资源网站推荐：

1

汤姆大叔的播客，关于设计模式的一些文章（canvas第一天资料里有链接）

2

---

this指向问题：

观察者模式中涉及到this的问题，也就是obj中fn函数里存在this，addListener的时候，访问的this是window，需要包一层function，通过构造函数的方式调用

```
var obj = {
  fn: function() {
    // console.log('fn 中的方法')
    console.log( this );
  }
};

publisher.addListener( obj.fn );
publisher.addListener( function() {
  // 涉及到 this 的问题，使用这种方式
  obj.fn();
} );
// publisher.addListener( obj.fn );
publisher.trigger();
```

- **面向对象中的一些设计模式（最佳实践）**
  - 工厂模式
  - 单例模式
  - 观察者模式
  -

设计模式简介

设计模式（Design pattern）代表了最佳的实践，通常被有经验的面向对象的软件开发人员所采用。设计模式是软件开发人员在软件开发过程中面临的一般问题的解决方案。这些解决方案是众多软件开发人员经过相当长的一段时间的试验和错误总结出来的。

设计模式是一套被反复使用的、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了重用代码、让代码更容易被他人理解、保证代码可靠性。毫无疑问，设计模式于己于他人于系统都是多赢的，设计模式使代码编制真正工程化，设计模式是软件工程的基石，如同大厦的一块块砖石一样。项目中合理地运用设计模式可以完美地解决很多问题，每种模式在现实中都有相应的原理来与之对应，每种模式都描述了一个在我们周围不断重复发生的问题，以及该问题的核心解决方案，这也是设计模式能被广泛应用的原因。

严格模式中，函数调用的话，内部this指向undefined，而不是window

单例模式：对于一个构造函数（类）在整个应用程序运行期间只能有一个实例对象（只能创建同一个）

```
/*var Person = function() {
  this.name = '';
};

var p = new Person();
var p2 = new Person();*/
```

单例模式代码：

```
// Fly.Game = Game;
// 通过 单例模式 保证只有一个实例对象

// 实例（对象）
var instance = null;
Fly.createGame = function() {
  // 1 判断实例对象是否为空
  if( instance === null ) {
    // 创建一个 实例对象 赋值给 instance
    instance = new Game();
  }

  return instance;
};
)( Fly );
```

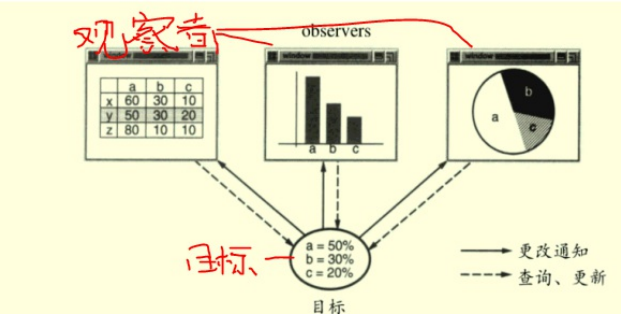
```
// 不管 createGame 方法调用多少次，获取到的结果都是同一个对象！
// 这样，就能够保证 整个程序运行期间值有一个实例对象！
var g = Fly.createGame();
var g1 = Fly.createGame();

console.log( g === g1 );
```

观察者模式：重要

1、什么是观察者模式

> 它定义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 都得到通知并被自动更新。

- 观察者模式又叫发布订阅模式（Publish/Subscribe）
  - publisher：发布者
  - subscriber：订阅者



3、观察者模式的说明

- 1 目标对象 与 观察者对象不知道对方的存在
- 2 各个观察者对象之间互不相识
- 3 观察者对象的数量没有限制
- 4 观察者 依赖于 目标对象，目标状态的改变，会通知所有观察者
- 5 观察者 接受到 目标对象的通知后，做出相应的改变

```
var dog = {
  // 狗叫的方法
  bark: function() {
    console.log('狗在想：弄啥咧??');

    // 发布消息，通知所有的订阅者触发自己的行为
    this.trigger();
  },

  // 订阅者数组，用于存储每一个订阅者的行为（方法）
  listeners: [],

  // 添加订阅者：
  addListener: function( fn ) {
    this.listeners.push( fn );
  },

  // 发布消息：
  trigger: function() {
    for(var i = 0 ; i < this.listeners.length; i++) {
      this.listeners[i]();
    }
  }
};
```

```
initRoles: function( imgList ) {
  var ctx = this.ctx, i, that = this;
```

```
// 添加小鸟碰撞的订阅：

// bind 方法也可以实现功能
this.hero.addLisener( this.gameOver.bind(this) );

/*this.hero.addLisener( function() {
  that.gameOver();
} );*/

// 错误的演示：
// this.hero.addLisener( this.gameOver );
```

看视频 this