ISSN 1009-3044
Computer Knowledge and Technology 电脑知识与技术
Vol.13, No16, June 2017

E-mail: xsjl@dnzs.net.cn
http://www.dnzs.net.cn
Tel:+86-551-65690963 65690964

JavaScript的面向对象特性分析

刘莲花,陈瑛

(广州商学院,广东 广州 511363)

摘要:传统面向对象的设计模式强调基于类的概念,而JavaScript不使用类。这让大多数开发者们在学习JavaScript的面向对象编程时候感到困惑。该文基于面向对象编程思想的本质,将JavaScript面向对象特性与传统面向对象语言进行比较分析,深入讨论了JavaScript的面向对象编程的特性。同时,用举例的方法阐述JavaScript是如何实现面向对象特性的。通过该文的分析,有利于大家对JavaScript面向对象特性的理解。

关键词:JavaScript;面向对象;原型对象;函数

中图分类号:TP312 **文献标识码**:A **文章编号**:1009-3044(2017)16-0063-03

Object-Oriented Features Analysis of JavaScript

LIU Lian-hua, CHEN Ying

(Guangzhou College of Commerce, Guangzhou 511363, China)

Abstract: Traditional object-oriented design patterns emphasize the concept of classes, while JavaScript does not use classes. This makes the majority of developers in learning JavaScript object-oriented programming confused. Based on the essence of object-oriented programming, this paper makes a comparative analysis of the object-oriented features of JavaScript with the traditional object-oriented language, and deeply discusses the characteristics of object-oriented programming of JavaScript. At the same time, some examples are given to illustrate how JavaScript implements the object oriented features. Through the analysis of this paper, it is helpful for the understanding of JavaScript object-oriented features.

Key words: JavaScript; object-oriented; prototype object; function

是否精通JavaScript现已成为招聘Web开发人员的决定性因素。如果Web开发人员在面试时被问到这样的问题:"JavaScript是一种面向对象语言吗?如果是,JavaScript中的继承关系是如何实现的呢?[1][6]"在现实中,即使是做了很多年JavaScript开发的程序员对这个问题也说不清道不明。很多人能熟练地使用各种框架,但其代码组织却不是很理想,这是由于其对JavaScript的原生语言特性如闭包、函数式编程、原型链等理解不够深入。本人在文献检索时发现,涉及JavaScript面向对象特性的主题文献不到10篇,其中黄华林等人提出"JavaScript是基于对象的语言,而非完全面向对象的。故JavaScript不可能实现面向对象语言的所有特性"[3]的观点。这是对JavaScript面向对象特性的误解。在有关JavaScript的著作中,题名出现"面向对象"字样的不超过10本,很多关于JavaScript的经典著作也只是用很小的篇幅来简单阐述它的面向对象特性。对JavaScript来说,大家关心的只是它的各种跨平台应用,但不能因为JavaScript中没有用到类,或没有接口的概念,就认为它不是完全面向对象的。恰恰相反,正是由于JavaScript的面向对象特性,它才能如此的灵活,应用的如此广泛。所以要真正理解JavaScript的面向对象特性,应该基于面向对象编程思想的本质,结合JavaScript自身的特点,从函数式编程及原型对象的角度来分析它的面向对象的特性。

本文从三个方面来展开分析讨论,一、面向对象编程思想的本质,二、JavaScript与传统基于类的面向对象语言的比较分析,三、JavaScript面向对象实现的举例说明。

1 面向对象编程思想的本质

面向对象编程思想的本质体现在——面向对象编程语言必须包含这些特性:对象、属性(变量)、行为(方法)、类、封装、继承、多态等。面向对象编程思想的重点在"对象"上,对象实质上是"客观事物"在程序设计语言中的表现形式,客观事物可以是某个客观存在的人或物,或者某些抽象的概念。事物的静态特征称之为属性,而事物的动态行为或动作称之为方法。将具有相同属性和行为的一组对象的集合抽象成为类的概念,可以基于同一个类实例化许多不同的对象,类是模板,而对象是在这些模板的基础上被创建出来的实体。封装的本质是为了实现信息的隐藏,就是把对象的属性和行为结合成一个独立的整体,这样做可以将对象的设计与使用分开,使用对象时不用去关心其内部的信息,只需通过传递消息来访问对象。继承是指在已经存在的类的基础上,构造出新的类,它很好地实现了代码复用。多态是指不同对象通过调用名字相同的方法来实

