**JS对象创建的几种方式整理**

作者：小智a 字体：[[增加](javascript:turnbig()) [减小](javascript:turnsmall())] 类型：转载 时间：2017-02-28[我要评论](http://www.jb51.net/article/107012.htm#comments)

本文主要介绍了JS对象创建的几种方式，具有很好的参考价值，下面跟着小编一起来看下吧

最近一直在看JS高级程序设计这本书，有空来梳理一下几种创建对象的方式。话不多说，直接步入正题。

**第一种：Object构造函数创建**

var Person = new Object();

Person.name = 'Nike';

Person.age = 29;

这行代码创建了Object引用类型的一个新实例，然后把实例保存在变量Person中。

**第二种：使用对象字面量表示法**

var Person = {};//相当于var Person = new Object();

var Person = {

name:'Nike';

age:29;

}

对象字面量是对象定义的一种简写形式，目的在于简化创建包含大量属性的对象的过程。也就是说，第一种和第二种方式创建对象的方法其实都是一样的，只是写法上的区别不同

在介绍第三种的创建方法之前，我们应该要明白为什么还要用别的方法来创建对象，也就是第一种，第二种方法的缺点所在：**它们都是用了同一个接口创建很多对象，会产生大量的重复代码，就是如果你有100个对象，那你要输入100次很多相同的代码。**那我们有什么方法来避免过多的重复代码呢，就是把创建对象的过程封装在函数体内，通过函数的调用直接生成对象。

**第三种：使用工厂模式创建对象**

function createPerson(name,age,job){

var o = new Object();

o.name = name;

o.age = age;

o.job = job;

o.sayName = function(){

alert(this.name);

};

return o;

}

var person1 = createPerson('Nike',29,'teacher');

var person2 = createPerson('Arvin',20,'student');

在使用工厂模式创建对象的时候，我们都可以注意到，在createPerson函数中，返回的是一个对象。那么我们就无法判断返回的对象究竟是一个什么样的类型。于是就出现了第四种创建对象的模式。

**第四种:使用构造函数创建对象**

function Person(name,age,job){

this.name = name;

this.age = age;

this.job = job;

this.sayName = function(){

alert(this.name);

};

}

var person1 = new Person('Nike',29,'teacher');

var person2 = new Person('Arvin',20,'student');

**对比工厂模式，我们可以发现以下区别：**

1.没有显式地创建对象

2.直接将属性和方法赋给了this对象

3.没有return语句

4.终于可以识别的对象的类型。对于检测对象类型，我们应该使用instanceof操作符，我们来进行自主检测：

alert(person1 instanceof Object);//ture

alert(person1 instanceof Person);//ture

alert(person2 instanceof Object);//ture

alert(person2 instanceof Object);//ture

var b1=new Boolean(true);

b1 instanceof Boolean;//true

同时我们也应该明白，**按照惯例，构造函数始终要应该以一个大写字母开头，而非构造函数则应该以一个小写字母开头。**

那么构造函数确实挺好用的，但是它也有它的缺点：

就是每个方法都要在每个实例上重新创建一遍，方法指的就是我们在对象里面定义的函数。如果方法的数量很多，就会占用很多不必要的内存。于是出现了第五种创建对象的方法

**第五种：原型创建对象模式**

function Person(){}

Person.prototype.name = 'Nike';

Person.prototype.age = 20;

Person.prototype.job = 'teacher';

Person.prototype.sayName = function(){

alert(this.name);

};

var person1 = new Person();

person1.sayName();

使用原型创建对象的方式，可以让所有对象实例共享它所包含的属性和方法。

如果是使用原型创建对象模式，请看下面代码：

function Person(){}

Person.prototype.name = 'Nike';

Person.prototype.age = 20;

Person.prototype.jbo = 'teacher';

Person.prototype.sayName = function(){

alert(this.name);

};

var person1 = new Person();

var person2 = new Person();

person1.name ='Greg';

alert(person1.name); //'Greg' --来自实例

alert(person2.name); //'Nike' --来自原型

当为对象实例添加一个属性时，这个属性就会**屏蔽**原型对象中保存的同名属性。

这时候我们就可以使用构造函数模式与原型模式结合的方式，构造函数模式用于定义实例属性，而原型模式用于定义方法和共享的属性

**第六种：组合使用构造函数模式和原型模式**

function Person(name,age,job){

this.name =name;

this.age = age;

this.job = job;

}

Person.prototype = {

constructor:Person,

sayName: function(){

alert(this.name);

}

}

var person1 = new Person('Nike',20,'teacher');

以上就是本文的全部内容，希望本文的内容对大家的学习或者工作能带来一定的帮助，同时也希望多多支持脚本之家！