**Javascript模块化编程（一）：模块的写法**

作者： [阮一峰](http://www.ruanyifeng.com)

日期： [2012年10月26日](http://www.ruanyifeng.com/blog/2012/10/)

随着网站逐渐变成"[互联网应用程序](http://en.wikipedia.org/wiki/Web_application)"，嵌入网页的Javascript代码越来越庞大，越来越复杂。



网页越来越像桌面程序，需要一个团队分工协作、进度管理、单元测试等等......开发者不得不使用软件工程的方法，管理网页的业务逻辑。

Javascript模块化编程，已经成为一个迫切的需求。理想情况下，开发者只需要实现核心的业务逻辑，其他都可以加载别人已经写好的模块。

但是，Javascript不是一种模块化编程语言，它不支持"[类](http://www.ruanyifeng.com/blog/2012/07/three_ways_to_define_a_javascript_class.html)"（class），更遑论"模块"（module）了。（正在制定中的[ECMAScript标准](http://en.wikipedia.org/wiki/ECMAScript)第六版，将正式支持"类"和"模块"，但还需要很长时间才能投入实用。）

Javascript社区做了很多努力，在现有的运行环境中，实现"模块"的效果。本文总结了当前＂Javascript模块化编程＂的最佳实践，说明如何投入实用。虽然这不是初级教程，但是只要稍稍了解Javascript的基本语法，就能看懂。



**一、原始写法**

模块就是实现特定功能的一组方法。

只要把不同的函数（以及记录状态的变量）简单地放在一起，就算是一个模块。

　　function m1(){  
　　　　//...  
　　}

　　function m2(){  
　　　　//...  
　　}

上面的函数m1()和m2()，组成一个模块。使用的时候，直接调用就行了。

这种做法的缺点很明显："污染"了全局变量，无法保证不与其他模块发生变量名冲突，而且模块成员之间看不出直接关系。

**二、对象写法**

为了解决上面的缺点，可以把模块写成一个对象，所有的模块成员都放到这个对象里面。

　　var module1 = new Object({

　　　　\_count : 0,

　　　　m1 : function (){  
　　　　　　//...  
　　　　},

　　　　m2 : function (){  
　　　　　　//...  
　　　　}

　　});

上面的函数m1()和m2(），都封装在module1对象里。使用的时候，就是调用这个对象的属性。

　　module1.m1();

但是，这样的写法会暴露所有模块成员，内部状态可以被外部改写。比如，外部代码可以直接改变内部计数器的值。

　　module1.\_count = 5;

**三、立即执行函数写法**

使用"[立即执行函数](http://benalman.com/news/2010/11/immediately-invoked-function-expression/)"（Immediately-Invoked Function Expression，IIFE），可以达到不暴露私有成员的目的。

　　var module1 = (function(){

　　　　var \_count = 0;

　　　　var m1 = function(){  
　　　　　　//...  
　　　　};

　　　　var m2 = function(){  
　　　　　　//...  
　　　　};

　　　　return {  
　　　　　　m1 : m1,  
　　　　　　m2 : m2  
　　　　};

　　})();

使用上面的写法，外部代码无法读取内部的\_count变量。

　　console.info(module1.\_count); //undefined

module1就是Javascript模块的基本写法。下面，再对这种写法进行加工。

**四、放大模式**

如果一个模块很大，必须分成几个部分，或者一个模块需要继承另一个模块，这时就有必要采用"放大模式"（augmentation）。

　　var module1 = (function (mod){

　　　　mod.m3 = function () {  
　　　　　　//...  
　　　　};

　　　　return mod;

　　})(module1);

上面的代码为module1模块添加了一个新方法m3()，然后返回新的module1模块。

**五、宽放大模式（Loose augmentation）**

在浏览器环境中，模块的各个部分通常都是从网上获取的，有时无法知道哪个部分会先加载。如果采用上一节的写法，第一个执行的部分有可能加载一个不存在空对象，这时就要采用"宽放大模式"。

　　var module1 = ( function (mod){

　　　　//...

　　　　return mod;

　　})(window.module1 || {});

与"放大模式"相比，＂宽放大模式＂就是"立即执行函数"的参数可以是空对象。

**六、输入全局变量**

独立性是模块的重要特点，模块内部最好不与程序的其他部分直接交互。

为了在模块内部调用全局变量，必须显式地将其他变量输入模块。

　　var module1 = (function ($, YAHOO) {

　　　　//...

　　})(jQuery, YAHOO);

上面的module1模块需要使用jQuery库和YUI库，就把这两个库（其实是两个模块）当作参数输入module1。这样做除了保证模块的独立性，还使得模块之间的依赖关系变得明显。这方面更多的讨论，参见Ben Cherry的著名文章[《JavaScript Module Pattern: In-Depth》](http://www.adequatelygood.com/2010/3/JavaScript-Module-Pattern-In-Depth)。

这个系列的第二部分，将讨论如何在浏览器环境组织不同的模块、管理模块之间的依赖性。