收稿日期:2017-05-10
基金项目:广东省创新强校工程项目(JXTD201601);广州商学院质量工程项目(ZLGC2015004)
作者简介:刘莲花,女,讲师,研究方向为计算机程序设计教学、机器视觉、模式识别、图像处理。

现各自行为的能力,这符合现实生活的习惯和要求,而且可以使程序更加简洁和一致。

面向对象是一种不依赖于某种语言而存在的编程思想。Java通过关键词class和extends很好地实现了面向对象编程,但并不是说这一思想只能通过这种方式来实现,一门编程语言可以借助其自身特点通过合适的某种方式来实现面向对象编程思想。所以,不能先入为主地认为基于类来实现面向对象的语言才是面向对象语言,而像JavaScript这样基于原型对象来实现面向对象的则不是面向对象语言。实际上,JavaScript虽然没有类的概念,没有class和extends关键词,但通过原型对象,它也实现了对象创建、封装、继承、多态等面向对象的特性。下面就来讨论基于类的面向对象和JavaScript基于原型对象的面向对象的差别。

2 JavaScript与传统基于类的面向对象的比较分析

在基于类的面向对象方式中,通过类来创建多个同属性和行为的对象。而ECMAScirpt中没有类的概念,其中的对象与基于类的语言中的对象有所不同,ECMA-262将对象定义为:"无序属性的集合,其属性可以包含基本值、对象或者函数[4]"。也就是说,JavaScript中的对象是一组"名-值"对,其中的值可以是数据,也可以是函数,当值是数据(变量)时,表示的是对象的属性,当值为函数时,表示的是对象的行为(方法)。这种基于原型对象的面向对象方式,对象是通过构造器函数利用原型对象构造出来的。以生产制造一辆机动车为例来说明这两种构造对象方式的差异。机械工程师设计出机动车的构造图纸,图纸详细规定了制作工艺及组装流程,工人依照图纸来生产制造。本例中的图纸就好比是基于类的面向对象语言中的类,而某辆车就是按照这个类制造出来的对象;而另一面,工人和机器利用各种零部件如发动机、车轮等将机动车组装制造出来,这里的零部件就如同是面向对象中的属性(可以是各种原生原型或对象),而机器相当于是构造器函数。

回到客观世界中,任意一个对象的产生都是通过其他已存在实物对象构造的结果,而抽象的"图纸"是不能产生"车子"的,也就是说,类是一个抽象概念而并非实体,而对象的构造是一个实体的产生;按照一切事物皆对象这个最基本的面向对象的法则来看,类本身并不是一个对象,而原型方式中的构造器和原型本身则是其他对象通过原型方式构造出来的对象。这更符合人们看待客观世界的方式,更能彻底地体现面向对象编程的思想。再者,在基于类的面向对象语言中,对象的属性值由每一个对象实例所独有,而对象的行为方法却统一在类内声明,由类拥有,在继承时,子类对象只能继承父类对象的结构和方法。而在基于原型的面向对象语言中,对象的行为、属性都属于对象本身,并且能够一起被继承,这也更贴近客观实际。

所以面向对象的编程思想除了能通过类来实现外,也可以通过其他的方式来实现,其中JavaScript就是这样的一种基于原型对象的面向对象语言,在接受了这一观点后,下面我们就举例来说明JavaScript是如何通过原型对象来实现面向对象的各种特性。

3 基于原型对象实现面向对象的举例

3.1 对象的创建

对象是面向对象编程的核心,比如:要定义一个机动车类,名字叫Vehicle,有两个属性:speed和wheelnum,另外有两个方法:一个show()方法,用于显示机动车当前的信息,一个speedup()方法,用于加速,那么用Java语言可以如下来实现:

```
public class Vehicle {
private int speed    //速度
private int wheelnum;  //轮子数
public Vehicle(int speed, int wheelnum){
this.speed=speed;
this.wheelnum=wheelnum;}
public int  getSpeed(){
return speed; }
public void setSpeed(int speed){
this.speed=speed; }
public int getWheelnum(){
return Wheelnum; }
public void show(){
System.out.println("My vehicle has " + getWheelnum () + " wheels. The speed is" +getSpeed()); }
public int speedup(int s){
speed+=s;}
public static void main(String[] args){
Vehicle V1 = new Vehicle(20,2);
V1.setSpeed(5)
V1.show();
V1.speedup(10)
V1.show();}}
```

