JavaScript的面向对象是基于原形的，所有对象都有一条属于自己的原型链。Object与Function可能很多看Object instanceof Function ， Function instanceof Object都为true而迷惑，所以首先看下对象的实例。

**一、JS中所谓的实例**

1. 如var a = new A();这样子通常的认为 “a为A函数的实例对象”。

2. new操作的过程是什么？

          1.new创建一个空对象{}称为小C

          2.然后将A.prototype放置到小C的原型链顶端。即小C.\_\_proto\_\_指向 A.prototype

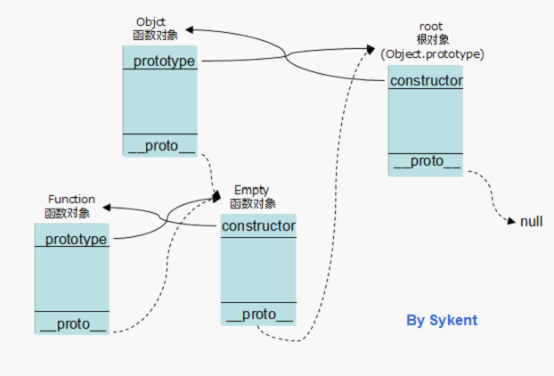
          3.执行A函数，将A中this指向小C，执行结束，如果没有return那么默认返回this引用。

那么new的其中一个的作用便是把A.prototype添加到了a的原型链中。

3. instanceof为js中判断实例的方法，判断的根据就是(例 a instanceof A) 判断a的原型链中是否含有A.prototype。

4. 综上包括常规认识new出来的实例。  所谓o 为fn的实例对象，实际就是指o原型链中含有fn.prototype

二、Object与Function的模糊图解(自己画的很搓)



解释： 所有函数的prototype属性均有constructor指向该函数。

       所有对象均有原型链(Object.prototype的原型链为null)。

      \_\_proto\_\_（chrome, ff中）代表[[proto]]内置属性。

看过这个图之后应该就明白为什么instanceof是那个的结果了。

三、对象与Object的关系，原型链终结于Object.prototype

     Object.prototype的原型链为空(即没有一个原型对象)。因此Object.prototype instanceof Object 为false。

    判断类型用typeof , typeof(Object.prototype) === ‘object’.所以不是所有的对象均为Object对象实例。应该是除了Object.prototype除外的所有对象均为Object对象的实例。