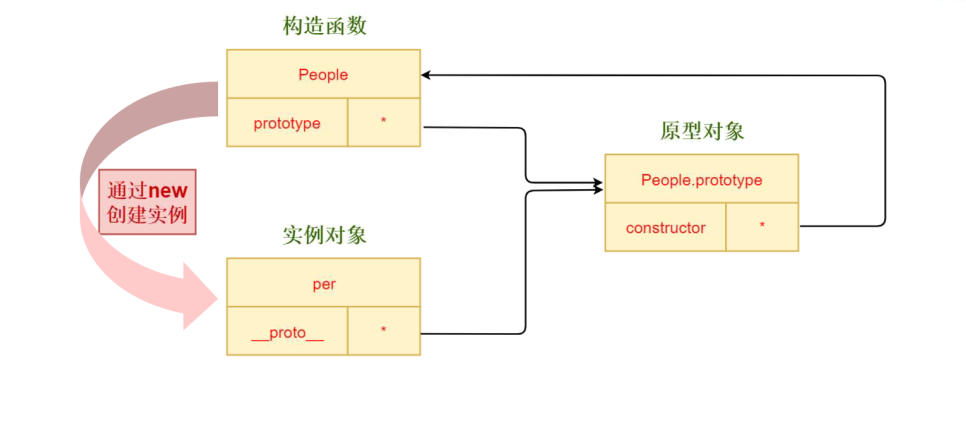
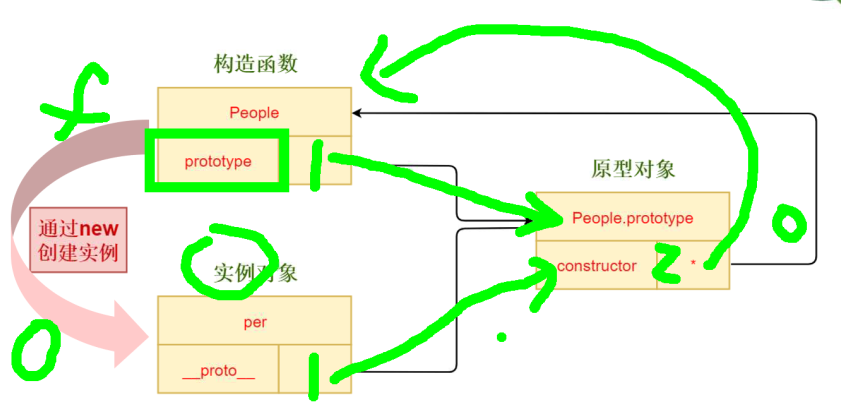


