

首页 (<http://www.open-open.com/>) 代码 (<http://www.open-open.com/code/>) 文档 (<http://www.open-open.com/doc/>) 问答

全部经验分类

Android (/lib/tag/Android) IOS (/lib/tag/IOS) JavaScript (/lib/tag/JavaScript)

(/lib/list/all) 

所有分类 (/lib/list/all) > 开发语言与工具 (/lib/list/36) > JavaScript开发 (/lib/list/145)

## 理解 Javascript 的闭包

闭包 (/lib/tag/闭包) JavaScript (/lib/tag/JavaScript) 2016-02-28 19:12:51 发布

您的评价: 0.0

收藏

0收藏

来自: <http://zilongshanren.com/blog/2016-02-28-understand-javascript-closure.html>  
(<http://zilongshanren.com/blog/2016-02-28-understand-javascript-closure.html>)

因为最近几个月一直在做 Cocos Creator 这个项目,大部分时间都在与 Javascript 打交道,所以接下来我有必要写几篇文章介绍一下 JS 里面几个比较让人迷惑的地方:闭包,变量作用域,变量提升和 this 绑定。

今天这篇文章我们来聊一聊闭包。

## 什么是闭包?

闭包是一个函数,它在函数内部创建,并且携带了自身创建时的所处环境信息(比如变量信息和其它函数信息)。

上面这段话是引用至 MDN,它很清楚地说明了什么是闭包。

闭包 = 函数内部创建的函数(或者简称内部函数) + 该函数创建时所处环境信息

所以闭包并不等于匿名函数,虽然也有人称这些在函数内部创建的函数为闭包函数,但是我觉得其实并不准确。

我们看一下下面这段代码:

```
function init() {  
    var name = "Zilongshanren"; // name 是在 init 函数里面创建的变量  
    // displayName() 是一个内部函数,即一个闭包。注意,它不是匿名的。  
    function displayName() {  
        console.log(name);  
    }  
    //当 displayName 函数返回后,这个函数还能访问 init 函数里面定义的变量。  
    return displayName;  
}  
var closure = init();  
closure();
```

Zilongshanren  
undefined

displayName 是一个在 init 函数内部创建的函数,它携带了 init 函数内部作用域的所有信息,比如这里的 name 变量。当 displayName 函数返回的时候,它本身携带了当时创建时的环境信息,即 init 函数里面的 name 变量。

## 闭包有什么作用?

在理解什么是闭包之后,接下来你可能会问:这东西这么难理解,它到底有什么用啊?

因为在 Js 里面是没有办法创建私有方法的,它不像 java 或者 C++有什么 private 关键字可以定义私有的属性和方法。Js 里面只有函数可以创建出属于自身的作用域的对象,Js 并没有块作用域!这个我后面会再写一篇文章详细介绍。

编程老鸟都知道,程序写得好,封装和抽象要运用得好!不能定义私有的属性和方法,意味着封装和抽象根本没法用。。。

不能定义私有的东西,所有变量和函数都 public 显然有问题, Global is Evil!

闭包是我们的救星!

我们看一下下面这段代码:

```
var makeCounter = function() {
  var privateCounter = 0;
  function changeBy(val) {
    privateCounter += val;
  }
  return {
    increment: function() {
      changeBy(1);
    },
    decrement: function() {
      changeBy(-1);
    },
    value: function() {
      return privateCounter;
    }
  }
};

var counter1 = makeCounter();
var counter2 = makeCounter();
console.log(counter1.value()); /* Alerts 0 */
counter1.increment();
counter1.increment();
console.log(counter1.value()); /* Alerts 2 */
counter1.decrement();
console.log(counter1.value()); /* Alerts 1 */
console.log(counter2.value()); /* Alerts 0 */
```

```
0
2
1
0
undefined
```

这里的 `privateCounter` 变量和 `changeBy` 都是私有的，对于 `makeCounter` 函数外部是完全不可见的。这样我们通过 `makeCounter` 生成的对象就把自己的私有数据和私有方法全部隐藏起来了。

这里有没有让你想到点什么？

哈哈，这不就是 **OO** 么？封装数据和操作数据的方法，然后通过公共的接口调用来完成数据处理。

当然，你也许会说，我用原型继承也可以实现 **OO** 呀。没错，现在大部分人也正是这么干的，包括我们自己。不过继承这个东西，在理解起来总是非常困难的，因为要理解一段代码，你必须理解它的所有继承链。如果一旦代码出 **bug** 了，这将是非常难调试的。

