## 了解分号插入的局限

说实在的，这一章我并没有多理解。。。

Js的一个便利是能够离开语句结束分号的工作，删除分号可以让代码看上去十分优雅。

之所以Js能允许你不用自己手动插入分号，是因为Js可以根据ECMAScript标准制定额的分号插入机制自动插入。但是这也使分号的插入有一定的局限。

这个规则十分的复杂，我并不是很理解他的规则，我不打算使用省略分号的方式编程。

分号插入的第一条规则：

分号仅在}标记之前；、一个或者多个换行之后和程序输入的结尾被插入。换句话说，你只能在一行、一个代码块和一段程序结束的地方省略分号。

第二条规则是，分号仅在随后的输入标记不能解析的时候插入。

有5个明确有问题的自负需要注意：(、[、+、-和/，每一个字符都能作为一个表达式运算符或者一条语句的前缀。

a = b

["a","b","c"].forEach(function(*key*,*index*,*arr*){

// something to do...

})

因为[可以被认为是一个表达式的开始，因此，程序不会在b处结束。而是当成了

a = b["a","b","c"].forEach(function(*key*,*index*,*arr*){

// something to do...

})

这看上去似乎是非法的，但是事实上却是合法的，在js中存在“,”分割表达式。逗号分割表达式会从左至右依次执行，并返回最后一个表达式的值。

对于该例子，它返回字符“c”。

另一个常见的情况是，省略分号可能导致脚本链接问题。每个文件可能由大量的函数调用表达式组成。

比如：

// 文件A

(function(){

//......

})()

// 文件B

(function(){

//...

})()

因为(不会插入分号，所以解析的结果是

// 文件A

(function(){

//......

})()(function(){ // 文件B

//...

})()

很明显导致了错误。所以我们最好在每一段程序的结束都加上”;”

为了防止这类的错误，尤其是在程序的第一个字符为这些类似(脆弱的字符的时候，我们最好在程序的最开始加上一个分号。

;(function(){

//...

})()

但是，我们会认为只要在程序改加;的地方加上;不就得了。求稳不要优雅。

这样的话我们就掉以轻心，比如一个常见的坑：

return

{

...

};

事实上，这是错误的，他会被解析成多条语句，return后面也会加一个;，所以出现错误很正常。

再有就是for循环()中的分号js是不会自动加的。因此这样也不行：

for(var i

i<=10

i++){

//......

}

还有一点值得注意的是，如果省略分号

a

++

b

会被认为是

a;

++b;

这正是为了消除代码的歧义而产生的固定规则。