第四课 策略模式

学习目录

- **使用场景**
- **使用方式**
- **优缺点**

一．使用场景

在我们日常 js 程序开发，肯定经常遇到 if else 逻辑判断条件语句，当逻辑语句的分枝很多时候，我们的语句就会写的越来越长，然后可能在语句内部还会嵌套条件语句，这样就会让代码看上去难以理解，假如加入新的条件语句只能让代码变得冗余。如果用 switch 语句来改写，其实只是在阅读代码上做了优化，本质上还是大量的条件语句。

还有一种情况更能体现策略模式的定义，当我们要实现某一个功能时可以由多种方式，这些方式都是为了实现这个功能，在生活中，我们要购买一套组合家具，我们可以有多种购买渠道，每种渠道的购买价格不同，服务不同，但是最终都能帮助我们购买家具。在程序开发中，比如我们要上传文件，在不同的条件下我们需要不同的上传方法，又比如我们要编写一套计算公司年终奖费用的算法，因为每个人的级别不同，工作成果不同，因此不同的员工需要不同的计算方式，还有我们之前在 js 进阶与组件化的课程中为大家编写的表单验证的组件，其中也用到了策略模式。

以上两种情况都可以采用策略模式来实现，既能保证程序逻辑正确，又能让程序结构看起来更清晰，最重要的是实现了代码的解耦，提升了代码的可维护性。

策略模式的定义：把需要定义的某一组算法一个一个的封装起来，让他们之间相互可以

替换，每个算法内部逻辑不同，但是可以在不同场景下相互替换使用。

策略模式的核心思想是把算法调用和算法本身分开。

二．使用方式

1.年终奖金计算

普通函数版

```
var calcFn = function(level,money){
    if(level == 'A+'){
        return money * 3
    }
    if(level == 'A'){
        return money * 2
    }
    if(level == 'B'){
        return money * 1.5
    }
    if(level == 'C'){
        return money * 1
    }
    if(level == 'D'){
        return money * 0.8
    }
```

```
}
console.log(calcFn('B',10000));
```

不同的员工，基本工资不同，绩效考核的年终奖系数也不同，因此要区别不同员工得到不同的年终奖金。

策略模式版

```
var levelObj = {
    'A+':function(money){
        return money * 3
    },
    'A':function(money){
        return money * 2
    },
    'B':function(money){
        return money * 1.5
    },
    'C':function(money){
        return money * 1
    },
    'D':function(money){
        return money * 0.8
    }
```

```
}
var calcWrapFn = function(level,money){
    return levelObj[level](money);
}
console.log(calcWrapFn('B',10000));
```

2.普通逻辑处理

```
var conditionObj = {
    'condition1': function (args) {
        console.log('condition1' + args);
    },
    'condition2': function (args) {
        console.log('condition2' + args);
    },
    'condition3': function (args) {
        console.log('condition3' + args);
    }
}
var doFn = function(condition,args){
    return conditionObj[condition](args);
}
doFn('condition1','星星课堂');
```

三．优缺点

1.策略模式是非常符合的开放封闭原则的模式，它是的算法独立于逻辑代码中，巧妙的把算法策略封装在了一个对象中，由另一个函数负责调用这些算法，调用和封装完全分开，更加有利于代码扩展与维护。

2.策略模式解决了大量条件语句带来的代码冗余。

3.策略模式会增加策略对象，如果策略对象过于庞大，也需要大量时间进行维护。

4.策略模式在使用时，使用者必须清楚不同策略算法的内部逻辑才能选择合适的算法。

谢谢观看！

我是星星课堂老师：周小周