

# 《Effective JavaScript》

## 图书基本信息

书名：《Effective JavaScript：编写高质量JavaScript代码的68个有效方法》

13位ISBN编号：9787111446231

10位ISBN编号：7111446232

出版时间：2014-1-1

出版社：机械工业出版社

作者：赫尔曼 (David Herman)

页数：164

译者：黄博文,喻杨

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：[www.tushu111.com](http://www.tushu111.com)

# 《Effective JavaScript》

## 内容概要

Effective 系列丛书经典著作，亚马逊五星级畅销书，Ecma 的JavaScript 标准化委员会著名专家撰写，JavaScript 语言之父、Mozilla CTO —— Brendan Eich 作序鼎力推荐！作者凭借多年标准化委员会工作和实践经验，深刻辨析JavaScript 的内部运作机制、特性、陷阱和编程最佳实践，将它们高度浓缩为极具实践指导意义的 68 条精华建议。

本书共分为 7 章，分别涵盖 JavaScript 的不同主题。第 1 章主要讲述最基本的主题，如版本、类型转换要点、运算符注意事项和分号局限等。第 2 章主要讲解变量作用域，介绍此方面的一些基本概念，以及一些最佳实践经验。第 3 章主要讲解函数的使用，深刻解析函数、方法和类，并教会读者在不同的环境下高效使用函数。第 4 章主要讲解原型和对象，分析 JavaScript 的继承机制以及原型和对象使用的最佳实践和原则。第 5 章主要介绍数组和字典，阐述将对象作为集合的用法以及使用数组和字典的一些陷阱。第 6 章介绍库和 API，讲解如何设计良好的 API 的技巧，以清楚、简洁和明确地表达程序，并提高可重用率。第 7 章讲解并发，在技术上讨论一些“约定成俗”的 JavaScript 用法。

# 《Effective JavaScript》

## 作者简介

David Herman，资深 JavaScript 技术专家，Ecma TC39 委员会成员，负责 JavaScript 的标准化工作。他拥有格林内尔学院的计算机科学学士学位和美国东北大学的计算机科学硕士及博士学位，现在 Mozilla 研究院担任高级研究员。

