**原型链（prototype chaining）**

prototype 对象的任何属性和方法都被传递给那个类的所有实例。原型链利用这种功能来实现继承机制。

function ClassA() {

}

ClassA.prototype.color = "blue";

ClassA.prototype.sayColor = function () {

alert(this.color);

};

function ClassB() {

}

ClassB.prototype = new ClassA();

ClassB.prototype.name = "";

ClassB.prototype.sayName = function () {

alert(this.name);

};

**注意：**调用 ClassA 的构造函数，没有给它传递参数。这在原型链中是标准做法。要确保构造函数没有任何参数。

在原型链中，instanceof 运算符的运行方式也很独特。对 ClassB 的所有实例，instanceof 为 ClassA 和 ClassB 都返回 true。例如：

var objB = new ClassB();

alert(objB instanceof ClassA); //输出 "true"

alert(objB instanceof ClassB); //输出 "true"

原型链的弊端是不支持多重继承。记住，原型链会用另一类型的对象重写类的 prototype 属性。