JavaScript没有用于定义类的关键词,甚至没有类的概念,但其可通过定义原型对象来实现上述的类[1][03][5],实现代码如下:

```
function Vehicle(speed, wheelnum) {
this.speed=speed;
this.wheelnum=wheelnum;
this.show = function() {
alert("My vehicle has " + this.wheelnum + " wheels. The speed is" +this.speed); };
this.speedup=function(s){
this.speed+=s;}}
```

上述代码通过函数式编程的方式创建了一个原型对象Vehicle,在形式上和Java中类的定义非常的相似,有一点不同就在于关键词function。在这个原型对象的基础上,如果要创建一个新对象,可以通过new操作符来实现,具体代码如下:

```
var motobike1 = new Vehicle(0, 2);
motobike1.speedup(10);
motobike1.show(); //My vehicle has 2 wheels.The speed is 10
```

3.2 封装的实现

封装是面向对象编程中的重要概念之一。下面我们通过一个具体的例子来讨论JavaScript是如何实现封装特性的。

```
function Vehicle(speed,wheelnum){
var _speed = 10;    // 私有属性
var _wheelnum=2;
this.speed = speed;    // 公有属性
this.wheelnum=2;
function setSpeed(s1)  // 私有方法
{_speed = s1;
```

```
this.speed=_speed;}
function setwheelnum(num)
{_wheelnum = num;
this.wheelnum = _wheelnum;}
this.show = function()  // 公有方法
{alert("My vehicle has " + this.wheelnum + " wheels. The speed is" +this.speed); }}
// 通过原型(prototype)增加公有方法
Vehicle.prototype.speedup = function(s)
{ this.speed+=s; }
//创建对象并调用公有方法
var motobike2 = new Vehicle(5,2);
motobike2.show();    //My vehicle has 2 wheels.The speed is 5
motobike2.speedup(10);
motobike2.show();  //My vehicle has 2 wheels.The speed is 15
```

在原型对象内部定义的属性和方法，其作用域只在原型内部，外部不能访问，这样的属性就是私有属性，方法就是私有方法。而在原型对象内部定义带有this修饰的属性和方法，或是通过原型对象的原生原型定义的属性和方法，其作用域不仅限于原型对象内部，那么这些属性和方法将成为公有属性和方法。在JavaScript中，对象的属性和方法可以动态地添加，正是因为这个，其要定义静态属性和方法是很容易的事情。

3.3 继承的实现

JavaScript中没有实现继承的关键词extends，也没有严格地定义抽象类，但JavaScript通过语言本身的特性实现了继承[6]。它支持多重继承，而且实现继承的方式也有很多种，如：对象冒充、通过call或apply、原型链及多种混合的方式，下面通过原型链的方式来实现继承，代码如下：

```
// 定义父级原型对象
function Vehicle(speed, wheelnum) {
this.speed=speed;
this.wheelnum=wheelnum;
this.show = function() {
alert("My vehicle has " + this.wheelnum + " wheels. The speed is" +this.speed);
};}
Vehicle.prototype.speedup=function(s){
this.speed+=s;}
// 定义子级原型对象——两轮摩托车
function Motorcycle(speed)
{ this.Motoc=Vehicle;
this.Motoc(speed,2);//继承父级原型对象的属性
}// 继承
Motorcycle.prototype = new Vehicle();
// 覆盖父级的speedup方法
Motorcycle.prototype.speedup = function(s)
{alert("向内转动右手转把，就可以加速了！ ");
this.speed+=s; }
// 测试
var mc = new Motorcycle(0);
mc.show();  //My vehicle has 2 wheels.The speed is 0
mc.speedup(5);  // 向内转动右手转把，就可以加速了！
mc.show();  //My vehicle has 2 wheels.The speed is 5
```

