淘淘笙悦
2018年10月01日 阅读 4619

关注

谈谈观察者模式和发布订阅模式

在网上看到许多关于观察者模式和发布订阅模式的博文，发现很多人都认为观察者模式即发布订阅模式，经过进一步的学习和理解，我认为观察者模式和发布订阅模式还是有一些区别的，下面谈谈我对观察者模式和发布订阅模式的理解「PS：欢迎各路大神指正」。

观察者模式（Observer）

观察者模式指的是一个对象（Subject）维持一系列依赖于它的对象（Observer），当有关状态发生变更时 Subject 对象则通知一系列 Observer 对象进行更新。

在观察者模式中，Subject 对象拥有添加、删除和通知一系列 Observer 的方法等等，而 Observer 对象拥有更新方法等等。

在 Subject 对象添加了一系列 Observer 对象之后，Subject 对象则维持着这一系列 Observer 对象，当有关状态发生变更时 Subject 对象则会通知这一系列 Observer 对象进行更新。

复制代码

```
function Subject(){
  this.observers = [];
}

Subject.prototype = {
  add:function(observer){  // 添加
    this.observers.push(observer);
  },
  remove:function(observer){  // 移除
    var observers = this.observers;
    for(var i = 0;i < observers.length;i++){
      if(observers[i] === observer){
        observers.splice(i,1);
      }
    }
  },
  notify:function(){  // 通知
    var observers = this.observers;
    for(var i = 0;i < observers.length;i++){
      observers[i].update();
    }
  }
}

function Observer(name){
  this.name = name;
}

Observer.prototype = {
  update:function(){  // 更新
    console.log('my name is '+this.name);
  }
}

var sub = new Subject();

var obs1 = new Observer('ttsy1');
var obs2 = new Observer('ttsy2');

sub.add(obs1);
sub.add(obs2);
sub.notify();  //my name is ttsy1、my name is ttsy2
```

上述代码中，我们创建了 Subject 对象和两个 Observer 对象，当有关状态发生变更时则通过 Subject 对象的 notify 方法通知这两个 Observer 对象，这两个 Observer 对象通过 update 方法进行更新。

在 Subject 对象添加了一系列 Observer 对象之后，还可以通过 remove 方法移除某个 Observer 对象对它的依赖。

复制代码

```
var sub = new Subject();

var obs1 = new Observer('ttsy1');
var obs2 = new Observer('ttsy2');

sub.add(obs1);
sub.add(obs2);
sub.remove(obs2);
sub.notify();  //my name is ttsy1
```

发布订阅模式（Publisher && Subscriber）

发布订阅模式指的是希望接收通知的对象（Subscriber）基于一个主题通过自定义事件订阅主题，被激活事件的对象（Publisher）通过发布主题事件的方式通知各个订阅该主题的 Subscriber 对象。

复制代码

```
let pubSub = {
  list:{},
  subscribe:function(key,fn){  // 订阅
    if (!this.list[key]) {
      this.list[key] = [];
    }
    this.list[key].push(fn);
  },
  publish:function(){  // 发布
    let arg = arguments;
    let key = [].shift.call(arg);
    let fns = this.list[key];

    if(!fns || fns.length<=0) return false;

    for(var i=0,len=fns.length;i<len;i++){
      fns[i].apply(this, arg);
```


目录

```
    }

  },
  unSubscribe(key) {  // 取消订阅
    delete this.list[key];
  }
};

pubSub.subscribe('name', (name) => {
  console.log('your name is ' + name);
});
pubSub.subscribe('sex', (sex) => {
  console.log('your sex is ' + sex);
});
pubSub.publish('name', 'ttsy1');  // your name is ttsy1
pubSub.publish('sex', 'male');  // your sex is male
```

