Atitit 模块导入之模块化 head.js [requirejs](http://requirejs.org/" \t "https://www.cnblogs.com/2050/archive/2012/07/14/_blank)和[seajs](http://seajs.com/" \t "https://www.cnblogs.com/2050/archive/2012/07/14/_blank)

## [用headjs来管理和加载js](https://www.cnblogs.com/2050/archive/2012/07/14/2591449.html)

现在有许多成熟的js模块加载器，比如[requirejs](http://requirejs.org/" \t "https://www.cnblogs.com/2050/archive/2012/07/14/_blank)和[seajs](http://seajs.com/" \t "https://www.cnblogs.com/2050/archive/2012/07/14/_blank)等，但是对一些小的项目来说，它们的功能可能太过于“强大”，也许我们只是想要有一个能动态加载js的功能，或许我们只是想装B一下，让自己写出的页面没有一大堆的<script src=”a.js”></script>这样的东西。前面说的两种js加载器都是更强调模块性，也就是更强调js文件的组织和管理，更适用于大的项目。

就像刚刚说的，我只想有一个js文件加载器，我只需提供给它一个js文件的地址就行了的那种呢？那就进入我们的主题了，使用[headjs](http://headjs.com/" \t "https://www.cnblogs.com/2050/archive/2012/07/14/_blank)吧。headjs其实是一整套的工具，但我只想介绍的是它其中的Javascript Loader功能。以下是它的基本用法：

|  |  |
| --- | --- |
| 1 | head.js("/path/to/file.js"); |

最简单的用法，提供一个js的文件的地址给它作为参数，然后它就会默默的在背后进行无阻塞的加载了，至于什么时候加载完可用了，谁也不知道。

|  |  |
| --- | --- |
| 1  2  3 | head.js("/path/to/file.js", function() {      /\*js加载完成\*/  }); |

最基本的用法，除了提供一个地址参数外，还提供一个回调函数作为第二个参数。在js加载完成后，回调用回调函数，你可以把依赖该js的代码写在回调函数里。

|  |  |
| --- | --- |
| 1 | head.js("file1.js", "file2.js", ... "fileN.js"); |

提供多个js文件地址，这些js会并行的加载，但会按照参数给定的先后顺序执行这些文件，例如即使file2.js比file1.js先加载完成，但它还会一直等到在file1.js加载和执行完后再执行。

|  |  |
| --- | --- |
| 1  2  3 | head.js("file1.js", "file2.js", function() {    }); |