扯远了，接下来，让我们看看如何正确地使用闭包。

## 如何正确地使用闭包？

闭包会占用内存，也会影响 **js** 引擎的执行效率，所以，如果一段代码被频繁执行，那么要谨慎考虑在这段代码里面使用闭包。

让我们来看一个创建对象的函数：

```
function MyObject(name, message) {
  this.name = name.toString();
  this.message = message.toString();
  this.getName = function() {
    return this.name;
  };

  this.getMessage = function() {
    return this.message;
  };
}

var myobj = new MyObject();
```

`var myobj = new MyObject();` 每一次被调用生成一个新对象的时候，都会生成两个闭包。如果你的程序里面有成千上万个这样的 `MyObject` 对象，那么会额外多出很多内存占用。

正确的做法应该是使用原型链：

```
function MyObject(name, message) {
  this.name = name.toString();
  this.message = message.toString();
}
MyObject.prototype.getName = function() {
  return this.name;
};
MyObject.prototype.getMessage = function() {
  return this.message;
};

var myobj = new MyObject();
```

现在 `MyObject` 原型上面定义了两个方法，当我们通过 `new` 去创建对象的时候，这两个方法只会在原型上面存有一份。

## 闭包的性能如何？

闭包也是一个函数，但是它存储了额外的环境信息，所以理论上它比纯函数占用更多的内存，而且 `Js` 引擎在解释执行闭包的时候消耗也更大。不过它们之间的性能差别在 `3%`和 `5%`之间（这是 `Google` 上得到的数据，可能不是太准确）。

但是，闭包的好处肯定是大大的。多使用闭包和无状态编程，让 `Bug` 从此远离我们。

## 小结

面向对象是穷人的闭包(`OO is an poor man's closure.`)

理解了闭包，你就能理解大部分 `FP` 范式的 `Js` 类库及其隐藏在背后的设计思想。当然仅有闭包还不够，你还需要被 `FP` 和无状态，`lambda calculus` 等概念洗脑。

## Reference

- <https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures> (<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Closures>)

## 同类热门经验

---

1. `Node.js` 初体验 (/lib/view/open1326870121968.html)
2. `JavaScript`开发规范要求 (/lib/view/open1352263831610.html)
3. 使用拖拉操作来自定义网页界面布局并保存结果 (/lib/view/open1325064347889.html)
4. `Nodejs`入门学习, `nodejs web`开发入门, `npm`、`express`、`socket`配置安装、`nodejs`聊天室开发 (/lib/view/open1329050007640.html)
5. 利用`HTML5`同时上传多个文件 - `resumable.js` (/lib/view/open1327591300671.html)
6. `nide`：一个不错的`Node.js`开发工具`IDE` (/lib/view/open1325834128750.html)