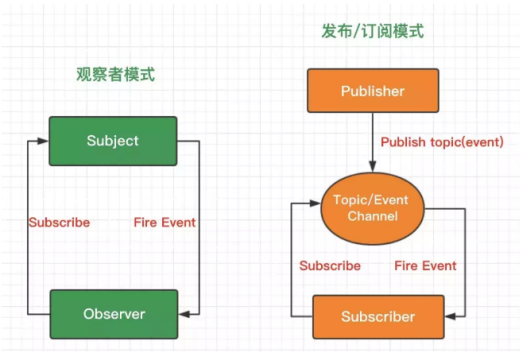
上述代码的订阅是基于 name 和 sex 主题来自定义事件，发布是通过 name 和 sex 主题并传入自定义事件的参数，最终触发了特定主题的自定义事件。

可以通过 unSubscribe 方法取消特定主题的订阅。

复制代码

```
pubSub.subscribe('name', (name) => {
  console.log('your name is ' + name);
});
pubSub.subscribe('sex', (sex) => {
  console.log('your sex is ' + sex);
});
pubSub.unSubscribe('name');
pubSub.publish('name', 'ttsy1');  // 这个主题被取消订阅了
pubSub.publish('sex', 'male');  // your sex is male
```

观察者模式 VS 发布订阅模式



观察者模式与发布订阅模式都是定义了一个一对多的依赖关系，当有关状态发生变更时则执行相应的更新。

不同的是，在观察者模式中依赖于 Subject 对象的一系列 Observer 对象在被通知之后只能执行同一个特定的更新方法，而在发布订阅模式中则可以基于不同的主题去执行不同的自定义事件。相对而言，发布订阅模式比观察者模式要更加灵活多变。

我认为，观察者模式和发布订阅模式本质上的思想是一样的，而发布订阅模式可以被看作是观察者模式的一个进阶版。

设计模式只是一种思想，某一种设计模式都可以有很多种不同的实现方式，各种实现都有其优劣之分，具体的实现方式需要基于不同的业务场景。上述是我对观察者模式和发布订阅模式学习之后的一些理解，望指正。

不定时分享个人在前端方面的学习经验，觉得还不错的小伙伴，可以关注一波公众号哦。

woow_wu7 Lv1

update挂在观察者对象的原型上，实例不会相互影响吗？不太合理把

5月前 回复

superchris Lv1 java

总结的很好

5月前 回复

萧萧fe Lv2 前端打字员 @ pdd

独秀

1年前 回复

小郭pp Lv1 前端开发工程师 @ ~

1.观察者模式维护的是一个单一事件对应多个依赖这个事件的对象之间关系
观察者是: event->[obj1,obj2obj3,....]
2.订阅发布模式维护的是多个主题(事件) 以及依赖于各个主题(事件)的对象之间的关系
订阅发布是:{
event1->[obj1,obj2.....

展开

1年前 13 回复

lx1036 Lv2 PHP/Go/TypeScript @ 3...

回复 小郭pp Lv1: 同意上面的理解。

1年前 回复

淘淘至悦 Lv3 (作者) 野生小前端 @ 晓教育

回复 小郭pp Lv1: 嗯嗯，对的。

1年前 回复

相关推荐

专栏 · Momomo · 18小时前 · 前端

[译]JavaScript 模块打包方案

18

专栏 · 非常记得你 · 9天前 · 前端

初、中级前端应该要掌握的手写代码实现

517 46

专栏 · 前端瓶子君 · 12天前 · 前端

是时候抛弃Postman了，试试 VS Code 自带神器插件👏👏👏

640 81

专栏 · isboyjc · 10天前 · JavaScript / 前端

「硬核JS」一次搞懂JS运行机制

644 59

专栏 · 恋猫de小郭 · 10天前 · 前端 / Flutter

我对移动端现状的看法

128 140

专栏 · 刘小夕 · 12天前 · 前端 / Visual Studio Code

动图演示23个鲜为人知的VSCode快捷键

619 44

专栏 · 染陌同学 · 14天前 · JavaScript / 前端

染陌的 2019 年度总结——我在阿里云做前端

365 81

热 · 专栏 · yck · 1月前 · JavaScript

看完跳槽少说涨 5 K，前端面试从准备到谈薪完全指南（近万字精华）

2856 217

专栏 · 汤小梦 · 12天前 · 前端

从零实现一套属于自己的UI框架-发布到npm

363 53

专栏 · channnxy · 1天前 · 前端

浅谈JavaScript闭包和作用域问题

23