharedEntrance = true;

// 调用对象实例的方法
house.unlock();
apartment.lock();
```

prototype这个关键字有一个强大特性是允许在对象实例已经被创建之后继续添加属性和方法，而这些新添属性和方法会自动添加到所有对象实例中，不管是已创建的还是将要创建的，如代码清单1-4所示。

代码清单1-4 为已经存在的对象实例动态添加属性和方法

```
// 定义一个名为Accommodation的构造函数
function Accommodation() {};

// 通过对象直接量为这个“类”添加属性和方法
Accommodation.prototype = {
 floors: 0,
 rooms: 0,
 sharedEntrance: false,
 lock: function() {},
 unlock: function() {}
};

// 创建一个对象实例
var house = new Accommodation();

// 为“类”的原型动态添加一个新的方法
Accommodation.prototype.alarm = function() {};

// 已有的对象实例自动获得新的方法
house.alarm();
```

3. 通过作用域添加属性和方法

函数体内定义的任何变量或函数，其作用域都限于该函数体内，就是说在该函数体以外无法访问这些变量或函数——对这些变量和函数来说，包裹它们的外层函数提供了一个沙箱般的编程环境，或者说一个闭包。这对开发者来说好极了，因为在一个函数内定义的变量不会对另一个函数中的变量造成任何影响；同一个变量名甚至可以用在不同的函数中而不会产生冲突。

如果将一个变量或函数定义在任何一个函数之外，直接将其放在一个JavaScript或者HTML文件里，那么这个变量或函数会被添加到全局作用域中，这意味着在代码的任何位置都能够使用该变量或函数，甚至在其他函数体内也可以。实际上，根据作用域原则，在所有嵌套函数中都可以访问定义在其父函数中的变量。代码清单1-5演示了这一原则。

代码清单1-5　变量作用域

```
// 定义在任何函数之外的变量在全局作用域内，可以在任何位置访问
var myLibrary = {
 myName: "Dennis"
};

function doSomething() {
 // 函数体内定义的变量无法在函数体外访问到
 var innerVariable = 123;

 // 全局变量可以在函数体内访问到
 myLibrary.myName = "Hello";

 function doSomethingElse() {
  // 外层作用域内定义的变量能够在内层访问到
  innerVariable = 1234;
 }

 doSomethingElse();

 alert(innerVariable); // 1234
}

doSomething();

// 该属性在doSomething函数中已被覆盖
alert(myLibrary.myName); // "Hello"

// 在一个函数之外访问其内部定义的变量将导致错误
alert(innerVariable); // ERROR!
```

4. 上下文和this关键字

JavaScript中this关键字代表的是一个函数的上下文环境，这个上下文环境在大多数情况下指的是函数运行时封装这个函数的那个对象。当不通过任何对象单独调用一个函数时，上下文环境指的就是全局的window对象。在一个对象的方法中使用this指向的是这个对象本身，比如下面例子中的house对象。使用this关键字而不是对象的变量名，这么做的好处在于你可以随意改变对象的变量名而不用担心对象中方法的行为受到影响。this关键字成为了其指向的对象的代名词，所以和对象一样，你也可以对这个关键字本身使用点标记法。代码清单1-6演示了上下文和this关键字的用法。

代码清单1-6　使用this关键字和点标记法

```
// 在所有函数之外，this表示的是全局的window对象
```