这种方式是通过创建一个父级对象赋给子级的prototype属性，子级对象就拥有了父级对象中的属性和方法，即实现了继承[7]。每个对象实例都有一个内部属性prototype，通过这个属性跟踪其原型对象，子级的原型对象的内部属性prototype获得了父级的一个对象实例，继续跟踪到了父级的原型对象，如此就形成了一个原型链。

3.4 多态的实现

多态主要是通过方法的重载和覆盖来实现。重载是指名字相同的方法可以有多种不同的实现，主要通过参数的类型、参数的个数及参数的顺序的不同来识别。而JavaScript在函数定义时可以不指定函数参数的类型和个数，所以说它的本性就支持重载。而覆盖是指子类中可以定义与父类同签名的方法，子类对象调用这些方法时，从父类中继承过来的同名方法将被隐藏。

在上述实现继承的例子中，再定义一个car的子类(原型对象)，实现代码如下：

```
// 定义子级原型对象——四轮小汽车
function Car(speed)
{ Vehicle.call(this,speed,4); }
// 继承
Car.prototype = new Vehicle();
// 覆盖父级的speedup方法
Car.prototype.speedup = function(s)
{alert("右脚向下踩油门，就可以加速了！ ");
this.speed+=s; }
// 多态的实现测试
var V1=new Vehicle();
V1=new Motorcycle(0);
V1.speedup(5);  //2个轮子的摩托通过右手向内转动转把加速了5码！
V1.show();    //My vehicle has 2 wheels.The speed is 5
V1=new Car(0);
V1.speedup(20);  //4个轮子的小汽车通过右脚向下踩油门加速了20码！
V1.show();      //My vehicle has 4 wheels.The speed is 20
```

一个父级的对象实例能接受不同子级的对象实例，在调用同一个签名的方法时，它能根据当前接受的子级对象的不同而有不同的反应，这就是多态的魅力所在。

4 结束语

传统基于类的面向对象思维在一定程度上妨碍了大家对JavaScript面向对象特性的理解，本文对面向对象编程的本质进行了分析，通过比较分析及举例说明的研究方法深入讨论了JavaScript的面向对象特性。它的弱类型，简单易用性、解释性及跨平台性，使其在Web的应用开发中可以说是无处不在，但它的无类动态对象、原型链继承、闭包等特性[8]使其具有更高的灵活性。所以彻底地理解并应用JavaScript的面向对象特性是非常有意义和有必要的。

参考文献：

[1] Stoyan Stefanov, Kumar Chetan Sharma.JavaScript面向对象编程指南[M]. 2版.北京：人民邮电出版社,2015:4-6.

(下转第93页)

2 组件开发研究

组件开发研究主要介绍组件开发的策略和组件工程化需要解决的问题。

2.1 组件开发分析

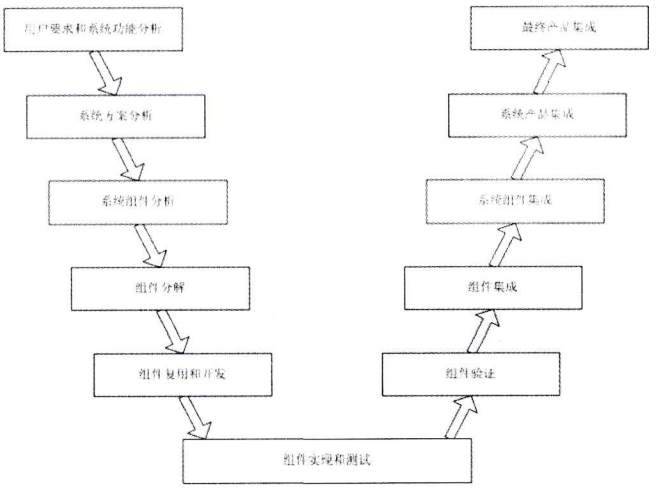

图9　组件开发策略

如图9所示，组件开发思路包括系统功能分析，系统方案分析，系统组件分析，组件分解，组件复用和开发，组件实现和测试等工作。组件开发对系统方案和软件开发提出了更高的要求。系统方案和组件复用与开发是实现支撑系统平台和支撑系统平台组件方案的关键。

