js中最难的就是对象，而所谓的javascript面向对象，其实并不是真正的面向对象，因为js中没有类

什么是js的封装?
我们需要将一些属性和方法隐藏起来，所以我们将这些属性和方法进行封装。然后通过外部的一个特定的接口（公有的方法）调用

涉及知识：公有属性、公有方法、私有属性、私有方法

封装的优点：

1、良好的封装可以减少耦合性（了解高内聚低耦合）

高内聚低耦合参考：https://blog.csdn.net/fangkang7/article/details/82684819

2、类的内部结构可以自由修改

3、可以对成员进行精确的控制

4、隐藏信息，实现细节

```
// 构造函数封装
function Person(name,age,sex){
 this.name = name;//公有变量
 var age = age;//私有变量
 var sex = sex;//私有变量

 //公有方法
 this.show = function(){
 console.log('age:'+age+',sex:'+sex)
  }
}
```

```
var person = new Person('songxin',18,'女')
console.log(person);
console.log(person.name);//songxin

console.log(person.age);//undefined 私有属性,在外部不能访问
console.log(person.sex);//undefined 私有属性,在外部不能访问

person.show();//age:18,sex:女 通过公共接口(公共方法)调用
```

判断对象的所属关系

```
js中的安全模式:instanceof 判断是否由当前对象创建
instanceof判断结果为Boolean,属于则为true 不属于为false
```

封装继承

继承的意义：继承其实就是当多个方法存在相同的属性和方法时，就把这些相同的属性和方法提取到一个公共的方法中，通过原型prototype继承该方法，当然你也可以使用apply()、call()方法去继承方法中的属性和方法

```
// 定义一个人 Person将人所有的通用的属性放在了一起
function Person(name,age,sex){
 // 公用属性
 this.name = name;
 this.age = age;
 this.sex = sex;
 // 公用的方法
 this.show = function(){
 console.log('age:'+this.age+',sex:'+this.sex);
  }
}

// 方法一:通过apply和call方法继承
// 定义一个学生
// 构造函数中的this指代实例化对象
function Student(name,age,sex,score){
  // this.name = name;
```

```
 // this.age = age;
 // this.sex = sex;
 this.score = score;//this代表实例化对象

 // apply(obj,arr)切换对象 arr为参数
 Person.apply(this,[name,age,sex])//this代表实例化对象
}

var student = new Student('宋鑫',18,'女',80)
console.log(student);//{score: 80, name: "宋鑫", age: 18, sex: "女", show: f}
console.log(student.name)
student.show()

// 定义一个工人
function Worker(name,age,sex,job){
 // this.name = name;
 // this.age = age;
 // this.sex = sex;
 this.job = job;

 // call(obj,arg1,arg2,arg3...)切换对象,参数需要单个传入
 Person.call(this,name,age,sex)

}

var worker = new Worker('瑶瑶姐',30,'女','HTML5讲师')
console.log(worker)
console.log(worker.name)
worker.show()


// 方式二:通过prototype继承(推荐使用)
Teacher.prototype = new Person('静静姐',18,'女')

// 将构造函数蛇者为Teacher
// Teacher.prototype.constructor = Teacher;

function Teacher(type){
 this.type = type;
}
```

```
58
59
60 var teacher = new Teacher('班主任')
61 console.log(teacher)
62 console.log(teacher.name)
63 console.log(Teacher.prototype)
```