## 书籍目录

本书赞誉

译者序

序

前言

第1章 让自己习惯JavaScript 1

第1条：了解你使用的JavaScript版本 1

第2条：理解JavaScript的浮点数 6

第3条：当心隐式的强制转换 8

第4条：原始类型优于封装对象 13

第5条：避免对混合类型使用`==`运算符 14

第6条：了解分号插入的局限 16

第7条：视字符串为16位的代码单元序列 21

第2章 变量作用域 25

第8条：尽量少用全局对象 25

第9条：始终声明局部变量 27

第10条：避免使用`with` 28

第11条：熟练掌握闭包 31

第12条：理解变量声明提升 34

第13条：使用立即调用的函数表达式创建局部作用域 36

第14条：当心命名函数表达式笨拙的作用域 38

第15条：当心局部块函数声明笨拙的作用域 41

第16条：避免使用`eval`创建局部变量 43

第17条：间接调用`eval`函数优于直接调用 44

第3章 使用函数 46

第18条：理解函数调用、方法调用及构造函数调用之间的不同 46

第19条：熟练掌握高阶函数 48

第20条：使用`call`方法自定义接收者来调用方法 51

第21条：使用`apply`方法通过不同数量的参数调用函数 53

第22条：使用`arguments`创建可变参数的函数 54

第23条：永远不要修改`arguments`对象 56

第24条：使用变量保存`arguments`的引用 58

第25条：使用`bind`方法提取具有确定接收者的方法 59

第26条：使用`bind`方法实现函数柯里化 61

第27条：使用闭包而不是字符串来封装代码 62

第28条：不要信赖函数对象的`toString`方法 63

第29条：避免使用非标准的栈检查属性 65

第4章 对象和原型 67

第30条：理解`prototype`、`getPrototypeOf`和`__proto__`之间的不同 67

第31条：使用`Object.getPrototypeOf`函数而不要使用`__proto__`属性 69

第32条：始终不要修改`__proto__`属性 70

第33条：使构造函数与`new`操作符无关 71

第34条：在原型中存储方法 73

第35条：使用闭包存储私有数据 75

第36条：只将实例状态存储在实例对象中 76

第37条：认识到`this`变量的隐式绑定问题 78

第38条：在子类的构造函数中调用父类的构造函数 81

第39条：不要重用父类的属性名 84

- 第40条：避免继承标准类 86
- 第41条：将原型视为实现细节 88
- 第42条：避免使用轻率的猴子补丁 88
- 第5章 数组和字典 91
- 第43条：使用Object的直接实例构造轻量级的字典 91
- 第44条：使用null原型以防止原型污染 94
- 第45条：使用hasOwnProperty方法以避免原型污染 95
- 第46条：使用数组而不要使用字典来存储有序集合 99
- 第47条：绝不要在Object.prototype中增加可枚举的属性 102
- 第48条：避免在枚举期间修改对象 103
- 第49条：数组迭代要优先使用for循环而不是for...in循环 108
- 第50条：迭代方法优于循环 109
- 第51条：在类数组对象上复用通用的数组方法 113
- 第52条：数组字面量优于数组构造函数 114
- 第6章 库和API设计 116
- 第53条：保持一致的约定 116
- 第54条：将undefined看做“没有值” 117
- 第55条：接收关键字参数的选项对象 121
- 第56条：避免不必要的状态 125
- 第57条：使用结构类型设计灵活的接口 127
- 第58条：区分数组对象和类数组对象 130
- 第59条：避免过度的强制转换 134
- 第60条：支持方法链 137
- 第7章 并发 140
- 第61条：不要阻塞I/O事件队列 140
- 第62条：在异步序列中使用嵌套或命名的回调函数 143
- 第63条：当心丢弃错误 147
- 第64条：对异步循环使用递归 150
- 第65条：不要在计算时阻塞事件队列 153
- 第66条：使用计数器来执行并行操作 156
- 第67条：绝不要同步地调用异步的回调函数 160
- 第68条：使用promise模式清洁异步逻辑 162

## 精彩短评

- 1、前面几张巩固JS基础，后面一些比较基础
- 2、发现书中提到的很多点早已在编程中摸索着用上了，而且用了ES6之后很多以前要做的“优化”/绕的弯路也都不必要了，不过书还是不错的。（所以ES6之后这本书基本可以做为快速翻阅的厕所读物了
- 3、一本好书，没有兼容性，只有更先进，拥抱ES5，拥抱js异步
- 4、翻译太差了！
- 5、感觉一般
- 6、真知灼见，值得经常翻阅。
- 7、仅仅读了前半部分，由于工作原因暂时搁浅了，得找时间继续读完
- 8、总结的很不错、合适进阶。
- 9、呼~ 第四本！看到最后一章并发的部分就很吃力了，显然这是一本进阶的js书籍，还是先把那本权威指南啃完吧！
- 10、感觉译者“翻译”的水平有限，很多语句有点“直译”的感觉，导致一句中文会很长很长，给人语句不通顺的感觉，有些语句读几次不知道在讲什么，懂得这个知识点的人可能能猜测出原文在说什么，不懂这个知识点的人可能就不行了。推荐读者下载英文原书的PDF对照着看。这里也有中文版试读[http://wenku.it168.com/d\\_001329728.shtml](http://wenku.it168.com/d_001329728.shtml)
- 11、和其他Effective系列的书一样，书中的建议都很接地气，实用性很强，对于编写更好的JavaScript很有帮助。值得多看几遍，然后运用到实践中~
- 12、书中给出了一些易错点、应该避免的点。在js中，最难的莫过于变量作用域、函数、对象以及数组，书中都一一列出。在看这本书之前，很多点其实我知道要这么写，但总是不知道为什么，书中都有将原因一一列出，这也是我强烈推荐的原因。
- 13、不提糟糕的译文质量，单说原书，评价真有那么高吗？除了最后一章讲并发，其它所有内容在《JavaScript高级程序设计》中均有涉及，后者的讲解更加简明、清晰、有力。
- 14、我叫看不懂 还得读。。。javascript工具手册集
- 15、感觉会略有有点的落后于时代，先在都es6了。不过极大的提醒我的是，我还在用很多es3的语法，伤心
- 16、还好。比较实用的一些技巧。
- 17、javascript: the bad parts
- 18、很好的一本书，讲的清楚详细，看过能对js本身有更深入的理解，也有助于规范开发
- 19、翻译上比较生硬，大部分还算过得去，有几条烂的不忍卒读。内容也一般，68条里大部分都是基础知识而非best practice，部分反面教材极蠢，比effective c++差得太远，几乎不是一类书
- 20、这本书看的我好难受呀，翻了一半实在看不下去了
- 21、这书写的比较浅啦
- 22、锦上添花
- 23、很棒，适合有一点基础的人看
- 24、查缺补漏审视自己还是有不少JS问题没注意到
- 25、作者是ECMAScript标准制定小组的成员，所以写的非常细致。很值得前端开发人员一读。
- 26、非常好的一本JavaScript内功心法。
- 27、JavaScript进阶篇经典。虽然翻译难免有点瑕疵，但瑕不掩瑜，值得一看！
- 28、不错的书，值得反复读几遍，相比很多静态语言，掌握JavaScript更需要避免错误的实践，使用好正确的特性
- 29、差强人意
- 30、赞啊，理解透彻，讲解清楚。