2.2 组件工程化

在解决方案提及到组件开发独立于型号课题，但源于和应用于型号课题，并纳入系统工程及资产库(知识库)管理。组件工程化开发难度较大，需要解决如下几个问题：

1) 组件开发问题：组件化开发思路与原有开发思路存在本质的差异，需要研制团队转变开发思路和观念，注重需求整合与引导，产品模块整体架构，系统方案，系统平台化，组件化和标准化，组件裁减等。对系统方案和组件开发等人员等提出了更高的要求。

2) 组件资质问题：型号课题要求多样，组件开发需要满足多个型号课题要求，因此涉及组件工程化，定型及合法化等问题。

3) 组件应用问题：组件如何应用于多个型号课题项目，涉及组件应用方案策略，组件集成，组件测试及组件变更等。

3 组件应用

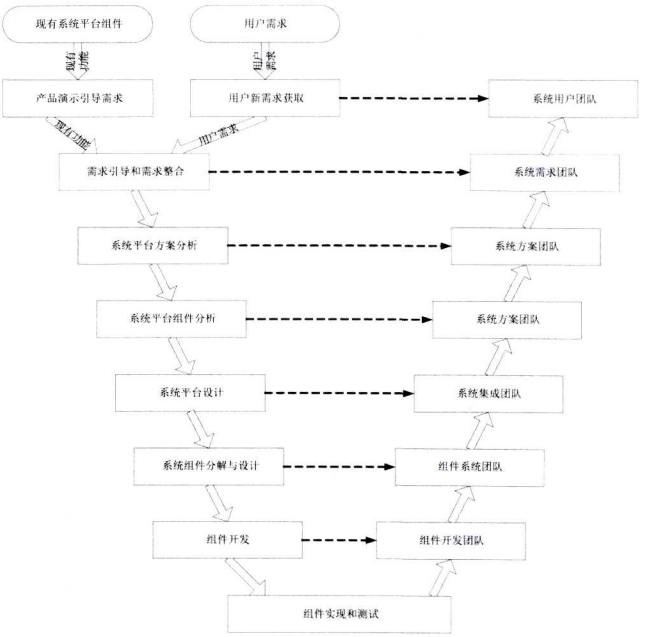

图10　组件应用模型

如图10所示，给出了组件应用思路。组件应用包括产品演示，依据用户需求构建系统方案，依据系统方案构建组件方案，组件的分解与集成，最后组件的集成与开发。系统方案和组件复用与开发是组件应用的关键。

4 结束语

软件开发的最佳方法是不进行任何开发。重用就是实现上述目标的一种方法[2]。基于组件的软件重用是产品重用的主要形式，软件组件技术是当前重用研究的焦点[1]。本文从构建系统平台的角度提出了系统平台组件的构建策略和思路，分析了组件应用及组件开发过程中，组件管理需要解决的问题等。本文研究了支撑系统平台解决方案，组件开发和应用模型。这将对航空领域同类型项目提供软件复用的基础，为航电系统型号项目软件提供可参考意义和借鉴价值。

参考文献：

[1] 周淳,邓中亮.嵌入式组件技术的研究与应用[J].现代电子技术,2009(6):50-52.

[2] 储赟.面向组件的软件重用技术[J].电脑知识与技术,2007(3):1584-1621.

(上接第65页)

[2] 黄华林,宋阳秋.JavaScript面向对象特性浅析与范例[J].安庆师范学院学报:自然科学版,2005(4):85-88.

[3] ECMAScript 官方标准. Standard ECMA-262[S]. http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[4] Zakas N C. JavaScript高级程序设计[M]. 3版.北京:人民邮电出版社,2015:147-161.

[5] Douglas Crockford. Private Members in JavaScript[EB/OL]. [2010-01-01].http://javascript.crockford.com/zh/private.html.

[6] John Resig. Simple JavaScript Inheritance[EB/OL].[2011-01-01].http://ejohn.org/blog/simple-javascript-inheritance/.

[7] Nicholas C.Zakas. JavaScript面向对象精要[M].北京:人民邮电出版社,2014:53-60.

[8] Douglas Crockford. JavaScript: The Wrrrld's Most Misunderstood Programming Language [EB/OL]. [2013-01-01].http://javascript.crockford.com/zh/javascript.html.