JavaScript 的继承方式

现有一个People 类，People 类有姓名，性别和说话的成员 ; 再实现一个Programmer 类,程序员也是人，所以让他继承People 类，同时他有自己的一个特性🡪写代码

怎么实现呢？

1. **方法一：使用 Object.create 原型继承**

//pepole 类

var pepole = {

name:'',

say:function(){

alert("hi, I am " + this.name +"!");

}

}

//programmer类：create 一个新的对象作为programmer 的原型

var programmer = Object.create(pepole);

programmer.coding = function(){

alert("I am a programmer, I love coding!");

}

//调用继承下来的方法行为

programmer.name='Alex Yu';

programmer.say();

//自身特有行为

programmer.coding();

function pepole(){  
 this.name ='';  
 this.say = function(){  
 alert("hi, I am " + this.name +"!");  
 };  
}

1. **方法二：使用 new 原型继承**  
   //programmer类：new 一个新的对象作为programmer 的原型来实现继承  
   var programmer = new pepole();   
   programmer.coding = function(){   
    alert("I am a programmer, I love coding!");  
   }  
     
   //调用继承下来的方法行为  
   programmer.name='Alex Yu';  
   programmer.say();  
   //自身特有行为  
   programmer.coding();
2. **方法三: Pseudoclassical  方式**  
   function pepole(){  
    this.name ='';  
    this.say = function(){  
    alert("hi, I am " + this.name +"!");  
    };  
   }  
     
   //programmer类：new 一个新的对象作为programmer 的原型来实现继承  
   function programmer(){  
    this.coding = function(){   
    alert("I am a programmer, I love coding!");  
   }  
   }   
     
   programmer.prototype = new pepole();  
     
     
   //调用继承下来的方法行为  
   var p = new programmer();  
   p.name='Alex Yu';  
   p.say();  
   //自身特有行为  
   p.coding();

Javascript 继承一般使用原型继承和pseudoclassical 模式继承两种。

以下方法一和方法二都属于原型继承方式。这两种模式都有的一个缺点，它们不会提供任何成员隐私。有时候成员隐私无关紧要，但有时候却很重要。在 《The Good Parts 》中，将其称之为函数继承模式。  
  
4.**方法四 :函数继承模式**  
   
      
var pepole= function(){  
 this.salary =100;  
 var that ={};  
 that.name ='';   
 that.say = function(){  
 alert("hi, I am " + this.name +"!");  
 };  
 return that;  
}  
  
//programmer类：new 一个新的对象作为programmer 的原型来实现继承  
function programmer(){  
 var that = new pepole()  
 that.coding = function(){   
 alert("I am a programmer, I love coding!");  
   
 }  
 return that;  
}   
  
   
//调用继承下来的方法行为  
var p = new programmer();  
p.name='Alex Yu';  
p.say();  
alert(p.salary);   
//自身特有行为  
p.coding();