### 严格模式下进行脚本链接的问题

### 一、严格模式

ES5引入了另一种版本控制的考量——严格模式（strict mode）。此特性与虚拟选择在受限制的Javascript版本中禁止使用一些Javascript语言中问题较多或者易于出错的特征。但是需要注意的是，被严格模式所判定错误的语句或者逻辑不一定是真正意义上是错误的，也许是逻辑上允许，但是可能会造成误会或者容易带来意想不到的错误的东西。

使用严格模式即在函数的最顶端或者整个程序的最顶端写上”use strict”；即可，不要落下“；”。使用字符串最为指令的语法虽然有点奇怪，但是可以实现向下兼容，对不能执行严格模式的浏览器（支持版本ES3及以前的或者特殊的浏览器）而言，这个字符串并不会带来任何副作用，解释引擎解释执行完这句话，就立刻抛弃其值。

"use strict";

function myFunction(){

//...

}

或者是

function myFunction(){

"use strict";

//...

}

注意”use strict”;必须在顶端！

### 严格模式下连接脚本需要注意的

我认为的连接脚本就像是webpack的打包，将多个独立的js文件连接成一个js文件。

书上有提到的是“连接成一个单一的文件”以及“脚本连接工具”的概念，上网查了什么是连接脚本，并没有得到明确的定义。似乎并不是在一个html文件中逐个引入js。经过试验，结论应该是前者。

在一个程序中，如果一个作用域的顶端声明了”use strict”，之后的use strict就没有意义了，并且整个程序都是在严格模式下。如果作用域最顶端并没有”use strict”，则整个程序都不再严格模式下，即使之后在相同作用域声明的非顶端处声明也不会进入严格模式。但是需要注意的是，我所说的相同作用域并不包含子作用域，也就是说，顶端没有，但是某个函数的顶端有，则那个函数在严格模式下处理。

因此，在连接脚本时，应该注意不同文件对“use strict”的声明。因为很可能会出现，某个文件不需要严格模式，但是连接在他前一个的js脚本却声明的严格模式，结果导致不需要的出现了错误。或者是需要严格模式的，但是因为上一个不是严格模式，结果后者的严格模式就被取消了。

"use strict"; // a.js 规定为严格模式

function a(){

// var arguments = [];

*console*.log("来自a");

}

/\*\*\*\*\*\*上面是a.js引来的，下面是b.js引来的\*\*\*\*\*\*\*\*\*\*/

// b并没有声明严格模式

function b(){

var arguments = []; // arguments是函数自带的内部对象，包含和参数相关的一些属性和方法。

// 在严格模式下不能重定义arguments对象。否则会报错，但是非严格可以。

*console*.log("来自b");

}

为了避免出现这样的问题，总结出了3个有效方法：

1. **不要将进行严格模式检查的文件和不进行严格模式检查的文件连接在一起。**

方法确实可行，但是会限制文件结构，给开发带来困难。

1. **通过将其自身包裹在立即调用的函数表达式（Immediately Incoked Function Expression，IIFE）中的方式连接多个文件。**

(function(){

"use strict"; // a.js 规定为严格模式

function a(){

*console*.log("来自a");

}

a();

})();

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*a.js b.js的代码都用IIFE包裹\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

(function(){

// b并没有声明严格模式

function b(){

var arguments = []; // arguments是函数自带的内部对象，包含和参数相关的一些属性和方法。

// 在严格模式下不能重定义arguments对象。否则会报错，但是分严格可以。

*console*.log("来自b");

}

b();

})();

这样的虽然解决了问题，但是还是很难受，因为每个js文件的内容都被放置在了一个单独的作用域中（IIFE形成了一个匿名函数，将每个文件全部代码的作用域放在了这个匿名函数的作用域中），因此var和function声明的全局变量和全局函数都将不会被视为全局变量。

如果去掉a(),b()。而将其移动到html文档的script标签中，会出现以下错误。

<script>

a();

b();

</script>



这正是因为函数的“全局性”已经没有了。

**（三）编写文件使其在两种模式下行为一致**

要想获得最大的兼容性，最简单的方式是在严格模式下编写代码，并显式的将代码本地一个立即调用的匿名函数中（IIFE）。说白了，就是永远都在严格模式下写代码。

(function(){

"use strict"

//...

function myFunction(){

//...

}

//....

})();

**（四）我的看法**

我的看法不保证正确，如果真的需要区分开严格模式或者费严格模式，可以使用第二种方式，然后通过闭包的方式将需要暴露在全局作用域内的变量或者函数暴露出来，使用的时候在进行引用暴露出的部分。

或者是主体部分在非严格模式下编写，就以控制严格模式或者非严格模式代码部分的作用域为原则，将需要特殊对待的代码进行函数封装，然后在那个函数中单独声明“use strict”即可。

**书中的提示：**

决定你的应用程序支持Javascript的哪些版本。

总是在执行严格模式检查的环境中测试严格代码。

当心连接那些在不同严格模式下有不同预期的脚本。