## 阅读目录

---

什么是闭包？

闭包有什么作用？

如何正确地使用闭包？

闭包的性能如何？

小结

## Reference

- |  |  |   |
|--|--|---|
| 相关文档 — 更多 ( <a href="http://www.open-open.com/doc">http://www.open-open.com/doc</a> )  | 相关经验 — 更多 ( <a href="http://www.open-open.com/lib">http://www.open-open.com/lib</a> )  | 相关讨论 — 更多 ( <a href="http://www.open-open.com/solution">http://www.open-open.com/solution</a> )   |
| • Javascript 闭包.pdf ( <a href="http://www.open-open.com/doc/view/fd2e6c26ba63455ebfc96ce6410ef59b">http://www.open-open.com/doc/view/fd2e6c26ba63455ebfc96ce6410ef59b</a> )  | • 理解Javascript的闭包 ( <a href="http://www.open-open.com/lib/view/open1331093205921.html">http://www.open-open.com/lib/view/open1331093205921.html</a> )                    | • 什么是闭包(Closure)? ( <a href="http://www.open-open.com/solution/view/1326292703265">http://www.open-open.com/solution/view/1326292703265</a> )     |
| • 《JavaScript高级程序设计(第2版)》(Professional JavaScript for Web Developers, 2nd Edition).pdf ( <a href="http://www.open-open.com/doc/view/2dbab4a3ee2a47f9a3c142c32dd655f5">http://www.open-open.com/doc/view/2dbab4a3ee2a47f9a3c142c32dd655f5</a> ) | • javascript闭包 js闭包理解 高级JS程序员员的必经之路。 ( <a href="http://www.open-open.com/lib/view/open1390879109398.html">http://www.open-open.com/lib/view/open1390879109398.html</a> ) | • Javascript内存泄露 ( <a href="http://www.open-open.com/solution/view/1336633857125">http://www.open-open.com/solution/view/1336633857125</a> )      |
| • JavaScript权威指南(第5版)中文版(下).pdf ( <a href="http://www.open-open.com/doc/view/a5eba19e3c07490588693b5d2101909d">http://www.open-open.com/doc/view/a5eba19e3c07490588693b5d2101909d</a> )  | • JavaScript的this和闭包 ( <a href="http://www.open-open.com/lib/view/open1348033124302.html">http://www.open-open.com/lib/view/open1348033124302.html</a> )                 | • Javascript闭包学习 ( <a href="http://www.open-open.com/solution/view/1321491048452">http://www.open-open.com/solution/view/1321491048452</a> )      |
| • 你不知道的Javascript (英文).pdf ( <a href="http://www.open-open.com/doc/view/cf1c06fd69824b79bee25942511bbd4b">http://www.open-open.com/doc/view/cf1c06fd69824b79bee25942511bbd4b</a> )   | • JavaScript闭包, 你理解吗? ( <a href="http://www.open-open.com/lib/view/open1338779176812.html">http://www.open-open.com/lib/view/open1338779176812.html</a> )                | • 我为什么向后端工程师推荐Node.js ( <a href="http://www.open-open.com/solution/view/1322451238921">http://www.open-open.com/solution/view/1322451238921</a> ) |
| • JavaScript内核.pdf ( <a href="http://www.open-open.com/doc/view/72e1e2d9367645fbb047e9813b411de1">http://www.open-open.com/doc/view/72e1e2d9367645fbb047e9813b411de1</a> )   | • 深入理解Javascript闭包(closure) ( <a href="http://www.open-open.com/lib/view/open1374588023699.html">http://www.open-open.com/lib/view/open1374588023699.html</a> )          | • 北京猎头推荐: 高级前端 js主管职位 ( <a href="http://www.open-open.com/solution/view/1374309260871">http://www.open-open.com/solution/view/1374309260871</a> ) |
| • JavaScript闭包 .pdf ( <a href="http://www.open-open.com/doc/view/a03421da67e94d99bff289abbbc90024">http://www.open-open.com/doc/view/a03421da67e94d99bff289abbbc90024</a> )  | • JavaScript变量作用域和闭包的基础知识 ( <a href="http://www.open-open.com/lib/view/open1430190224959.html">http://www.open-open.com/lib/view/open1430190224959.html</a> )            | • PHP程序员的技术成长规划 ( <a href="http://www.open-open.com/solution/view/1414478644325">http://www.open-open.com/solution/view/1414478644325</a> )       |
| • JavaScript 匿名函数.ppt ( <a href="http://www.open-open.com/doc/view/f5807a2d4f66449ca0ac1daade00a6ee">http://www.open-open.com/doc/view/f5807a2d4f66449ca0ac1daade00a6ee</a> )  | • Javascript闭包简单理解 ( <a href="http://www.open-open.com/lib/view/open1421284279031.html">http://www.open-open.com/lib/view/open1421284279031.html</a> )                   | • Node.js 究竟是什么? ( <a href="http://www.open-open.com/solution/view/1324519368764">http://www.open-open.com/solution/view/1324519368764</a> )      |
