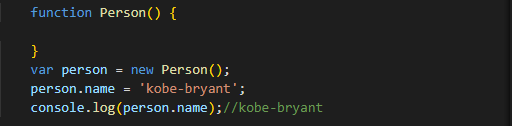
**构造函数创建对象**

我们先使用构造函数创建一个对象：



在这个例子中，Person就是一个构造函数，我们使用new创建了一个实例对象person。

很简单吧，接下来进入正题：

**Prototype**

每个函数都有一个prototype属性，就是我们经常在各种例子中看到的那个prototype，比如：

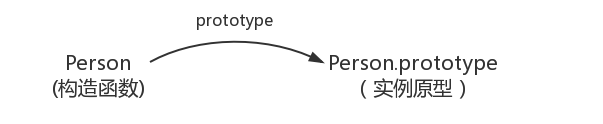


那这个函数的prototype属性到底指向的是什么呢？是这个函数的原型吗？

其实，函数的prototype属性指向了一个对象，这个对象正是调用该构造函数而创建的**实例**的原型，也就是这个例子中的person1和person2的原型。

那什么是原型呢？你可以这样理解：每一个JavaScript对象(null除外)在创建的时候就会与之关联另外一个对象，这个对象就是我们所说的原型，每一个对象都会从原型“继承”属性。

用一张图表示构造函数和实例原型之间的关系

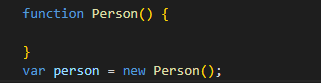


这张图中我们用Object.prototype表示实例原型

那么我们该怎么表示实例与实例原型，也就是person和Person.prototype之间的关系呢，这时候我们就要讲到第二个属性：

**\_proto\_**

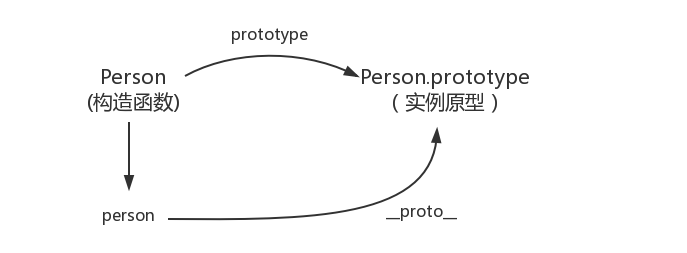
这是每一个JavaScript对象(除了null)都具有的一个属性，叫 \_\_proto\_\_ , 这个属性会指向该对象的原型。



为了证明这一点，我们可以在火狐或者谷歌中输入：



于是我们更新下关系图：

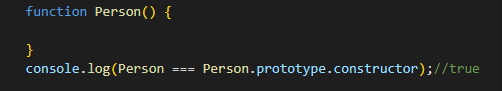


既然实例对象和构造函数都可以指向原型，那么原型是否有属性指向构造函数或者实例呢？

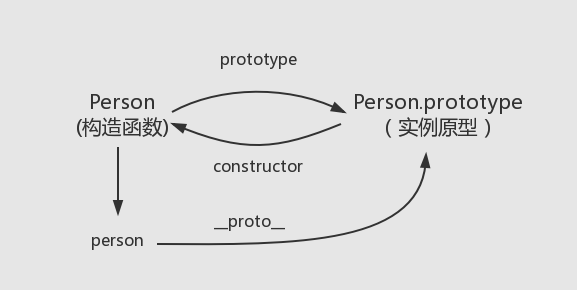
**constructor**

指向实例倒是没有，因为一个构造函数可以生成多个实例，但是原型指向构造函数倒是有的，这就要讲到第三个属性：constructor，每个原型都有一个constructor属性指向关联的构造函数。

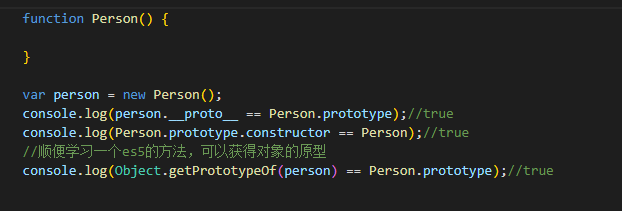
为了证明这一点，我们可以尝试：



所以容我再更新一下关系图：(下页见)



综上所述，我们已经得出：



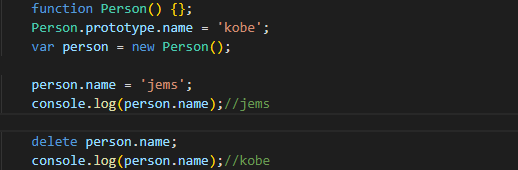
特别提示：\_\_proto\_\_的‘\_\_’是两个’\_’

了解了构造函数，实例原型、和实例之间的关系，接下来我们讲讲实例和原型的关系：

**实例与原型**

当读取实例的属性时，如果找不到，就会查找与对象关联的原型中的属性，如果还查不到，就去找原型的原型，一直找到对顶层为止。

举个例子：



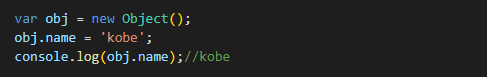
在这个例子中，我们给实例对象person添加了name属性，当我们打印person.name的时候，结果自然为jems

当删除了person的name属性时,读取person.name,从person对象中找不到name属性就会从person的原型也就是person.\_\_proto\_\_ , 也就是Person.prototype中查找，幸运的是我们找到了name属性，结果为kobe.

