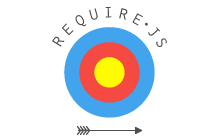
# RequireJS

当你要加载JavaScript模块时，就会使用script标签。为了加载依赖的模块，你就要先加载被依赖的，之后再加载依赖的。使用script标签时，你需要按照此特定顺序安排它们的加载，而且脚本的加载是同步的。可以使用async和defer关键字使得加载异步，但可能因此在加载过程中丢失加载的顺序。另一个选择是将所有的脚本捆绑打包在一起，但在捆绑的时候你仍然需要把它们按照正确的顺序排序。

任何疑问都可以咨询本人：148334733

AMD就是这样一种对模块的定义，使模块和它的依赖可以被异步的加载，但又按照正确的顺序。

require.js的诞生，就是为了解决这两个问题：



　　（1）实现js文件的异步加载，避免网页失去响应；

　　（2）管理模块之间的依赖性，便于代码的编写和维护。

RequireJS所做的是，在你使用script标签加载你所定义的依赖时，将这些依赖通过head.appendChild()函数来加载他们。当依赖加载以后，RequireJS计算出模块定义的顺序，并按正确的顺序进行调用。这意味着你需要做的仅仅是使用一个“根”来读取你需要的所有功能，然后剩下的事情只需要交给RequireJS就行了。为了正确的使用这些功能，你定义的所有模块都需要使用RequireJS的API，否者它不会像期望的那样工作。

RequireJS API 存在于RequireJS载入时创建的命名空间requirejs下。其主要API主要是下面三个函数:

define– 该函数用户创建模块。每个模块拥有一个唯一的模块ID，它被用于RequireJS的运行时函数，define函数是一个全局函数，不需要使用requirejs命名空间.  
require– 该函数用于读取依赖。同样它是一个全局函数，不需要使用requirejs命名空间.  
config– 该函数用于配置RequireJS.  
在后面，我们将教你如果使用这些函数，但首先让我们先了解下RequireJS的加载流程。

data-main属性  
当你下载RequireJS之后，你要做的第一件事情就是理解RequireJS是怎么开始工作的。当RequireJS被加载的时候，它会使用data-main属性去搜寻一个脚本文件（它应该是与使用src加载RequireJS是相同的脚本）。data-main需要给所有的脚本文件设置一个根路径。根据这个根路径，RequireJS将会去加载所有相关的模块。下面的脚本是一个使用data-main例子：

1 <script src="scripts/require.js" data-main="scripts/app.js"></script>

另外一种方式定义根路劲是使用配置函数，后面我们将会看到。requireJs假设所有的依赖都是脚本，那么当你声明一个脚本依赖的时候你不需要使用.js后缀。

配置函数  
如果你想改变RequireJS的默认配置来使用自己的配置，你可以使用require.configh函数。config函数需要传入一个可选参数对象，这个可选参数对象包括了许多的配置参数选项。下面是一些你可以使用的配置：

baseUrl——用于加载模块的根路径。  
paths——用于映射不存在根路径下面的模块路径。  
shims——配置在脚本/模块外面并没有使用RequireJS的函数依赖并且初始化函数。假设underscore并没有使用  RequireJS定义，但是你还是想通过RequireJS来使用它，那么你就需要在配置中把它定义为一个shim。  
deps——加载依赖关系数组