Atitit.**实现继承的原理and方法...**

[1. 实现继承的问题 1](#_Toc9018)

[2. 继承方式与模型,,原型链继承与模版继承 2](#_Toc9236)

[3. 实现继承的几种方法的总结：1、构造继承法2、原型继承法3、实例继承法4、拷贝继承法 2](#_Toc388)

[4. 继承在编程中有两种说法，一个叫 inherit，另一个是 extend 2](#_Toc21356)

[5. Js的继承实现 2](#_Toc16504)

[5.1. 如何拷贝基类方法？采用prototype原型方式，通过冒充对象 2](#_Toc11450)

[5.2. 2.属性如何继承？ 3](#_Toc31410)

[5.3. 几种继承方式的比较  (构造继承 4](#_Toc4548)

[5.4. 原型继承 4](#_Toc24077)

[5.5. 实例继承 4](#_Toc18272)

[5.6. 拷贝继承 4](#_Toc22736)

[6. Java是通过隐式的组合来实现继承 5](#_Toc21623)

[6.1. 常用的apply ,call方法 5](#_Toc18700)

[7.](#_Toc3677)**[Ruby语言利用Mix-in实现装配的继承](#_Toc3677)** [5](#_Toc3677)

[8. 参考 5](#_Toc2091)

# ****实现继承的问题****

JavaScript中要实现继承，其实就是实现三层含义：  
1、子类的实例可以共享父类的方法；  
2、子类可以覆盖父类的方法或者扩展新的方法；  
3、子类和父类都是子类实例的“类型”。

要解决两个问题：1.如何拷贝基类方法？2.属性如何继承？

JavaScript中，并不直接从语法上支持继承，但是可以通过模拟的方法来实现继承，以下是关于实现继承的几种方法的总结：  
1、构造继承法  
2、原型继承法  
3、实例继承法  
4、拷贝继承法

# 继承方式与模型,,原型链继承与模版继承

Javascript 仍然是世界上唯一使用 Prototype 继承模型的主要语言。 这使得它没有设计先例可以参考。

Java的模版继承，却不知道另一种继承方式叫做“原型链继承”。

不然现在大多数人可能只知道Java的模版继承，却不知道另一种继承方式叫做“原型链继承”。

# 实现继承的几种方法的总结：1、构造继承法2、原型继承法3、实例继承法4、拷贝继承法

# 继承在编程中有两种说法，一个叫 inherit，另一个是 extend

。前者是严格意义上的继承，即存在父子关系，而后者仅仅是一个类****扩展****了另一个类的属性和方法

# ****Js的继承实现****

## 如何拷贝基类方法？采用prototype原型方式，通过冒充对象

1. 对于问题1，一般采用prototype原型方式，通过冒充对象。

|  |
| --- |
| /\*更改Member的prototype属性，以实现继承\*/ |

|  |  |
| --- | --- |
| 09 | Member.prototype=new User(); |

2.拷贝基类方法代码

|  |  |
| --- | --- |
| 1 | Member.prototype=new User(); |

　　此行代码的含义是new 一个User对象，将此引用传递给Member的原型，这样完成了基类方法的拷贝工作。（备注：此处代码一定要先执行，然后再添加新的方法）。

## 2.属性如何继承？

对于问题2，一般采用调用基类构造函数

|  |
| --- |
| /\*初始化基类参数\*/ |

|  |  |
| --- | --- |
| 06 | User.call(this,userid,uname,uage,uscore); |

1.调用基类构造函数代码

|  |  |
| --- | --- |
| 1 | User.call(this,userid,uname,uage,uscore); |

此行代码的含义是执行User类的构造函数，并且将Member的this指针域传递给User类，此时User类运行构造函数时，this指向的是Member对象。

|  |  |  |  |
| --- | --- | --- | --- |
| 几种继承方式的比较  (构造继承 | 原型继承 | 实例继承 | 拷贝继承 |

From JavaScript中继承的实现方法 - xieex - 博客园.htm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 比较项 | 构造继承 | 原型继承 | 实例继承 | 拷贝继承 |
| 静态属性继承 | N | Y | Y | Y |
| 内置（核心）对象继承 | N | 部分 | Y | Y |
| 多参多重继承 | Y | N | Y | N |
| 执行效率 | 高 | 高 | 高 | 低 |
| 多继承 | Y | N | N | Y |
| instanceof | false | true | false | false |

# Java是通过隐式的组合来实现继承

其实在**[Java](http://lib.csdn.net/base/javaee" \o "Java EE知识库" \t "http://blog.csdn.net/perfect2011/article/details/_blank)**内部，是通过隐式的组合来实现继承的。 子类对象中会保存一个实例对象的引用super，该引用指向其父类。 在实际的方法调用时，java会先在当前类的对象中寻找名称相同的方法，如果没有，就到super引用的父类对象中去寻找该方法，所以，若在子类中存在和父类方法的签名和返回值类型完全相同的方法（重写）的话，java就会直接调用该对象的方法而不用去父类去寻找调用方法了。而且在子类对象中，可以直接通过super来调用父类对象中的方法。s

## 常用的apply ,call方法

每种方法都有其适用环境，比如，如果父类带有有参构造函数：这种情况下，prototype就不适用了，可选用apply或call；

　在apply和call中，又该如何取舍呢？在OO的继承中，子类继承于父类，那么它应该也是父类的类型。即，ChildClassByCall、ChildClassByApply应该也是ParentClass类型，但我们用"instanceof"检测一下就会发现，通过apply继承的子类，并非ParentClass类型。所以，我们建议用call + prototype 来模拟实现继承。据说，Google Map API 的继承就是使用这种方式哟。

apply 实现

参考::JavaScript继承的模拟实现 - 牦牛 - 博客园.htm

# **Ruby语言利用Mix-in实现装配的继承**

参见Atitit.继承的实现原理 Mix-in功能.docx

# 参考

javascript实现类、继承、多态（原创） \_ 耕书录.htm

JavaScript中继承的实现方法 - xieex - 博客园.htm