| • JavaScript 闭包 (closure).ppt ( <a href="http://www.open-open.com/doc/view/f03dbdaa4e1d4edaada97bbc30d912b6">http://www.open-open.com/doc/view/f03dbdaa4e1d4edaada97bbc30d912b6</a> )  | • 深入理解javascript原型和闭包 ( <a href="http://www.open-open.com/lib/view/open1457782893635.html">http://www.open-open.com/lib/view/open1457782893635.html</a> )                |   |
| • 真正的JavaScript忍者秘籍.pdf ( <a href="http://www.open-open.com/doc/view/d697952596ec4682a4c835aa351fcb8c">http://www.open-open.com/doc/view/d697952596ec4682a4c835aa351fcb8c</a> )  | • 学习Javascript闭包 (Closure) ( <a href="http://www.open-open.com/lib/view/open1454503305526.html">http://www.open-open.com/lib/view/open1454503305526.html</a> )           |   |
| • JavaScript高级程序设计 (JavaScript for Web Developers).pdf ( <a href="http://www.open-open.com/doc/view/2f14e310a9a148faa5f258c03ba87acd">http://www.open-open.com/doc/view/2f14e310a9a148faa5f258c03ba87acd</a> )                                 | • 让你分分钟学会 JavaScript闭包 ( <a href="http://www.open-open.com/lib/view/open1450679996542.html">http://www.open-open.com/lib/view/open1450679996542.html</a> )               |   |
| • 精通JavaScript开发(Professional JavaScript for Web Developers).pdf ( <a href="http://www.open-open.com/doc/view/ceaa9dc36c774dc39073652e0cc2c43e">http://www.open-open.com/doc/view/ceaa9dc36c774dc39073652e0cc2c43e</a> )                       | • javascript进阶系列专题: 闭包 (Closure) ( <a href="http://www.open-open.com/lib/view/open1452049468073.html">http://www.open-open.com/lib/view/open1452049468073.html</a> )     |   |
| • 超实用的JavaScript代码段.pdf ( <a href="http://www.open-open.com/doc/view/22c46cdaa9914f70bc42af8eff901e0c">http://www.open-open.com/doc/view/22c46cdaa9914f70bc42af8eff901e0c</a> )  | • Javascript闭包与作用域 ( <a href="http://www.open-open.com/lib/view/open1410396143601.html">http://www.open-open.com/lib/view/open1410396143601.html</a> )                   |   |
| • 超实用的JavaScript代码段.pdf ( <a href="http://www.open-open.com/doc/view/ec6204317dd64fc7b076f7320c990639">http://www.open-open.com/doc/view/ec6204317dd64fc7b076f7320c990639</a> )  | • 高效使用 JavaScript 闭包 ( <a href="http://www.open-open.com/lib/view/open1463445178466.html">http://www.open-open.com/lib/view/open1463445178466.html</a> )                 |   |
| • JavaScript 核心及实践.pdf ( <a href="http://www.open-open.com/doc/view/ffcaed6dd21e49d690c4c08a7d40afae">http://www.open-open.com/doc/view/ffcaed6dd21e49d690c4c08a7d40afae</a> )   | • JavaScript 闭包究竟是什么 ( <a href="http://www.open-open.com/lib/view/open1418133591839.html">http://www.open-open.com/lib/view/open1418133591839.html</a> )                 |   |
| • JavaScript 第六版.doc ( <a href="http://www.open-open.com/doc/view/118e0fbefb1403ca0f95f8f30cbd98e">http://www.open-open.com/doc/view/118e0fbefb1403ca0f95f8f30cbd98e</a> )   |  |   |
| • [闭包权威指南] (Closure:The Definitive Guide) .pdf ( <a href="http://www.open-open.com/doc/view/7f0752cb9cf5409089c1a294da3175d6">http://www.open-open.com/doc/view/7f0752cb9cf5409089c1a294da3175d6</a> )   |  |   |
| • jQuery 基础教程 (第四版) .pdf ( <a href="http://www.open-open.com/doc/view/f40b7c8fa5594851807f0b3d60d1b62d">http://www.open-open.com/doc/view/f40b7c8fa5594851807f0b3d60d1b62d</a> )   |  |   |
| • jQuery学习第四版.pdf ( <a href="http://www.open-open.com/doc/view/88d7640dc3ad44b8bbd78783d8e934cd">http://www.open-open.com/doc/view/88d7640dc3ad44b8bbd78783d8e934cd</a> )  |  |   |
| • jQuery基础教程 (第四版) .pdf ( <a href="http://www.open-open.com/doc/view/b6d9eb3ac22d493eb30f59eded4812f0">http://www.open-open.com/doc/view/b6d9eb3ac22d493eb30f59eded4812f0</a> )  |  |   |
| • jQuery基础教程 (第4版) .pdf ( <a href="http://www.open-open.com/doc/view/66b951a14779422a92937c3c865b894c">http://www.open-open.com/doc/view/66b951a14779422a92937c3c865b894c</a> )  |  |   |



web\_id=1257892335)