**介绍**

原型模式（prototype）是指用原型实例指向创建对象的种类，并且通过拷贝这些原型创建新的对象。

**正文**

对于原型模式，我们可以利用JavaScript特有的原型继承特性去创建对象的方式，也就是创建的一个对象作为另外一个对象的prototype属性值。原型对象本身就是有效地利用了每个构造器创建的对象，例如，如果一个构造函数的原型包含了一个name属性（见后面的例子），那通过这个构造函数创建的对象都会有这个属性。

在现有的文献里查看原型模式的定义，没有针对JavaScript的，你可能发现很多讲解的都是关于类的，但是现实情况是基于原型继承的JavaScript完全避免了类（class）的概念。我们只是简单从现有的对象进行拷贝来创建对象。

真正的原型继承是作为最新版的ECMAScript5标准提出的，使用Object.create方法来创建这样的对象，该方法创建指定的对象，其对象的prototype有指定的对象（也就是该方法传进的第一个参数对象），也可以包含其他可选的指定属性。例如Object.create(prototype, optionalDescriptorObjects)，下面的例子里也可以看到这个用法：

// 因为不是构造函数，所以不用大写  
var someCar = {  
 drive: function () { },  
 name: '马自达 3'  
};  
  
// 使用Object.create创建一个新车x  
var anotherCar = Object.create(someCar);  
anotherCar.name = '丰田佳美';

Object.create运行你直接从其它对象继承过来，使用该方法的第二个参数，你可以初始化额外的其它属性。例如：

var vehicle = {  
 getModel: function () {  
 console.log('车辆的模具是：' + this.model);  
 }  
};  
  
var car = Object.create(vehicle, {  
 'id': {  
 value: MY\_GLOBAL.nextId(),  
 enumerable: true // 默认writable:false, configurable:false  
 },  
 'model': {  
 value: '福特',  
 enumerable: true  
 }  
});

这里，可以在Object.create的第二个参数里使用对象字面量传入要初始化的额外属性，其语法与Object.defineProperties或Object.defineProperty方法类型。它允许您设定属性的特性，例如enumerable, writable 或 configurable。

如果你希望自己去实现原型模式，而不直接使用Object.create 。你可以使用像下面这样的代码为上面的例子来实现：

var vehiclePrototype = {  
 init: function (carModel) {  
 this.model = carModel;  
 },  
 getModel: function () {  
 console.log('车辆模具是：' + this.model);  
 }  
};  
  
  
function vehicle(model) {  
 function F() { };  
 F.prototype = vehiclePrototype;  
  
 var f = new F();  
  
 f.init(model);  
 return f;  
}  
  
var car = vehicle('福特Escort');  
car.getModel();

**总结**

原型模式在JavaScript里的使用简直是无处不在，其它很多模式有很多也是基于prototype的，就不多说了，这里大家要注意的依然是浅拷贝和深拷贝的问题，免得出现引用问题。