## 精彩书评

1、放在书架上很久的一本书，周末抽空给看了。如果对js基础掌握不扎实的同学可以读一下，本书的一个好处就是作者本身是标准化委员会的，所以知识点都比较正确，问题是该书大多还是描述了表象，或者比较浅，所以很难把知识体系化。可以把该书作为索引去整体把握一下基础。对于javascript语言讲解的书，个人还是推荐读一下 @aimingoo 大大的语言精髓一书。

## 章节试读

### 1、《Effective JavaScript》的笔记-第1页

2014年看完的第20本书

### 2、《Effective JavaScript》的笔记-第37页

除了书上所说，还有另一种闭包方式如下，感觉更优雅

### 3、《Effective JavaScript》的笔记-第56页

之前只知道arguments是个伪数组，这章节说明了arguments还有隐藏的坑。。。说白了就是在非严格模式下，函数参数始终对应着arguments的索引值！

以章节中的callMethod函数为例，即callMethod(obj, method)中obj对应着arguments[0]，method对应着arguments[1]，这个是定死的，无论arguments变成什么样子，始终obj对应着arguments[0]，method对应着arguments[1]，所以章节例子中当shift了arguments前两个元素后，以为obj[method]是在引用obj.add方法，但实际上却是引用17[25]。。。

而在严格模式下，函数参数始却并不是终对应着arguments的索引值的，以strict(x)函数为例，即修改了arguments[0]却和x没有任何关系，但非严格模式下却是始终对应一起的。。。

截个ff下代码图，为了不报错，注释掉错误的obj[method].apply(obj, arguments)这句：注意标红和标紫代码对应的输出就明白了

### 4、《Effective JavaScript》的笔记-第1页

P9 isNaN 不可靠，用ES6 Number.isNaN

P21 空循环体的while循环需要显式的分号

```
() => {while(true)} //Uncaught SyntaxError:
```

```
() => {while(true);}
```

P29 正常的作用域中，会有与局部作用域中的变量同样多的作用域绑定储存在与之对应的环境层级中。但对于with作用域，绑定集合依赖于碰巧在给定时间点的对象。

```
var f = (x, y) => {  
  with(Math) {  
    return min(round(x), sqrt(y))  
  }  
}
```

```
f(5.5,4)
```

```
//2
```

```
Math.x = 0;
```

```
Math.y = 0;
```

```
f(5.5,4)
```

```
//0
```

P35 JavaScript没有块级作用域的一个例外，try...catch捕获的异常绑定到一个变量，该变量的作用域只



是catch语句。

```
(() => {
  var x = 'var', result = [];
  result.push(x);
  try{
    throw 'exception'
  }catch(x){
    x = 'catch'
  }
  result.push(x);
  return result;
})();
```

//["var", "var"]

P43 将eval封装到嵌套函数中防止作用域污染

```
var y = 'global';
var test = src => {
  (() => {eval(src)})(y);
  return y;
}
```

test("var y = 'local'")

test("var z = 'local'")

P44 间接调用eval优于直接调用

```
var x = 'global';
(() => {
  var x = 'local';
  return eval('x') //direct eval
})();
```

// local

```
var x = 'global';
```

```
(() => {
  var x = 'local';
  var f = eval;
  return f("x") //indirect eval
})();
```

//global

直接调用eval函数性能损耗高昂，间接调用eval失去对局部作用域的访问能力

强制间接调用eval(0,eval)('alert(0)')

P74

现代JavaScript引擎优化了原型查找

将方法储存在实例对象中将创建该函数的多个副本，比原型方法占用更多的内存

P87 继承标准类会由于内部属性（如[[Class]]）而被破坏

P96 使用hasOwnProperty方法避免原型污染

使用call避免hasOwnProperty被覆盖

```
function Dict() {}
```

```
dict = new Dict;
```

```
'toString' in dict;//true
```

```
Object.hasOwnProperty.call(dict, 'toString');//false
```

P103 如果确实需要在Object.prototype添加属性，使用Object.defineProperty enumerable 设false,不可枚举

P104 for...in 循环枚举一个对象属性时确保不要修改该对象

P114 数组行为：

将length属性值设为小于n的值会自动删除索引值大于或等于n的所有属性。

增加一个索引值为n（大于或等于length属性值）的属性会自动地设置length属性为n+1

p152 事件循环单次轮次中执行递归不会导致栈溢出

156 使用计数器执行并行操作

5、《Effective JavaScript》的笔记-第35页

中文版35页代码勘误，中文版的代码完全贴错了。。。审核也不认真，现贴上英文原版的代码

## 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:[www.tushu111.com](http://www.tushu111.com)