**constructor它来指向创建的对象**



**他们三个相互影响**

这个是系统创建的，它的地址指向构造函数People



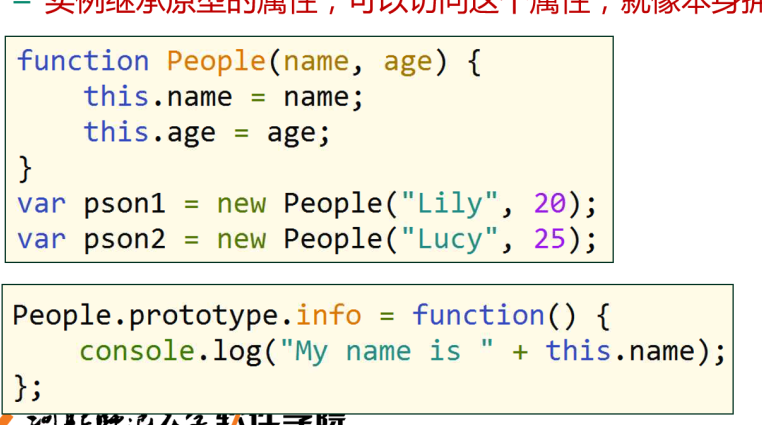
**f是function**

**o是object**

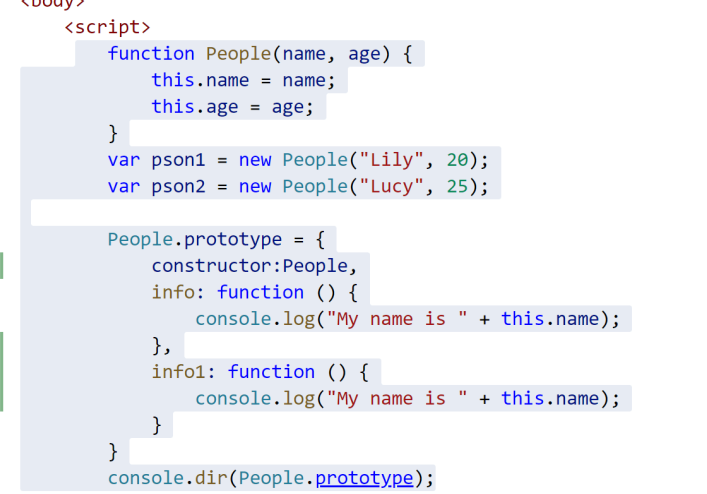
 **是前后两个下划线，\_\_proto\_\_**



**实例可以访问，使用原型的属性，但是本身是没有的。它的原型对象就是它的\_\_proto\_\_指向的对象。**

、

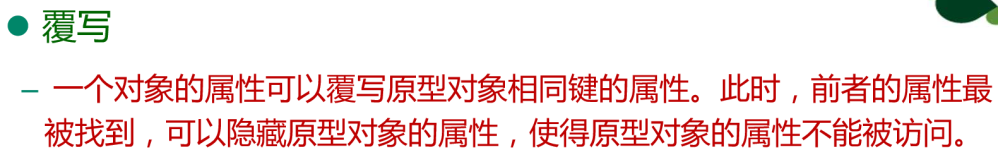
**这个会直接在原型对象里添加info**



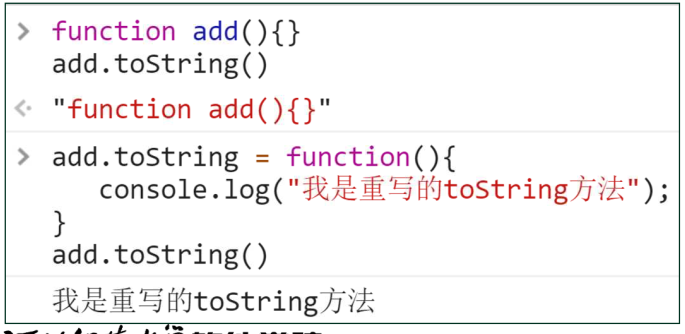
**这个是使用花括号在原型对象中添加多个属性。注意要加上constructor指向原型对象指向的构造函数**



