[JS中的prototype](http://www.cnblogs.com/yjf512/archive/2011/06/03/2071914.html)

2011-06-03 14:40 by 轩脉刃, 20894 阅读, 19 评论, [收藏](http://www.cnblogs.com/yjf512/archive/2011/06/03/2071914.html), [编辑](http://www.cnblogs.com/yjf512/admin/EditPosts.aspx?postid=2071914)

JS中的phototype是JS中比较难理解的一个部分

本文基于下面几个知识点:

1 原型法设计模式

在.Net中可以使用clone()来实现原型法

原型法的主要思想是，现在有1个类A,我想要创建一个类B,这个类是以A为原型的,并且能进行扩展。我们称B的原型为A。

2 javascript的方法可以分为三类：

a 类方法

b 对象方法

c 原型方法

例子：

[复制代码](javascript:void(0);)

function People(name)  
{  
this.name=name;  
//对象方法  
this.Introduce=function(){  
alert("My name is "+this.name);  
}  
}  
//类方法  
People.Run=function(){  
alert("I can run");  
}  
//原型方法  
People.prototype.IntroduceChinese=function(){  
alert("我的名字是"+this.name);  
}  
  
   
  
//测试  
  
var p1=new People("Windking");  
  
p1.Introduce();  
  
People.Run();  
  
p1.IntroduceChinese();

[复制代码](javascript:void(0);)

3 obj1.func.call(obj)方法

意思是将obj看成obj1,调用func方法

好了，下面一个一个问题解决：

prototype是什么含义？

javascript中的每个对象都有prototype属性，Javascript中对象的prototype属性的解释是：返回对象类型原型的引用。

A.prototype = new B();

理解prototype不应把它和继承混淆。A的prototype为B的一个实例，可以理解A将B中的方法和属性全部克隆了一遍。A能使用B的方法和属性。这里强调的是克隆而不是继承。可以出现这种情况：A的prototype是B的实例，同时B的prototype也是A的实例。

先看一个实验的例子：

[复制代码](javascript:void(0);)

function baseClass()  
{  
  
this.showMsg = function()  
{  
     alert("baseClass::showMsg");     
}  
  
}  
  
function extendClass()  
{  
  
}  
  
extendClass.prototype = new baseClass();  
  
var instance = new extendClass();  
  
instance.showMsg(); // 显示baseClass::showMsg

[复制代码](javascript:void(0);)

我们首先定义了baseClass类，然后我们要定义extentClass，但是我们打算以baseClass的一个实例为原型，来克隆的extendClass也同时包含showMsg这个对象方法。

extendClass.prototype = new baseClass()就可以阅读为：extendClass是以baseClass的一个实例为原型克隆创建的。

那么就会有一个问题，如果extendClass中本身包含有一个与baseClass的方法同名的方法会怎么样？

下面是扩展实验2：

[复制代码](javascript:void(0);)

function baseClass()  
{  
    this.showMsg = function()  
    {  
        alert("baseClass::showMsg");     
    }  
}  
  
function extendClass()  
{  
    this.showMsg =function ()  
    {  
        alert("extendClass::showMsg");  
    }  
}  
  
extendClass.prototype = new baseClass();  
var instance = new extendClass();  
  
instance.showMsg();//显示extendClass::showMsg

[复制代码](javascript:void(0);)

实验证明：函数运行时会先去本体的函数中去找，如果找到则运行，找不到则去prototype中寻找函数。或者可以理解为prototype不会克隆同名函数。

那么又会有一个新的问题：

如果我想使用extendClass的一个实例instance调用baseClass的对象方法showMsg怎么办？

答案是可以使用call:

extendClass.prototype = new baseClass();  
var instance = new extendClass();  
  
  
var baseinstance = new baseClass();  
baseinstance.showMsg.call(instance);//显示baseClass::showMsg

这里的baseinstance.showMsg.call(instance);阅读为“将instance当做baseinstance来调用，调用它的对象方法showMsg”

好了，这里可能有人会问，为什么不用baseClass.showMsg.call(instance);

这就是对象方法和类方法的区别，我们想调用的是baseClass的对象方法

最后，下面这个代码如果理解清晰，那么这篇文章说的就已经理解了：

[复制代码](javascript:void(0);)

<script type="text/javascript">  
  
function baseClass()  
{  
    this.showMsg = function()  
    {  
        alert("baseClass::showMsg");     
    }  
     
    this.baseShowMsg = function()  
    {  
        alert("baseClass::baseShowMsg");  
    }  
}  
baseClass.showMsg = function()  
{  
    alert("baseClass::showMsg static");  
}  
  
function extendClass()  
{  
    this.showMsg =function ()  
    {  
        alert("extendClass::showMsg");  
    }  
}  
extendClass.showMsg = function()  
{  
    alert("extendClass::showMsg static")  
}  
  
extendClass.prototype = new baseClass();  
var instance = new extendClass();  
  
instance.showMsg(); //显示extendClass::showMsg  
instance.baseShowMsg(); //显示baseClass::baseShowMsg  
instance.showMsg(); //显示extendClass::showMsg  
  
baseClass.showMsg.call(instance);//显示baseClass::showMsg static  
  
var baseinstance = new baseClass();  
baseinstance.showMsg.call(instance);//显示baseClass::showMsg  
  
</script>

[复制代码](javascript:void(0);)

作者：轩脉刃(yjf512)  
出处：（<http://www.cnblogs.com/yjf512/>）  
版权声明：本文的版权归作者与博客园共有。欢迎转载阅读，转载时须注明本文的详细链接。