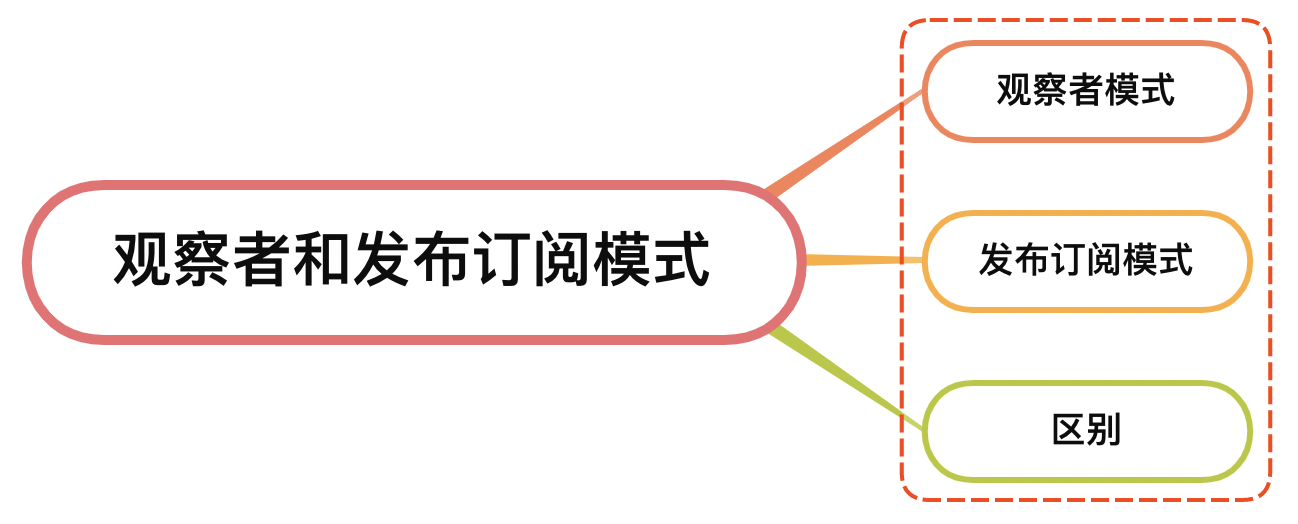
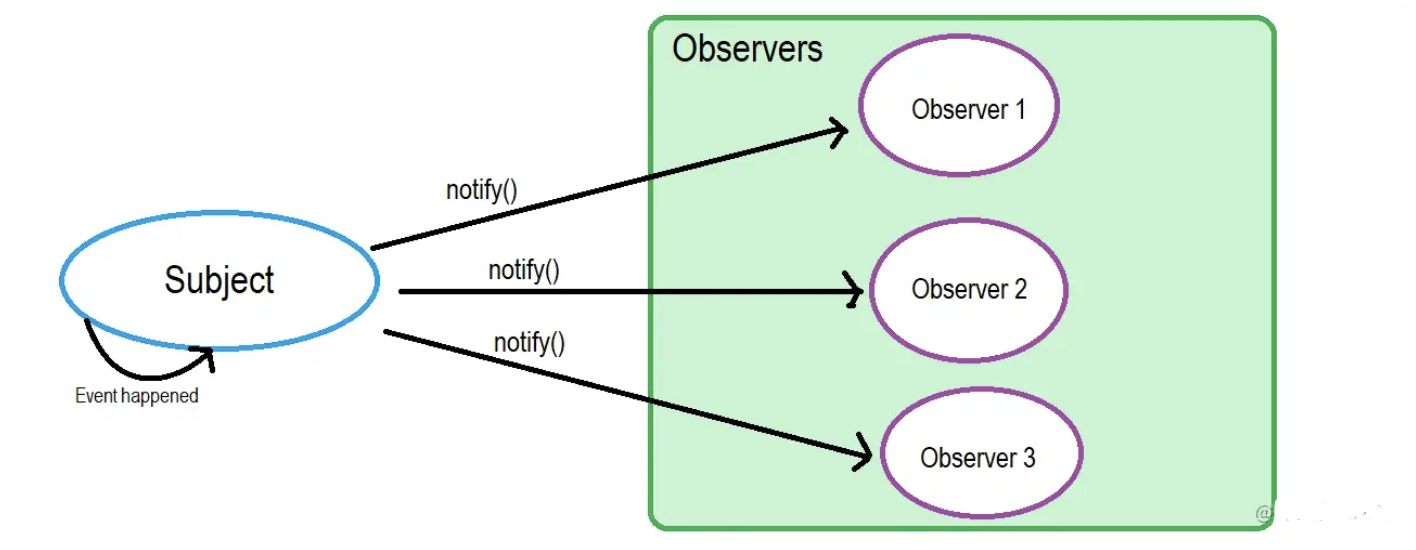
# 面试官：说说你对发布订阅、观察者模式的理解？区别？



