JavaScript闭包

> 闭包是JavaScript语言的一大特点，主要应用场合为：设计私有的方法和变量。

闭包三个特性

1. 函数嵌套函数
2. 函数内部可以引用外部的参数和变量
3. 参数和变量不会被垃圾回收机制回收

闭包的定义

闭包是指有权访问另一个函数作用域中的变量的函数，创建闭包的最常见的方式就是在一个函数内部创建另一个函数，通过另一个函数访问这个函数的局部变量。

闭包的缺点

闭包的缺点就是常驻内存，会增大内存使用量，使用不当很容易造成内存泄漏。

一般函数执行完毕后，局部活动对象就会被销毁，内存中仅仅保存全局作用域。但闭包的情况不同。

```
function out (){
    var n = 1;
    return function (){
        return n++;
    }
}
var fn = out();
console.log(fn()) //1
console.log(fn()) //2
console.log(fn()) //3
```

JavaScript的垃圾回收机制

1. 在JavaScript中，如果一个对象不再被引用，那么这个对象就会被GC回收。

2. 如果两个对象互相引用，而不被第三者所引用，那么这两个互相引用的对象也会被回收。

使用闭包的好处

那么使用闭包有什么好处呢？

1. 希望一个局部变量长期驻扎在内存中
2. 避免全局变量的污染
3. 私有成员的存在

局部变量长期驻扎内存

```
function out (){
    var n = 1;
    return function (){
        return n++;
    }
}
var fn = out();
console.log(fn())
console.log(fn())
console.log(fn())
```

在循环中直接找到对应的元素的索引

```
<ul>
   <li>1111111</li>
   <li>1111111</li>
   <li>1111111</li>
</ul>

var lis=document.getElementsByTagName("li");
for(var i=0;i<lis.length;i++){
    (function(i){
        lis[i].onclick=function(){
            alert(i);//0 1 2
        }
    })(i)
}
```

避免全局变量污染

但我们声明一个函数的时候其实就相当于创建一个全局变量
上一个例子中总共占有2个全局变量。
函数表达式 函数自调用：

```
(function(){
})()
```

```
var out = (function(){
    var n1 = 1;
    var n2 = 100;
    return function(){
        n1++;
        n2--;
        return n1+n2;
    }
})()
out() // 1
out() // 2
```

私有成员的存在

模块化代码

```
var aa=(function(){
   var a=10;
   function aaa(){
       a++;
       alert(a);
   }
   function bbb(){
       a+=10;
       alert(a);
   }
   return {
      a:aaa,
      b:bbb
   }
})()
aa.a(); //11
aa.b(); //21
```

这样做的好处就是 bbb aaa 这两个函数我们在外面是访问不到的