**javascript的prototype继承**

今天测试了一下prototype继承 非常失望 决定以后再也不用prototype继承了 所以此帖事劝你不要用prototype的 而不是讨论如何用prototype的  
当然我发表如此偏激的言论 也是希望听到一些反对的声音   
  
基本的用法 把ClassA的一个实例赋值给ClassB ClassB就继承了ClassA的所有属性  
  
   提示：您可以先修改部分代码再运行  
  
从原型继承理论的角度去考虑 js的原型继承是引用原型 不是复制原型  
所以 修改原型会导致所有B的实例的变化  
  
   提示：您可以先修改部分代码再运行  
  
然而 子类对象的写操作只访问子类对象中成员 它们之间不会互相影响  
因此 写是写子类 读是读原型（如果子类中没有的话）  
  
   提示：您可以先修改部分代码再运行  
  
每个子类对象都执有同一个原型的引用 所以子类对象中的原型成员实际是同一个  
  
   提示：您可以先修改部分代码再运行  
  
构造子类时 原型的构造函数不会被执行  
  
   提示：您可以先修改部分代码再运行  
  
接下来是致命的，在子类对象中访问原型的成员对象：  
  
   提示：您可以先修改部分代码再运行  
  
所以 在prototype继承中 原型类中不能有成员对象！ 所有成员必须是值类型数据（string也可以）  
  
再加上prototype不支持多继承，还有什么理由用它么？  
instanceof ? 比起如此严重的问题instanceof算什么呢？

用prototype继承有执行效率高，不会浪费内存，为父类动态添置方法后子类中马上可见等的优点。  
  
我就非常喜欢用prototype继承。  
  
prototype继承是通过把子类的原型对象（prototype）设置成父类的一个实例来进行继承的。  
  
只简单的这样设置继承的确如楼主所说，有不少缺点。总的来说有四个缺点：  
  
　　缺点一：父类的构造函数不是像JAVA中那样在给子类进行实例化时执行的，而是在设置继承的时候执行的，并且只执行一次。这往往不是我们希望的，特别是父类的构造函数中有一些特殊操作的情况下。  
  
　　缺点二：由于父类的构造函数不是在子类进行实例化时执行，在父类的构造函数中设置的成员变量到了子类中就成了所有实例对象公有的公共变量。由于JavaScript中继承只发生在“获取”属性的值时，对于属性的值是String，Number和Boolean这些数据本身不能被修改的类型时没有什么影响。但是Array和Object类型就会有问题。  
　　缺点三：如果父类的构造函数需要参数，我们就没有办法了。  
  
　　缺点四：子类原本的原型对象被替换了，子类本身的constructor属性就没有了。在类的实例取它的constructor属性时，取得的是从父类中继承的constructor属性，从而constructor的值是父类而不是子类。  
  
  
我也曾经为了这四个缺点头疼过，于是对prototype继承进行改造。  
我试了几种方法，下面是我觉得最好的一种。我把它写成Function对象的一个方法，这样用的时候方便。方法如下：  
  
   提示：您可以先修改部分代码再运行

下面用这方法对楼主几个例子进行解决  
  
解决：构造子类时 原型的构造函数不会被执行  
  
   提示：您可以先修改部分代码再运行

解决：接下来是致命的，在子类对象中访问原型的成员对象：  
  
   提示：您可以先修改部分代码再运行