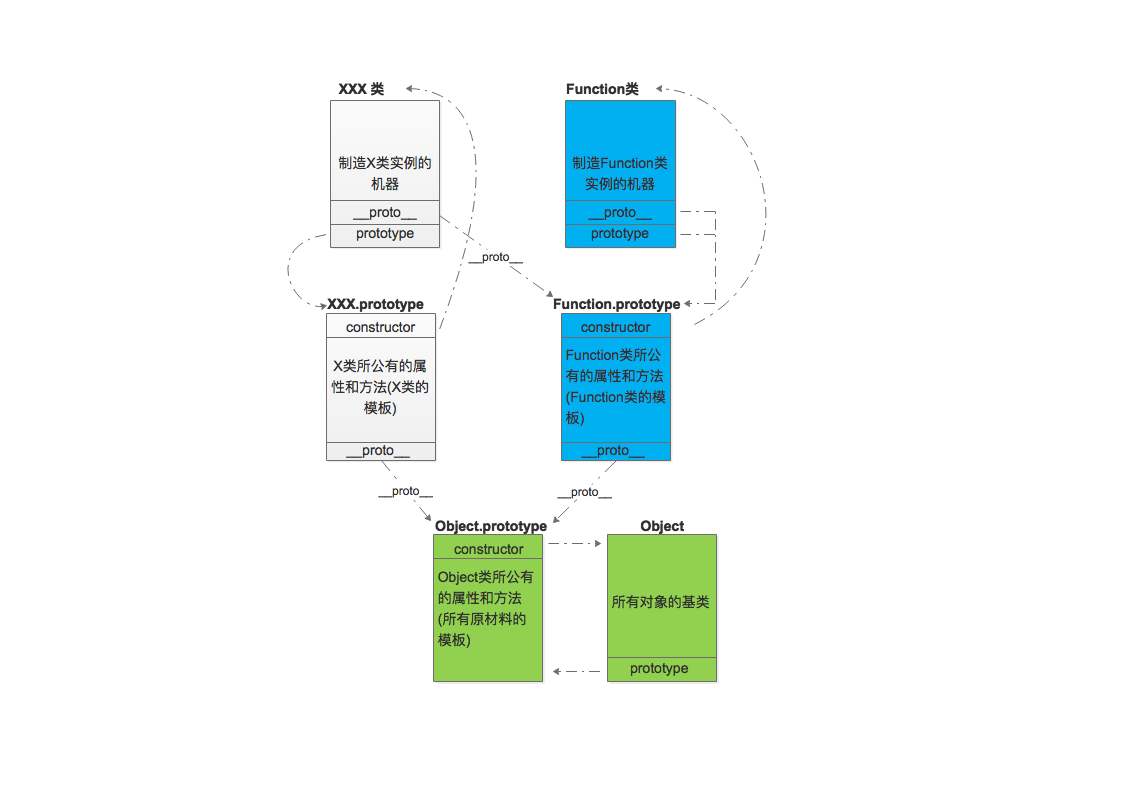
关于面向对象的粗略总结



总结：

1. 一个对象的\_\_proto\_\_指向哪一个类的原型prototype,那么该对象就属于哪个类
2. 所有类的\_\_proto\_\_都指向Function原型，所以得出所有的类(内置类和自定义类)都属于函数类，也是函数的一个实例; Function类的\_\_proto\_\_指向它自己的原型，所以Function.\_\_proto\_\_===Function.prototype
3. 所有类的原型中都包括特定类的实例天生都具备的属性和方法(继承)
4. 所有类的原型(包括函数类的原型)的\_\_proto\_\_都指向Object.prototype, 都属于对象类的一个实例，所以说万物皆对象。

Object相当生产原材料（Object.prototype）的工厂，Function类相当于一个可以制造出各种**类**的机器(函数)来生产加工东西(实例instance)。但是生产某一个类的东西需要一个模板(prototype)，我们在工厂Object生产原材料来做出一个属于某类的模板(xxx.prototype)，所以全部类的模板都基于工厂Object生产的原材料(xxx.prototype.\_\_proto\_\_===Object.prototype)。

比如生产飞机，需要以下这些步骤:

//造出一部制造飞机类的机器  
**function** Aircraft(model) {  
 **this**.model = model //飞机的型号(私有属性)  
}  
  
//飞机需要一个模板，并需要指定出飞机类天生就自带的特性(属性)和功能(方法)  
Aircraft.prototype = {  
 constructor:Aircraft, //指定该模板是用于Aircraft这部机器的(构造函数)  
 wings:2, //有两个机翼  
 engine:1, //一部引擎  
 fly:**function**(){  
 //可以飞  
 },  
 radio:**function**(){  
 //可以和地面通讯  
 }  
}  
  
//开始生产一部747飞机  
**var** aircraft1=**new** Aircraft(747);  
  
console.log(aircraft1.model); //型号是747  
console.log(aircraft1.\_\_proto\_\_ === Aircraft.prototype); //TRUE 747是按照Aircraft的模板(Aircraft.prototype)来生产的  
console.log(aircraft1 **instanceof** Aircraft); //TRUE 747本来就是按照Aircraft的模板(Aircraft.prototype)做出来的一架飞机(一个飞机类的实例)  
console.log(aircraft1 **instanceof** Object); //TRUE 747使用的模板(Aircraft.prototype)是基于原材料Object.prototype，所以她归根结底也是Object的一个实例，万物皆对象~！