理解 Javascript 的闭包

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

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
```

这里面的 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，你必须要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

闭包的性能如何？

小结

Reference

web_id=1257892335)