this相关

this是js中特有的一个特殊对象，它代表的是使用this对象的当前环境。
不管是对于什么语言，程序都在运行在计算机之中。
对于值对象来说，数据都是直接存在栈之中的，都直接可以在栈中取到数值。
对于引用对象来说，只是负责把引用存储在栈之中，引用指向的堆内存才是存放对象的真正地方。

总结回来对于this来说，this其实就是指向引用对象的上一级对象是什么。
在js中函数都说可以直接在外部运行的，如下：

```
obj:{
    age:1,
    getAge():function(){
        this.age;
    }
}
age=2;
getAge();
```

如果是obj.getAge()的话，就确定是obj指向的getAge，那么getAge()中的this即是obj,所以值是1
如果是直接getAge()的话，那么调用的对象就是全局对象代表是this，那么this.age就是2

闭包相关

闭包的概念：为了能让函数外部能使用内部变量的函数，通常都是通过返回闭包来实现的，所以闭包其实也是定义在函数内部的函数。
正常来说在函数的外部是不能使用函数的内部变量的，但是闭包的产生就是为了解决这个问题。
闭包主要有两个作用：一个是能在函数的外部使用函数的内部变量，第二个是能让变量在内存中能长久的存在，而不至于被回收。
对于第二个作用来说：系统的内存回收机制是根可达原理，如果所使用的对象还能被引用链接到的话就不会被内存回收。对于闭包来说，后面是赋值给了一个外部的全局变量所以

会一直存在。但是如果大量用闭包的话，会导致内存增加，解决方法是，在退出函数之前，将不使用的局部变量全部删除。

代码片段一。

```
var name = "The Window";
var object = {
    name : "My Object",
    getNameFunc : function(){
        return function(){
            return this.name;
        };
    }
};
alert(object.getNameFunc()());
```

这个例子中this是object对象所以是 my object

代码片段二。

```
var name = "The Window";
var object = {
    name : "My Object",
    getNameFunc : function(){
        var that = this;
        return function(){
            return that.name;
        };
    }
};
alert(object.getNameFunc()());
```

这个例子中object.getNameFunc()返回的是闭包对象function()，然后function()()现在这个对象已经相当于是全局函数使用，this就是全局对象了

（完）