JavaScript闭包（closure）

github.com/tsrot JavaScript闭包

地址：http://blog.xieliqun.com/2016/08/07/javascript-closure/

> JavaScript变量分为全局变量和局部变量。Javascript语言的特殊之处，就在于函数内部可以直接读取全局变量，而在函数外部自然无法读取函数内的局部变量。当你需要在函数外调用函数内的局部变量时，此时就要用到一些方法。这个过程就是闭包。

一、JavaScript为什么会有闭包这种东西

JavaScript没有像其它后端语言一样可以直接定义一个变量可供其它外部函数调用的关键字或者方法。于是就产生了闭包这种东西。举个例子：

```
//内部函数可以访问外部变量
var name = "tsrot";
function f1(){
    console.log(name);
}
f1();   //tsrot
```

```
//函数外部无法访问函数内部的局部变量
function f2(){
    var name = "tsrot"; //注意:不用var定义的变量，将会默认为全局变量
}
console.log(name); //error undefined
```

如果是java，一个类的私有属性，可以通过公共的方法来获取，比如：

```
class Person{
    private String name;
    public String getName(){
       return name;
    }
}
```

二、闭包的概念

闭包，官方对闭包的解释是：一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数），因而这些变量也是该表达式的一部分。闭包的特点：

- 作为一个函数变量的一个引用，当函数返回时，其处于激活状态。
- 一个闭包就是当一个函数返回时，一个没有释放资源的栈区。

我的理解就是：让函数外部能调用函数内部变量的一个过程就是闭包。举个例子：

```
function f1(){
    var name1 = "tsrot";
    function f2(){
        return name1;
    }
    return f2;
}
var fun = f1();     //此时就访问到了name1的值
console.log(fun()) //tsrot
```

三、闭包的用途

闭包可以用在许多地方。它的最大用处有两个，一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中。举个例子：

```
function f1(){
    var n = 0;
    function f2(){
        n++;
        console.log(n);
    }
    return f2; //注意此时应该写f2而不是f2()
}
var fun = f1();
fun();  //1
fun();  //2  此时n就保存在了内存中
```

四、闭包的写法

1、原型调用写法

```
//在函数内部添加属性，然后在外部调用
function Circle(r) {
    this.r = r;
}
Circle.PI = 3.14159;
Circle.prototype.area = function() {
    return Circle.PI * this.r * this.r;
}
var c = new Circle(1.0);
console.log(c.area());
```

2、函数赋值调用写法

```
var Circle = function() {
   var obj = new Object();
   obj.PI = 3.14159;
   obj.area = function(r) {
       return this.PI * r * r;
   }
   return obj;
};
var c = new Circle();
console.log(c.area(1.0));
```

3、对象赋值调用写法（常用写法）

```
var Circle = new Object();
Circle.PI = 3.14159;
Circle.area = function(r) {
       return this.PI * r * r;
};
consol.log(Circle.area(1.0));
```

4、声明对象调用写法（比较好的一种写法）

```
var Circle={
    PI : 3.14159,
    area : function(r){
              return this.PI * r * r;
        }
};
console.log(Circle.area(1.0))
```

5、Function对象调用写法

```
//比较少见的一种写法
var Circle = new Function("this.PI = 3.14159;this.area = function( r ) {r
eturn r*r*this.PI;}");
var c = new Circle();
console.log(c.area(1.0));
```

6、匿名函数调用写法（常用写法）

```
(function(r){
    var PI = 3.14159;
    var area = PI * r * r;
    console.log(area);
})(1.0)
```

7、函数返回值写法（常用写法）

```
function Circle(r){
    var PI = 3.14159;
    function area(){
        return PI * r * r;
    }
    return area();
}
console.log(Circle(1.0));
```

五、思考

思考1：修改下面函数使之输出数组内元素？

```
function test(){
    var arr = [1,2,3,4,5];
    for(var i=0;i<arr.length;i++){
        setTimeout(function(){
            console.log(arr[i]);
        },1000);
    }
}
test();
```

思考2：修改下面代码使之输出它想表达的结果

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <ul id="list">
        <li>1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
        <li>5</li>
    </ul>
<script>
    var list = document.getElementById('list');
    var li = list.getElementsByTagName('li');
    for(var i=0;i<li.length;i++){
        li[i].onclick = function(){
            alert(li[i].innerHTML); //这里报错
        }
    }
</script>
</body>
</html>
```

答案1

```
function test(){
    var arr = [1,2,3,4,5]
    for(var i=0;i<5;i++){
        (function(i){
            setTimeout(function(){
                console.log(arr[i]);
            },1000);
        })(i)
    }
}
test();
```

答案2

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
</head>
<body>
    <ul id="list">
        <li>1</li>
        <li>2</li>
        <li>3</li>
        <li>4</li>
        <li>5</li>
    </ul>
<script>
    var list = document.getElementById('list');
    var li = list.getElementsByTagName('li');
    for(var i=0;i<li.length;i++){
        (function(i){
            li[i].onclick = function(){
                alert(li[i].innerHTML);
            }
        })(i)
    }
</script>
</body>
</html>
```