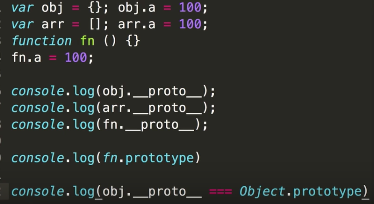
**原型和原型链**

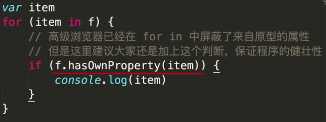
* 构造函数：
  + 构造函数特点：函数名首字母大写；
  + new对象返回的过程：
    - 创建一个新对象；
    - this指向这个新对象；
    - 通过传参给this赋值；
    - 返回this；
    - 赋值给变量；
* 构造函数-扩展：
  + var a = { } 其实是 var a = new Object( );
  + var a = [ ] 其实是 var a = new Array( );
  + function Foo( ){ } 其实是var Foo = new Function( );
  + 使用instanceof判断一个函数是否是一个变量的构造函数；
* 原型规则和实例：



* + 所有的引用类型，都具有对象特性，即可自由扩展属性（除了“null”）；
  + 所有的引用类型（除了“null”），都有一个\_proto\_（隐式原型）属性，属性值是一个普通的对象，指向自身的原型。
  + 所有的函数，都有一个prototype（显示原型）属性，属性值也是一个普通的对象，指向原型对象；
  + 所有的引用类型，\_proto\_属性值指向它的构造函数的 ”prototype” 属性值；
  + 当试图得到一个对象的某个属性时，如果这个对象本身没有这个属性，那么会去它的\_proto\_（即它的构造函数的显示原型）中寻找；

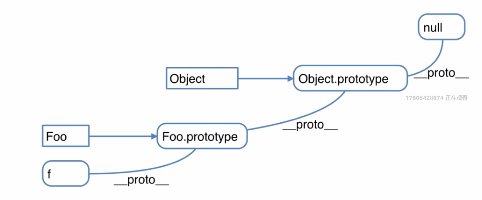


* 通过对象的属性的形式去执行函数时，无论这个属性是自身属性，还是从函数原型中找到的属性，this永远指向函数自身；
* 循环对象自身的属性时，需要加一个判断：



* 原型链：真正形成原型链的是每个对象的\_proto属性，而不是函数的prototype属性；





* instanceof：
  + 用于判断引用类型属于哪个构造函数的方法（f instanceof Foo）；
  + 判断逻辑：函数的\_proto\_一层一层往上，能否对应构造函数的prototype；
* 原型链继承的例子：

