JavaScript进阶

---JS闭包（closure）

河北师范大学软件学院
Software College of Hebei Normal University

内容提纲

- **闭包的概念**
- **闭包的常见形式**
- **闭包的作用及常用场景**

闭包（引入案例）

- 思考：函数内的局部变量，是否能在函数外得到
- 有什么方法能读写函数内部的局部变量

```
function f1(){
    var x = 1;
    function f2(){
        return x++;
    }
    return f2();
}
var f3 = f1();
//f1中的x变量是否被释放
console.log(f3);//输出?
console.log(f3);//输出?
```

```
function f1(){
    var x = 1;
    function f2(){
        return x++;
    }
    return f2;
}
var f3 = f1();
//f1中的x变量是否被释放
console.log(f3());//输出?
console.log(f3());//输出?
```


思考上述两段代码的区别，观察f1中的x变量，在chrome中进行断点调试

闭包（closure）的概念

- 闭包是由函数和与其相关的引用环境组合而成的实体
- 闭包是词法作用域中的函数和其相关变量的包裹体

```
function createInc(startValue){
    return function(step){
        startValue+=step;
        return startValue;
    }
}
var inc = createInc(5);
console.log(inc(1));//输出多少？
console.log(inc(2));//输出多少？
inc = createInc(5);
console.log(inc(1));//输出多少？
```

前两次输出中，startValue常驻内存

第三次输出前，新创建了一个闭包，startValue重新创建

思考：若将倒数第二行的inc = createInc（5）改为var inc2 = createInc（5）则：inc（1）和inc2（1）为多少

参见实例demo11中的 例一、例二

闭包（closure）的概念

- 若一个函数离开了它被创建时的作用域，它还是会与这个作用域的变量相关联
- 闭包是一个函数外加上该函数创建时所建立的作用域

```
function foo() {
    var i = 0;
    function bar() {
        console.log(++i);
    }
    return bar;
}
var a = foo();
var b = foo();
a();//1
a();//2
b();//1
```

函数bar和其相关词法上下文中的变量i，构成了一个闭包

返回的函数bar，依然能够访问到变量i（藕断丝连）

a和b对应的是否为同一个闭包?

思考：foo和它相关作用域的变量是否形成闭包?

参见实例demo11中的 例三、例四

内容提纲

- 闭包的概念
- 闭包的常见形式
- 闭包的作用及常用场景

闭包的常见形式（以函数对象形式返回）

```
var tmp = 100;//注意：词法作用域
function foo(x) {
    var tmp = 3;//注意：词法作用域
    return function (y) {
        console.log(x + y + (++tmp));
    }
}
var fee = foo(2); // fee 形成了一个闭包
fee(10);//
fee(10);//
fee(10);//
```

思考：若屏蔽此行，则又会输出多少？

思考：此实例中fee函数对象相关作用域的变量都有哪些？形成的闭包是否包含foo函数之外（即第一行）的自由变量tmp？foo中的tmp是否调用后就释放？

参见实例demo12 Part1

闭包的常见形式（以函数对象形式返回）

```
function foo(x) {
    var tmp = 3;
    return function (y) {
        x.count = x.count ? x.count + 1 : 1;
        console.log(x + y + tmp,x.count);
    }
}
var age = new Number(2);
var bar = foo(age); //和相关作用域形成了一个闭包
bar(10); //输出什么?
bar(10); //输出什么?
bar(10); //输出什么?
```

思考：此实例中bar函数对象相关作用域的变量都有哪些？foo中的tmp是否调用后就释放?

使用断点调试查看代码的运行状况

参见实例demo12 Part2

闭包的常见形式（作为对象的方法返回）

```
function counter() {
    var n = 0;
    return {
        count:function () {return ++n;},
        reset:function () {n = 0;return n;}
    }
}
var c = counter(),d = counter();
console.log(c.count());
console.log(d.count());
console.log(c.reset());
console.log(c.count());
console.log(d.count());
```

思考：此实例中总共有几个闭包？

使用断点调试查看代码的运行状况

参见实例demo13 Part1

内容提纲

- 闭包的概念
- 闭包的常见形式
- **闭包的作用及常用场景**

闭包的作用

- 可通过闭包来访问隐藏在函数作用域内的局部变量
- 使函数中的变量被保存在内存中不被释放（单例模式）

```
function f1(){
    var n = 999;
    function f2(){
        console.log(++n);
    }
    return f2;
}
var f = f1();
f();//输出多少?
f();//输出多少?
```

左侧实例中，无法在f1函数外直接得到变量n的值，可以通过闭包间接的在f1函数外访问和修改n

注意：由于闭包的存在n在f1调用后并不直接释放

参见实例demo14 参见相关实例

闭包的实际应用案例

```
function fn() {
    var a;
    return function() {
        return a || (a = document.body.appendChild(document.createElement('div')));
    }
};
var f = fn();
f();

function closureExample(objID, text, timedelay) {
    setTimeout(function() {
        //document.getElementById(objID).innerHTML = text;
        console.log(objID,text);
    }, timedelay);
}
closureExample("myDiv","Closure is Create", 1000);
```

单例模式实例：因为闭包，所以a常驻内存，始终存在

定时修改DOM节点案例，1秒后执行，仍能访问到objID

实例demo15 闭包应用案例
实例demo16 index16.html 函数相关操作综合实例

闭包的注意事项

- 由于闭包会使得函数中的变量都被保存在内存中，内存消耗很大，所以不能滥用闭包
- 使用闭包时要注意不经意的变量共享问题，可以通过立即执行表达式来解决

Thank You!

作业

- codefordream网站上JavaScript基础-初级训练营
- 雪梨上完成任务（要求有截图，体现完成的项目，用户名，完成的程度）
- 复习本章内容及练习