## 一、观察者模式

观察者模式定义了对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都将得到通知，并自动更新

观察者模式属于行为型模式，行为型模式关注的是对象之间的通讯，观察者模式就是观察者和被观察者之间的通讯



例如生活中，我们可以用报纸期刊的订阅来形象的说明，当你订阅了一份报纸，每天都会有一份最新的报纸送到你手上，有多少人订阅报纸，报社就会发多少份报纸

报社和订报纸的客户就形成了一对多的依赖关系

实现代码如下：

被观察者模式

class Subject {  
  
 constructor() {  
 this.observerList = [];  
 }  
  
 addObserver(observer) {  
 this.observerList.push(observer);  
 }  
  
 removeObserver(observer) {  
 const index = this.observerList.findIndex(o => o.name === observer.name);  
 this.observerList.splice(index, 1);  
 }  
  
 notifyObservers(message) {  
 const observers = this.observeList;  
 observers.forEach(observer => observer.notified(message));  
 }  
  
}

观察者：

class Observer {  
  
 constructor(name, subject) {  
 this.name = name;  
 if (subject) {  
 subject.addObserver(this);  
 }  
 }  
  
 notified(message) {  
 console.log(this.name, 'got message', message);  
 }  
}

使用代码如下：

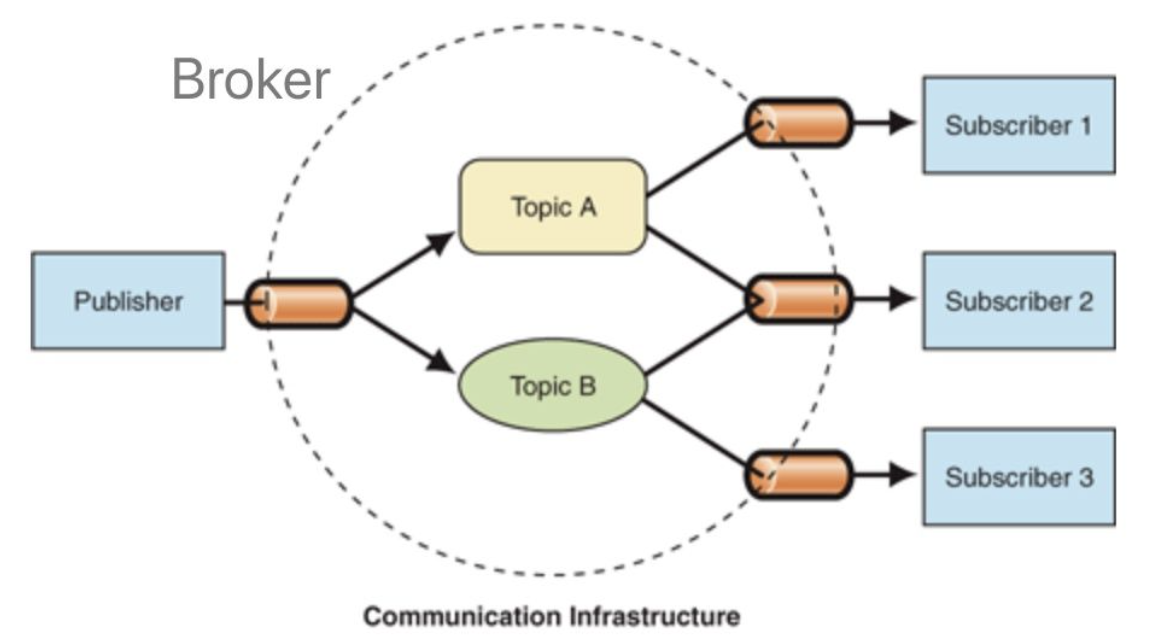
const subject = new Subject();  
const observerA = new Observer('observerA', subject);  
const observerB = new Observer('observerB');  
subject.addObserver(observerB);  
subject.notifyObservers('Hello from subject');  
subject.removeObserver(observerA);  
subject.notifyObservers('Hello again');

上述代码中，观察者主动申请加入被观察者的列表，被观察者主动将观察者加入列表

## 二、发布订阅模式

发布-订阅是一种消息范式，消息的发送者（称为发布者）不会将消息直接发送给特定的接收者（称为订阅者）。而是将发布的消息分为不同的类别，无需了解哪些订阅者（如果有的话）可能存在

