第十课 中介者模式

学习目录

- **使用场景**
- **使用方式**
- **优缺点**

一. 使用场景

有时候我们在编写代码的过程中会按照面向过程的逻辑来编写，许多对象相互之间会有不少相互调用的紧耦合关系，其实这在之后的应用维护阶段是很蛋疼的事情。俗话说的好前人挖坑，后人填坑，从一个公司到另一个公司，难的不是开发新项目，难的是维护之前人开发的项目。

其实在开发好一个项目之后，很有必要对项目代码中的紧耦合情况做适当调整与优化，一个是方便之后的同事接手，另一方便也方便代码的维护与更新。中介者模式就是为解除对象间的紧耦合而来的，使用中介者模式可以让对象与对象之间不用相互了解和调用，大家通过第三方中介者对象就能进行通信和调用，当一个对象发生改变时，只需要通知中介者即可，中介者在用其他方法通知其他需要进行改变的对象，这样特别方便。

就好比火车一样，不同的普通火车、动车、高铁相互之间只要负责把车运行好即可，至于调度工作就由车站调度室来做好就可以了。

中介者模式的核心思想是让一个对象尽可能少的去了解另一个对象，通过中介者对象建立他们之间的相互联系以及相互调用。

二．使用方式

先来看下基本的中介者模式的基础结构，在不使用中介者模式的情况下，一个对象要想使用另一个对象，必须要耦合在一起，通过中介者可以让这个对象间接的调用另一个对象。

加减次数

```
function Fn1(num){
    this.num = num;
}
Fn1.prototype.add = function(){
    this.num ++ ;
    middleFn.addFn(this.num);
}
Fn1.prototype.less = function(){
    this.num -- ;
    middleFn.lessFn(this.num);
}

var middleFn = {
    addArr:[],
    lessArr:[],
    init:function(num){
```

```
            var fn1 = new Fn1(num);
            return fn1;
        },
        addFn:function(num){
            this.addArr.push(num);
        },
        lessFn:function(num){
            this.lessArr.push(num);
        },
        getCount:function(){
            return {
                addNum:this.addArr.length,
                lessNum:this.lessArr.length
            }
        }
    }

    var wrapFn = function(){
        var $div = document.createElement('div');
        $div.innerHTML = '增加：' + middleFn.getCount().addNum + '减少：' +
middleFn.getCount().lessNum;
        document.body.appendChild($div);
```

```
}

var f1 = middleFn.init(100);
f1.add();
f1.add();
f1.less();
wrapFn();
```

三．优缺点

1.中介者模式可以简化对象或者构造函数的复杂度，单个对象只关注于单一职责。

2.中介者模式可以解除两个对象之间的强耦合关系，让某一个对象的修改不会过多的影响到另一个对象。

3.中介者模式中会产生一个中介者对象需要维护，随着程序复杂度增加，维护难度也会增加。

4.中介者模式在耦合对象职责不明确的时候不要随便使用，因为这样可能还会增加代码的复杂度。

谢谢观看！

我是星星课堂老师：周小周