JavaScript模块化开发小谈

在网上已经有很多关于模块化开发的文章了，这里还是按照自己的理解来整理一下。

随着项目文件的越来越大和需求的越来越贴近现实（我发现现在客户不如：一个领导说我要审批你们报上来的资料，系统发布以后用的还不错，可能是有那个新鲜感，等领导烦了说：每个文件我都这么操作太累了，能不能一次报，到最后说能不能自动判断对不对……），以后的需求估计要让做个人工智能出来……借着上面继续，需求的内容越来越复杂，一个人开发时间长，多个人开发这样那样的问题一堆等等这时候Javascript模块化开发诞生了。

先说说发展历程

刚开始是这样写代码的：

functionfunc1(){}

functionfunc2(){}

后来经过变形是这样的：

varobj={

func1:function(){

},

func2:function(){

}

}

最后经过总价实践后是这样的：

(function(){})();

闭包：至于这种方式**防止全局变量的污染，有效的提高了开发的思路和开发的效率。**

这里重点说下我设想的思路:



上面的项目文件列表是这样的；index主页上面有不同的标签，每个标签下面有不同的功能（很复杂的）。Index的代码如下

<!DOCTYPEhtml>

<html>

<head>

<metahttp-equiv="Content-Type"content="text/html; charset=utf-8"/>

<title>**JavaScript模块化编程**</title>

<scriptsrc="../jquery.js"></script>

<scriptsrc="Js/main.js"></script>

<scriptsrc="Js/a.js"></script>

<scriptsrc="Js/b.js"></script>

<scriptsrc="Js/c.js"></script>

</head>

<body>

</body>

</html>

mian处理公用的业务逻辑，a，b，c不同的功能块

main.js的代码如下：

$(function(){

varsum=myModule.addMethod(1),

sub=myModule.subMethod(1,3),

mod=myModule.modMethod(1,3);

myModule.alert=function(){

alert($().jquery);

};

});

a.js的代码如下:

(function(my){

varx=8;

my.addMethod=function(){

returnarguments[0]+x;

}

})(window.myModule=window.myModule||{});

b和c的代码就不帖了和a的雷同；

为什么这么写呢？假如说写成下面的样子行吗？

(function(my){

my.modMethod=function(){

returnarguments[0]/arguments[1];

};

//比如执行完异步后执行

my.callBack=function(){

if(typeofarguments[0]=="function"){

arguments[0]();

}

}

})(window.myModule||{});

如果写成了以上的这种，在index页面上引入js的顺序变化有可能window.myModule就是undefined.

a,b,c都是一个页面上不用的业务逻辑处理，当然相互可以关联，导入。

这样就有效的阻止了**全局变量的污染, 可以找不同的开发者开发同一个模块。**

那这样暴漏出另一个问题就是index页面上引入的js过于过，而且会影响加载速度，在访问index的时候也许只用到a脚本，点击某个按钮才用到b脚本。这就引入下个话题："require.js"