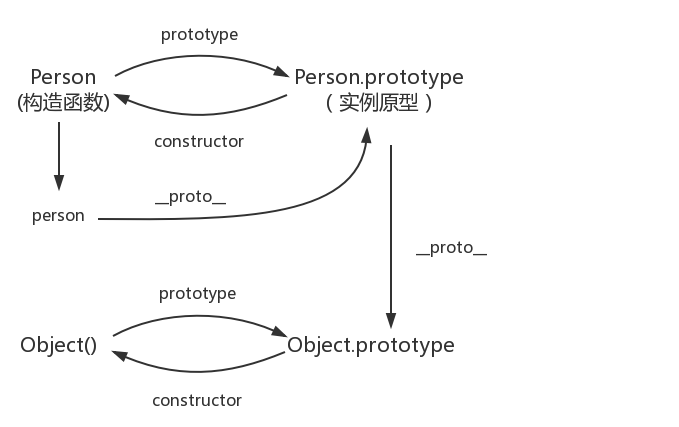
有点同学又会问了，那如果没找到呢？原型的原型又是什么呢？

**原型的原型**

在前面你，我们已经讲了原型也是一个对象，既然是对象，我们就可以用最原始的方式创建它：



其实原型对象就是通过Object构造函数生成的，结合之前所述，实例的\_\_proto\_\_指向构造函数的prototype，所以我们再来更新一下关系图：



**原型链**

那Object.prototype的原型的？

null , 我们可以打印：



竟然是null？然而null究竟代表了什么呢？

既然提到了null，那我们就来学习下吧

undefined和null的区别

大多数计算机语言，有且仅有一个表示“无”的值，比如，C语言的NULL，Java语言的null，Python语言的None, Ruby语言的nil。

有点奇怪的是，JavaScript语言居然有**两个**表示“无”的值：undefined和null。这是为什么？

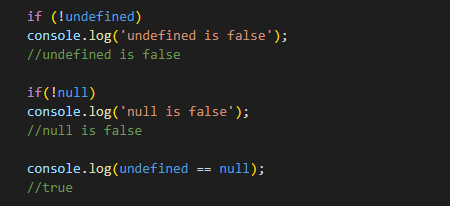
1. **相似性**

在JS中，将一个变量赋值为undefined或null，老实说，几乎没区别。



上面代码中，a变量分别被赋值为undefined和null，这两种写法几乎等价。

Undefined和null在if语句中，都会被自动转为false，相等运算符甚至直接报告两者相等。

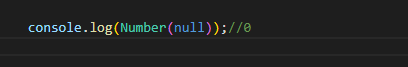


上面代码说明，两者的行为是何等相似！

既然undefined和null的含义与用法都差不多，为什么要同时设置两个这样的值，这不是无端增加js的复杂度嘛？Google公司开发的javascript语言的替代品Dart语言，就明确规定只有null，没有undefined!

**2.历史原因**

这和js的历史有关，1995年js诞生时，最初就像java一样，只设置了null作为表示“无”的值。根据C语言的传统，null被设计成可以自动转为0.如下



但是，js的设计者觉得这样还不够，有两个原因

首先，null像在java里一样，被当成一个对象，但是，js的数据类型分成原始类型(primitive)和合成类型(complex)两大类，Brendan Eich（作者）觉得表示“无”的值最好不是对象。

其次，js最初版本没有包括错误处理机制，发生数据类型不匹配时，往往是自动转换类型或者默默的失败。作者觉得，如果null自动转为0，很不容易发现错误。

因此，Brendan Eich又设计了一个undefined。

**3.最初设计**

Js的最初版本是这样区分的：null是一个表示“无”的对象，转为数值时为0；undefined是一个表示“无”的原始，转为数值时为NaN。



**4.目前用法**

上面这样的区分，在实践中很快就被证明不可行。目前，null和undefined基本是同义的，只有一些细微的差别。

null表示“没有对象”，即该处不应该有值，典型用法是：

1. 作为函数的参数，表示该函数的参数不是对象
2. 作为对象原型链的终点



Undefined表示“缺少值”，就是此处应该有一个值，但是还没有定义。典型用法是：

1. 变量被声明了，但没有赋值时，就等于undefined
2. 调用函数时，应该提供的参数没有提供，该参数等于undefined
3. 对象没有赋值的属性，该属性的值为undefined
4. 函数没有返回值时，默认返回undefined