同样的，订阅者可以表达对一个或多个类别的兴趣，只接收感兴趣的消息，无需了解哪些发布者存在



实现代码如下：

class PubSub {  
 constructor() {  
 this.messages = {};  
 this.listeners = {};  
 }  
 // 添加发布者  
 publish(type, content) {  
 const existContent = this.messages[type];  
 if (!existContent) {  
 this.messages[type] = [];  
 }  
 this.messages[type].push(content);  
 }  
 // 添加订阅者  
 subscribe(type, cb) {  
 const existListener = this.listeners[type];  
 if (!existListener) {  
 this.listeners[type] = [];  
 }  
 this.listeners[type].push(cb);  
 }  
 // 通知  
 notify(type) {  
 const messages = this.messages[type];  
 const subscribers = this.listeners[type] || [];  
 subscribers.forEach((cb, index) => cb(messages[index]));  
 }  
}

发布者代码如下：

class Publisher {  
 constructor(name, context) {  
 this.name = name;  
 this.context = context;  
 }  
 publish(type, content) {  
 this.context.publish(type, content);  
 }  
}

订阅者代码如下：

class Subscriber {  
 constructor(name, context) {  
 this.name = name;  
 this.context = context;  
 }  
 subscribe(type, cb) {  
 this.context.subscribe(type, cb);  
 }  
}

使用代码如下：

const TYPE\_A = 'music';  
const TYPE\_B = 'movie';  
const TYPE\_C = 'novel';  
  
const pubsub = new PubSub();  
  
const publisherA = new Publisher('publisherA', pubsub);  
publisherA.publish(TYPE\_A, 'we are young');  
publisherA.publish(TYPE\_B, 'the silicon valley');  
const publisherB = new Publisher('publisherB', pubsub);  
publisherB.publish(TYPE\_A, 'stronger');  
const publisherC = new Publisher('publisherC', pubsub);  
publisherC.publish(TYPE\_C, 'a brief history of time');  
  
const subscriberA = new Subscriber('subscriberA', pubsub);  
subscriberA.subscribe(TYPE\_A, res => {  
 console.log('subscriberA received', res)  
});  
const subscriberB = new Subscriber('subscriberB', pubsub);  
subscriberB.subscribe(TYPE\_C, res => {  
 console.log('subscriberB received', res)  
});  
const subscriberC = new Subscriber('subscriberC', pubsub);  
subscriberC.subscribe(TYPE\_B, res => {  
 console.log('subscriberC received', res)  
});  
  
pubsub.notify(TYPE\_A);  
pubsub.notify(TYPE\_B);  
pubsub.notify(TYPE\_C);

上述代码，发布者和订阅者需要通过发布订阅中心进行关联，发布者的发布动作和订阅者的订阅动作相互独立，无需关注对方，消息派发由发布订阅中心负责

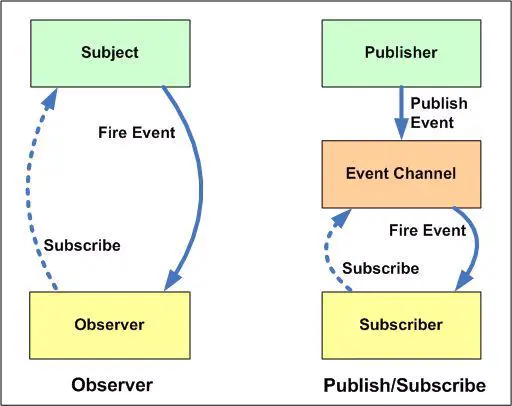
## 三、区别

两种设计模式思路是一样的，举个生活例子：

* 观察者模式：某公司给自己员工发月饼发粽子，是由公司的行政部门发送的，这件事不适合交给第三方，原因是“公司”和“员工”是一个整体
* 发布-订阅模式：某公司要给其他人发各种快递，因为“公司”和“其他人”是独立的，其唯一的桥梁是“快递”，所以这件事适合交给第三方快递公司解决

上述过程中，如果公司自己去管理快递的配送，那公司就会变成一个快递公司，业务繁杂难以管理，影响公司自身的主营业务，因此使用何种模式需要考虑什么情况两者是需要耦合的

两者区别如下图：



* 在观察者模式中，观察者是知道Subject的，Subject一直保持对观察者进行记录。然而，在发布订阅模式中，发布者和订阅者不知道对方的存在。它们只有通过消息代理进行通信。
* 在发布订阅模式中，组件是松散耦合的，正好和观察者模式相反。
* 观察者模式大多数时候是同步的，比如当事件触发，Subject就会去调用观察者的方法。而发布-订阅模式大多数时候是异步的（使用消息队列）

## 参考文献

* https://zh.wikipedia.org/zh-hans/%E8%A7%82%E5%AF%9F%E8%80%85%E6%A8%A1%E5%BC%8F
* https://zh.wikipedia.org/wiki/%E5%8F%91%E5%B8%83/%E8%AE%A2%E9%98%85
* https://www.cnblogs.com/onepixel/p/10806891.html
* https://juejin.cn/post/6978728619782701087