**只用prototype实现复杂“继承”**

现在JavaScript的继承基本上有 原型、构造、拷贝、实例 四种方式  
这四种方式各有优劣  
目前比较完备的扩展机制通常是通过实现以属性拷贝为主的混合继承来模拟出来的  
但是个人并不喜欢那种拷贝属性的机制，因为它会有副作用（例如永久不可恢复地覆盖了父类的属性），而且具有属性数量规模的线性复杂度（O(N)）  
而实际上，我认为以原型拷贝为主的混合继承拥有更高的效率和更加优美的语法形式，下面是一个初步的探索——  
  
   提示：您可以先修改部分代码再运行  
  
这种方式的好处是  
1) 不用copy属性  
2) 真正的类继承  
3) 自动调用基类构造函数  
4) 支持多继承  
5) 不产生副作用  
6) 扩展属性较多的复杂对象的时候不会有很大的时间开销

[[复制]](http://bbs.51js.com/viewthread.php?tid=76251&highlight=###)

**Code:**

<script>  
Function.prototype.createInstance = function()  
{  
        var p = this.getPrototype();  
        this.apply(p,arguments);  
        return p;  
}  
Function.prototype.getPrototype = function()  
{  
        var p = function(){};  
        p.prototype = this.prototype;  
        return new p;  
}  
Object.prototype.override = function()  
{  
    for(var i=0;i<arguments.length;i++)  
       for(var j in arguments[i])  
       {  
          this[j] = arguments[i][j];  
       }  
}  
\_getPrototypeType = function(obj)  
{  
        var p = function(){};  
        p.prototype = obj;  
        return p;  
}  
Function.prototype.Extends = function()  
{  
    var $args = Array.prototype.slice.apply(arguments);  
    var parent = $args[$args.length-1] || function(){};  
    parent = parent instanceof Function ? parent : \_getPrototypeType(parent);  
    var $p = this;  
    var ret = function(){  
        parent.apply(this, arguments);  
        $p.apply(this, arguments);  
    }  
    ret.prototype.override(parent.getPrototype(),$p.prototype);  
  
    var $par = parent;  
    ret.Extends = function()  
    {  
        return Function.prototype.Extends.apply(this, arguments);      
    }  
  
    if($args.length > 1)  
    {  
        ret = ret.Extends.apply(ret,$args.slice(0,-1));      
    }  
      
    return ret;  
}  
function test(){this.p=1;}  
function par(){this.p2=2;}  
function par2(){this.p3=3;}  
par2.prototype.f = 4;  
par.prototype.f2 = 5;  
var ret = test.Extends(par2,par);  
var o = new ret;  
alert(o.p+"\r"+o.p2+"\r"+o.p3);  
alert(o.f+"\r"+o.f2);  
</script>

  
   提示：您可以先修改部分代码再运行