<http://www.jb51.net/article/65554.htm>

<https://blog.csdn.net/chy555chy/article/details/52703385>

# **NodeJs中的VM模块详解**

转载  2015-05-06   投稿：junjie   [IMG_256 我要评论](http://www.jb51.net/article/65554.htm" \l "comments)

这篇文章主要介绍了NodeJs中的VM模块详解,本文讲解了什么是VM？ 、VM模块的runInThisContext、runInThisContext方法等内容,需要的朋友可以参考下

**什么是VM？**

VM模块是NodeJS里面的核心模块，支撑了require方法和NodeJS的运行机制，我们有些时候可能也要用到VM模板来做一些特殊的事情。

通过VM，JS可以被编译后立即执行或者编译保存下来稍后执行（JavaScript code can be compiled and run immediately or compiled, saved, and run later.）  
VM模块包含了三个常用的方法，用于创建独立运行的沙箱体制，如下三个方法  
vm.runInThisContext(code, filename);

此方法用于创建一个独立的沙箱运行空间，code内的代码可以访问外部的global对象，但是不能访问其他变量

而且code内部global与外部共享

复制代码代码如下:

var vm = require("vm");  
   
var p = 5;  
global.p = 11;  
   
vm.runInThisContext("console.log('ok', p)");// 显示global下的11  
vm.runInThisContext("console.log(global)"); // 显示global  
   
console.log(p);// 显示5  
vm.runInContext(code, sandBox);

此方法用于创建一个独立的沙箱运行空间，sandBox将做为global的变量传入code内，但不存在global变量

sandBox要求是vm.createContext()方法创建的sandBox

复制代码代码如下:

var vm = require("vm");  
var util = require("util");  
   
var window = {  
    p: 2,  
    vm: vm,  
    console: console,  
    require: require  
};  
   
var p = 5;  
   
global.p = 11;  
   
vm.createContext(window);  
vm.runInContext('p = 3;console.log(typeof global);', window); // global是undefined  
   
console.log(window.p);// 被改变为3  
   
console.log(util.inspect(window));  
vm.runInNewContext(code, sandbox, opt);

这个方法应该和runInContext一样，但是少了创建sandBox的步骤

**比较**



**更为复杂的情形**  
如果runInContext里面执行runInThisContext会是怎么样，runInThisContext访问到的global对象是谁的？

**如下代码将会怎么执行？**

复制代码代码如下:

var vm = require("vm");  
var util = require("util");  
   
var window = {  
    p: 2,  
    vm: vm,  
    console: console,  
    require: require  
};  
   
window.global = window;  
   
var p = 5;  
   
global.p = 11;  
   
vm.runInNewContext('p = 3;console.log(typeof global);require(\'vm\').runInThisContext("console.log(p)");', window);

runInThisContext里面的代码可以访问外部的global对象,但外面实际上不存在global对象(虽然有，但本质不是global对象）,只要记住一点，runInThisContext只能访问最顶部的global对象就OK了

**执行结果如下**

复制代码代码如下:

object (global存在）  
11 （顶部global的p)

这个还是有点疑问global.p不是被改变了吗？应该是3才对呀

avaScript代码可以被编译然后立即运行，或者编译，保存，并且之后运行。

## **Class: vm.Script**

Added in:   
Instances of the vm.Script class contain precompiled scripts that can be executed in specific sandboxes (or “contexts”).

vm.Script类的实例包含的预编译脚本可以在沙箱中被执行（或者上下文环境中）。

### **new vm.Script(code, options)**

Added in: v0.3.1

* code <string> The JavaScript code to compile.
* options
  + filename <string> Specifies the filename used in stack traces produced by this script.
  + lineOffset <number> Specifies the line number offset that is displayed in stack traces produced by this script.
  + columnOffset <number> Specifies the column number offset that is displayed in stack traces produced by this script.
  + displayErrors <boolean> When true, if an Error error occurs while compiling the code, the line of code causing the error is attached to the stack trace.
  + timeout <number> Specifies the number of milliseconds to execute code before terminating execution. If execution is terminated, an Error will be thrown.
  + cachedData <Buffer> Provides an optional Buffer with V8’s code cache data for the supplied source. When supplied, the cachedDataRejected value will be set to either true or false depending on acceptance of the data by V8.
  + produceCachedData <boolean> When true and no cachedData is present, V8 will attempt to produce code cache data for code. Upon success, a Buffer with V8’s code cache data will be produced and stored in the cachedData property of the returned vm.Script instance. The cachedDataProduced value will be set to either true or false depending on whether code cache data is produced successfully.
* code <string> 待编译的JavaScript代码
* options 选项
  + filename <string> 指定这个脚本产生的堆栈跟踪信息所使用的文件名
  + lineOffset <number> 指定这个脚本产生的堆栈跟踪信息的行号偏移量
  + columnOffset <number> 指定这个脚本产生的堆栈跟踪信息的列号偏移量
  + displayErrors <boolean> 当被设置为真的时候，如果在编译代码时发生了错误，造成错误的行号会被附加到堆栈跟踪信息中。
  + timeout <number> 指定中断执行前代码执行的最长时间，如果执行被中断，就会抛出一个错误。
  + cachedData <Buffer> 提供一个可选的V8代码缓冲数据给提供的源使用。当提供该值的时候，cachedDataRejected值将会被设置，要么为真要么为假，取决于V8接受的数据。
  + produceCachedData <boolean> 当设置为真，且没有cachedData的时候，V8将会尝试产生代码来缓存原代码的数据。执行成功后，V8代码缓存数据的字节数组就会被生产并存储在返回的vm.Script实例的cachedData属性中。cachedDataProduced 要么设置为真要么设置为假，取决于是否代码还从数据被成功生成。

Creating a new vm.Script object compiles code but does not run it. The compiled vm.Script can be run later multiple times. It is important to note that the code is not bound to any global object; rather, it is bound before each run, just for that run.

创建一个新的vm.Script对象编译代码但是并不运行。编译的vm.Script之后可以被多次运行。必须注意的是该代码并没有绑定到任何的全局对象；相反的它们只在每次运行前被绑定到那次的运行环境上。

script.runInContext(context);可以提供上下文环境