第十五课 发布订阅模式

学习目录

- **使用场景**
- **使用方式**
- **优缺点**

一．使用场景

有时候我们想找对象，我们有哪些渠道找对象呢，要么我们自己学习工作生活中自己谈一个，要么呢家里七大姑八大姨给我们介绍一个，还有呢我们可以找个婚介中心给我们推荐适合我们的对象。当然了不管哪种，找到自己喜欢的对象才是最重要的。

假如我们从婚介中心找对象，那么婚介中心为了赚我们的中介费，肯定会尽心尽力的给我们推荐各种符合我们要求的对象资料，我们收到这些对象资料就可以做自己的挑选，有时候婚介中心定期会更新会员资料，我们不需要去婚介中心他们直接一个微信消息就能把最新更新的符合我们需要的对象资料发给我们，我们要做的就是用微信关注婚介中心即可，这种找对象的方法其实我们订阅婚介中心，婚介中心发布对象资料。

在 js 中发布订阅模式的原理跟找对象是差不多，我们需要定义一个发布订阅模式的对象，从这个对象中，我们可以使用它提供的订阅方法和发布方法，因此我们可以保证其他对象之间的通信由发布订阅模式对象来中转，其他对象之间也不需要进行耦合调用。

发布订阅模式的核心思想是用一个对象中心来为不同的对象之间构建事件通信服务，订阅者订阅指定事件到对象中心，当发布者发布这个事件到对象中心时，所有订阅者能够获取到这个事件的数据。

二．使用方式

婚介中心发布订阅模式

```
var blindDateCenter = (function(){
    var blindDateData = {};
    var _listenFn = function(type,fn){
        if(!blindDateData[type]){
            blindDateData[type] = [];
        }
        blindDateData[type].push(fn);
    }
    var _emitFn = function(){
        var type = Array.prototype.shift.apply(arguments);
        var fnEvent = blindDateData[type];
        if(fnEvent && fnEvent.length > 0){
            for(var i=0;i<fnEvent.length;i++){
                var fn = fnEvent[i];
                fn.apply(this,arguments);
            }
        }else{
            console.log('没有人订阅'+type+'相亲数据');
            return;
```

```
            }
        }
        return {
            listen:_listenFn,
            emit:_emitFn
        }
    })();

    blindDateCenter.listen('thinGirl',function(e){//小张订阅瘦小姐姐相亲信息
        console.log(e);
    });
    blindDateCenter.listen('fatGirl',function(e){//小程订阅胖小姐姐相亲信息
        console.log(e);
    });

    var girlData = {
        name:'小姐姐',
        age:21,
        height:'170cm',
        weight:'45kg',
        money:20000
    };
```

```
blindDateCenter.emit('thinGirl',girlData);//发布瘦小姐姐相亲信息


var girlData = {
    name:'小姐姐',
    age:21,
    height:'170cm',
    weight:'80kg',
    money:20000
};
blindDateCenter.emit('fatGirl',girlData);//发布胖小姐姐相亲信息
```

。从这例子中我们已经可以保证订阅者可以根据发布者发布的事件获取到相关数据，但是有时候可能婚介中心介绍的几个对象都不合适，我们就不想让他介绍了，那我们是不是就要取消这个订阅，因此我们还可以增加一个取消订阅的方法。

同时我们这个方法不能仅仅只给婚介中心使用，我们是想这个发布订阅能够在全局状态下使用，因此我们可以改写一下这个对象模式。

全局发布订阅模式（增加取消订阅方法）

```
var publishSubscribe = (function(){

    var _data = {};

    var _listen = function(type,fn){
```

```
        if(!_data[type]){
            _data[type] = [];
        }
        _data[type].push(fn);
    }

    var _emit = function(){
        var type = Array.prototype.shift.apply(arguments);
        var fnEvent = _data[type];
        if(fnEvent && fnEvent.length > 0){
            for(var i=0;i<fnEvent.length;i++){
                var fn = fnEvent[i];
                fn.apply(this,arguments);
            }
        }else{
            console.log('没有订阅者订阅' + type + '的发布事件');
            return;
        }
    }

    var _remove = function(type,fn){
        var fnEvent = _data[type];
```

```
            if(!fn){
                fnEvent = [];
                fnEvent.length = 0;
                return;
            }
            if(!fnEvent){
                return;
            }
            for(var i=0;i<fnEvent.length;i++){
                if(fnEvent[i] == fn){
                    fnEvent.splice(i,1);
                }
            }
        }

        return {
            listen:_listen,
            emit:_emit,
            remove:_remove
        }

})();
```

```
var xiaozhangFn = function(e){//小张订阅瘦小姐姐相亲信息
    console.log('小张');
    console.log(e);
}

var xiaoxuFn = function(e){//小徐订阅瘦小姐姐相亲信息
    console.log('小徐');
    console.log(e);
}

publishSubscribe.listen('thinGirl',xiaozhangFn);
publishSubscribe.listen('thinGirl',xiaoxuFn);
publishSubscribe.remove('thinGirl',xiaozhangFn);

var girlData = {
    name:'小姐姐',
    age:21,
    height:'170cm',
    weight:'45kg',
    money:20000
};
```

```
publishSubscribe.emit('thinGirl',girlData);//发布瘦小姐姐相亲信息
```

三．优缺点

1.发布订阅模式可以避免模块与模块之间的耦合，假如我们要封装两个模块，使用发布订阅模式让他们之间相互通信时最好的方式。

2.发布订阅模式还可以用来处理异步编程问题，因为可以等待异步操作完成之后在让订阅者进行其他操作，这样就避免了大量的异步回调函数的嵌套。

3.发布订阅模式需要构建订阅者，这些订阅者本身的构建过程会增加额外的内存消耗。

4.发布订阅模式不适合嵌套过多，这和我们使用 vue 组件通信之间的嵌套是差不多的概念，如果组件嵌套太深，就不适合一层一层的传递通信了，因此发布订阅模式如果有多个订阅者和多个发布者嵌套太多的时候，也会让程序调试变得复杂。

谢谢观看！

我是星星课堂老师：周小周