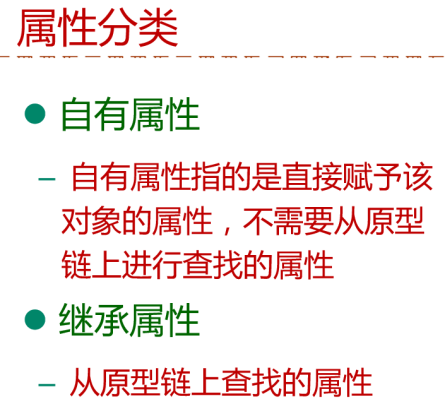


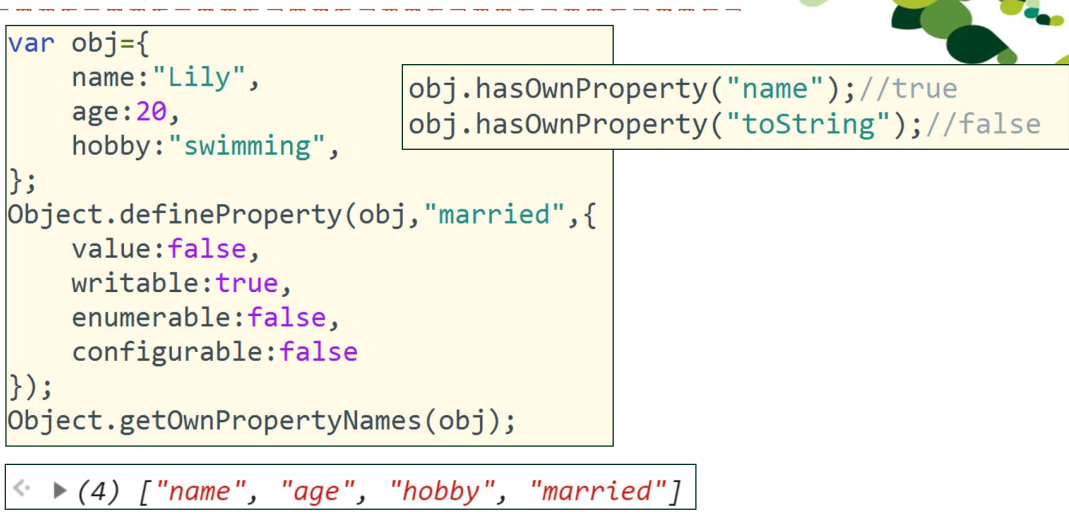


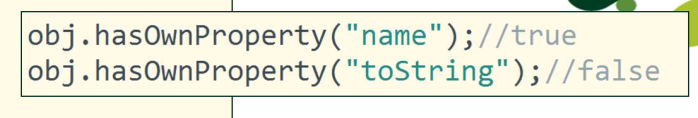
**覆写用的最多的是toSting**



**这里的toString是在原型对象上找到的，本身是没有的。**







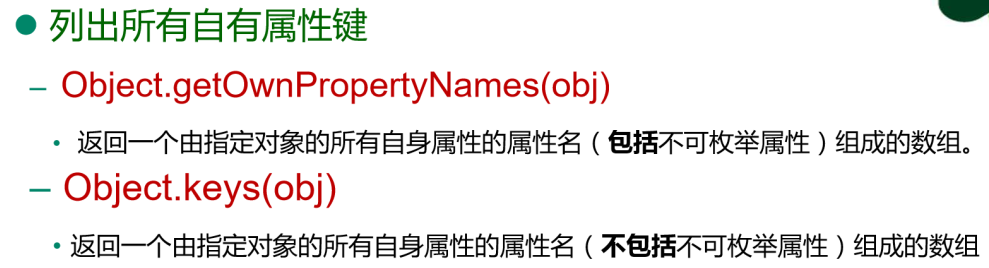
**hasOwnProperty是盘多这个obj里有没有name属性，有没有toString属性。**

