**js继承中关于对象字面量重写原型**

原创 2017年05月04日 14:21:12

* 213
* 0
* 0

function SuperType(){

this.property=true;

}

SuperType.prototype.getSuperValue=function(){

return this.property;

};

function SubType(){

this.subproperty=false;

}

//继承了SuperType

SubType.prototype=new SuperType();

//使用字面量添加新方法，会导致上一行代码无效

SubType.prototype={

getSubValue:function(){

return this.subproperty;

},

someOtherMethod:function(){

return false;

}

};

var instance=new SubType();

alert(instance.getSuperValue());//error！

[http://static.blog.csdn.net/images/save_snippets.png](javascript:;)

（2）

function SuperType(){

this.property=true;

}

SuperType.prototype.getSuperValue=function(){

return this.property;

};

function SubType(){

this.subproperty=false;

}

//继承了SuperType

SubType.prototype=new SuperType();

//添加新方法

SubType.prototype.getSubValue=function(){

return this.subproperty;

};

//重写超类型中的方法

SubType.prototype.getSuperValue=function(){

return false;

};

var instance=new SubType();

alert(instance.getSuperValue());//false

关于（1）（2）代码中为什么（1）中不能正常运行呢？

之前翻看了许多的资料都没有找到合理的解释。

今天偶然之间明白。

首先：创建对象的方法有哪些呢？

（1）Object（）构造函数；（2）使用对象字面量创建；（3）工厂模式；（4）原型模式；（5）构造函数模式；（6）组合使用构造函数的原型模式；（7）动态原型模式；

（8）寄生构造函数模式；（9）稳妥构造函数模式。

看（1）中代码可看出使用对象字面量添加新方法在new之后，但与此同时也创建了新对象。

因为每创建一个函数，就会创建它的prototype对象，这个对象也会自动获得constructor属性。而我们使用对象字面量，本质上完全重写了默认的prototype对象，因此

constructor属性也就变成了新对象的constructor属性。

说的可能不太清楚。

详细参考 js红皮书：p155和p165~166;