

立即执行函数的使用

闭包永远和立即执行函数一起用;

1. 我们不想要全局变量
2. 我们要使用局部变量
3. ES 5 里面, 只有函数有局部变量
4. 于是我们声明一个 function xxx, 然后 xxx.call()
5. 这个时候 xxx 是全局变量 (全局函数)
6. 所以我们不能给这个函数名字
7. function(){}.call()
8. 但是 Chrome 报错, 语法错误
9. 试出来一种方法可以不报错:
 1. !function(){}.call() (我们不在乎这个匿名函数的返回值, 所以加个 ! 取反没关系)
 2. (function(){}).call() 方方不推荐

```
xxx  
(function(){}).call() } → 有个bug
```

立即执行函数的使用

使用场景

编写代码时不想用全局变量, 只想使用局部变量

解决方案

ES5里面, 只有函数有局部变量, 可以声明一个function xxx, 然后 xxx.call()。此时xxx成为了全局变量 (全局函数), 违背初衷, 故采用匿名函数function(){}.call()。这种写法符合逻辑, 但是Chrome 报错, 提示语法错误。这种错误可以通过'取反'处理, 即最终写法为:

```
1 !function(){  
2     //其它代码  
3     }.call()
```

注: 上述写法并不在乎这个匿名函数的返回值, 所以加个!取反没关系

访问局部变量

通过全局变量window访问

```
1 !function(){  
2     var person = window.person { //局部、全局变量都存下匿名对象的地址  
3         name: 'tom'  
4     }  
5     console.log (person)  
6     console.log (window.person) //打印结果相同  
7 }.call  
8
```

```

9  !function(){
10     var person = window.person
11     console.log (person)
12     console.log (window.person) //打印结果同上
13 }.call

```

通过匿名函数与闭包的结合访问

通常匿名函数会和闭包、全局变量一起使用，形成封装。

```

1  !function(){
2     var person = {
3         name: 'tom',
4         age: 18
5     }
6     window.tomGrowup = function(){//闭包
7         person.age = person.age + 1
8         return person.age
9     }
10 }.call
11
12 !function(){
13     var newAge = window.tomGrowup () //间接操作age，且只能增加
14     console.log (newAge)
15 }.call
16 //如果没有闭包，这里的person就是全局变量
17 //1.立即执行函数使得person无法被外部访问
18 //2.闭包使得匿名函数可以操作person
19 //3.window.tomGrowup保存了匿名函数的地址
20 //4.任何地方都可以使用window.tomGrowup
21 //总结：任何地方都可以使用window.tomGrowup间接操作person，当不能直接访问person
22

```

下面代码与上同效

```

1  var accessor = function(){
2     var person = {
3         name: 'tom',
4         age: 18
5     }
6     return function(){//返回匿名函数
7         person.age = person.age + 1
8         return person.age
9     }
10 }
11 var growUp = accessor.call()//立即执行，返回值为函数

```

```
12 growUp.call()//继续执行
13 //1.声明一个匿名函数并立即执行
14 //2.accessor匿名函数返回值是另一个匿名函数
15 //3.将accessor匿名函数返回值赋给growUp
16 //4.再次执行growUpgrowUp函数
17
18
```

注：立即执行函数用来隔离作用域

闭包永远和立即执行函数一起用;