**因此，hasOwnProperty是作判断。**



**只能对自有的对象上自有的属性操作**

**可以对可枚举的自有属性，继承属性操作**

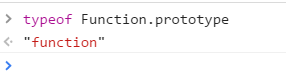


**原型链：**

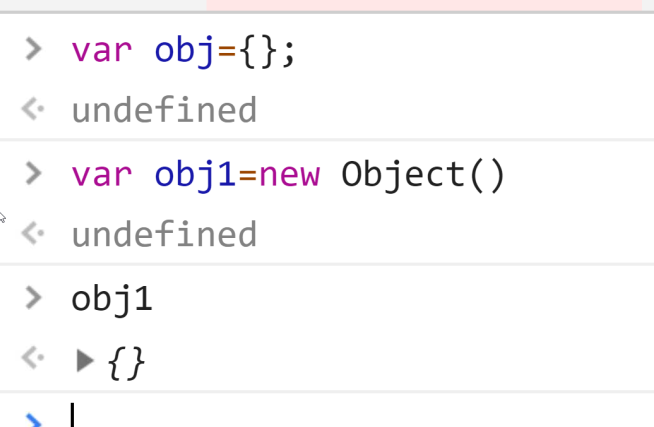
**只要是对象，就会拥有\_\_proto\_\_**

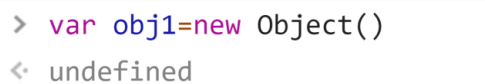
**函数也是对象，是对象中的“一等公民”**

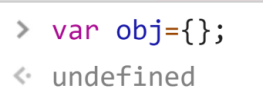
**函数都是通过new Function出来的。**



**注意，函数通过typeof检测，他的类型是function**

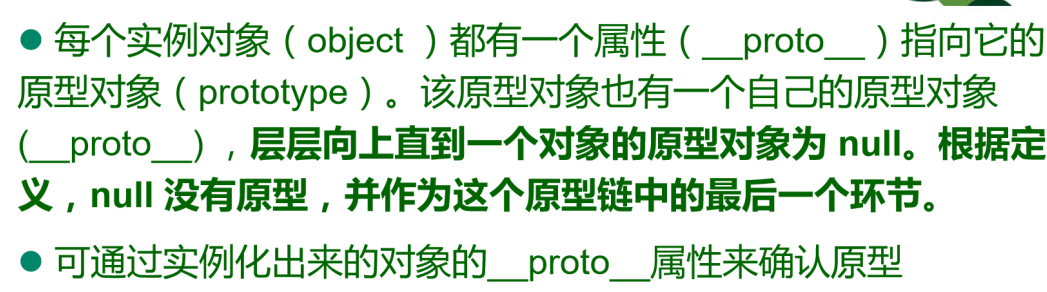


**只要是Object类型，都是通过****创建出来的**

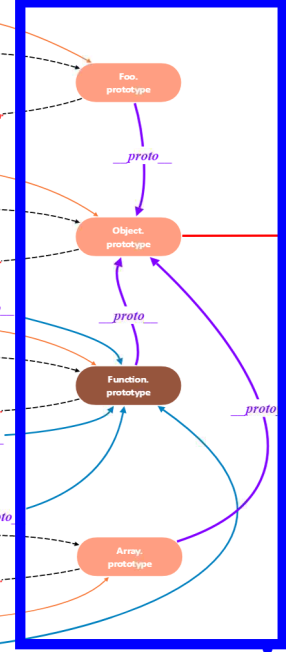
**这个是省略的写法**

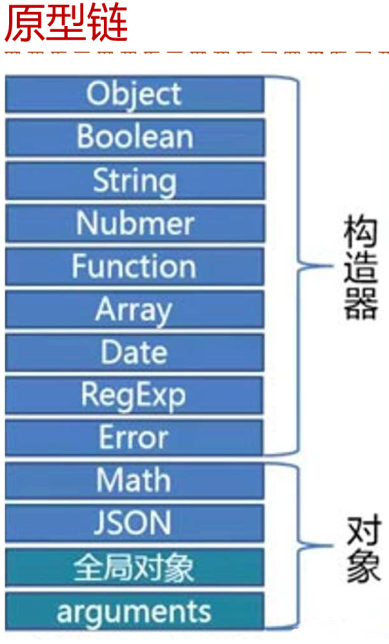
**不是所有的函数都拥有prototype属性，只要拥有prototype属性，它就是一个函数。特殊例子：Function没有prototype属性。**

**JS是基于原型的语言**



**所有原型链的头是Object.prototype**

**都是系统创建的**



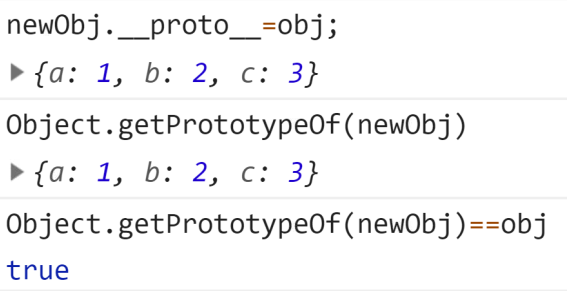
 **使用typrof检测，是个function,（是个函数）它没有prototype属性**

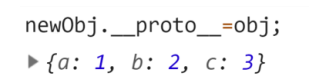
**是函数的起源（函数的头）**

**是对象的起源**

**Console.dir()可以查找自身有没有这个属性**

**构造器的意思就是能不能做一个构造函数**



这条语句改变了原型对象的指向。