

初识代理模式!

=====

什么是代理模式

代理模式 (Proxy Pattern) 是一种设计模式，用于为其他对象提供一种代理以控制对这个对象的访问。代理模式可以用于多种不同的场景，比如虚拟代理、远程代理、保护代理和智能指引等。

今天，我们就以“小明送花给小美”的温馨故事为背景，通过JavaScript的代理模式，演绎一场代码编织的浪漫剧目，展现如何将人际间的细腻情商融入到严谨的编程逻辑中，让每一次的代码敲击都充满智慧和温度。

角色设定与代码实现

> 首先，这里的场景是男主角小明遇到了自己的心仪的女生名叫小美，一见钟情！恰逢520啊，此时小明决定给小美送上一捧玫瑰，来表达自己的心意。

1. 小明和小美登场

用JavaScript来描述就是先创建小明和小美对象，小明有送花的sendFlower方法，而小美则是有receiveFlower方法表示收花的行为。

...

```
const xm = {  
  name: '小明',  
  school: '东华理工大学',  
  hometown: '鹰潭',  
  
  sendFlower: function (target) {  
    target.receiveFlower(this)  
  }  
}
```

```
const xm = {
```

```

    name: '小美',

    receiveFlower: function(sender) {
        console.log(`${this.name}收到了来自${sender.name}送出的花`)
    }
}

```

...

这样简单的情景显然无法满足现实生活的需求。现实中，小明作为与小美初遇的男生，考虑到如果直接向小美送花可能多有不妥，如小美已有男朋友，此时就需要第三人的出现来解决这一问题，也就是小美的好朋友小丽登场！

2. 小丽登场

> 在我们的故事里，小丽作为小美的代理，不仅具有接收鲜花的相同接口，还能在特殊情况下代表小美收下这份心意。

...

```

const xl = {
    name: '小丽',
    hometown: '鹰潭',

    receiveFlower: function(sender) {
        // 代替小美收花 实现同样的方法，也就是接口
        if(xm.mood < 90) {
            console.log('等一等');
            setTimeout(function() {
                xm.mood = 92
                xm.receiveFlower(sender)
            },2000)
        }else {
            xm.receiveFlower(sender)
        }
    }
}

```

...

...

```

const xm = {
    name: '小美',

```

```
mood: 50,

receiveFlower: function(sender) {
    console.log(`${this.name}收到了来自${sender.name}送出的花`)
    if(this.mood >= 90) {
        console.log("万达走一波");
    }else {
        console.log('不约，我们不约');
    }
}
}
}
...

```

这里我们合理的给小美添加了mood情绪值属性，当mood大于90时，代表小美的心情很不错，可以接受小明送出的花。

送花的情节展开

终于到了送花的那一刻，小明通过小丽，成功地传达了他的心意：

起初的小美心情低落，小丽告知小明先不要送花。

待小美心情好转，小丽告知小明把握机会送出花去，得到了小美的认可，最后和小美没羞没燥的在一起了！

![image.png](https://p1-juejin.byteimg.com/tos-cn-i-k3u1fbpfcp/fe623ad456a64754b14dac239d12e026~tplv-k3u1fbpfcp-jj-mark:3024:0:0:0:q75.awebp#?w=298&h=114&s=7111&e=png&b=ffffff)

情商到码商的转变

通过这个简短的代码演示，我们见证了代理模式如何在JavaScript中优雅地解决了“小明送花给小美”的问题，更重要的是，它展示了设计模式如何将现实生活中的“情商”转化成代码中的“码商”。代理模式不仅限于此，它在诸如远程服务调用、权限控制、懒加载等领域都有着广泛的应用。在编写代码时，思考如何利用设计模式解决复杂问题，是每位开发者向更高层次进阶的关键一步。让代码不仅仅是执行逻辑的堆砌，更是思维智慧的体现，正如JavaScript所展现的那样，用最简洁的语言表达最复杂的思想，将“情商”化为无形，流淌在每一行代码之中。

原文链接: <https://juejin.cn/post/7369761632275021834>