《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

《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。

《Effective JavaScript》

书籍目录

《Effective JavaScript》

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
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，最难的不过于变量作用域、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。

《Effective JavaScript》

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。

《Effective JavaScript》

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) =&gt; {
    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。

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

《Effective JavaScript》

版权说明