#### 1.原型

原型链作是实现继承的主要方法。

利用原型链让一个引用类型去继承另一个引用类型的属性和方法

#### 1.1.1原型对象

原型对象就相当于一个公共的区域，所有同一个类的实例都可以访问到这个原型对象。

则可以将对象中共有的对象和方法，统一设置到原型对象中，也不会影响到全局作用域，就可以使每个对象都具有这些属性和方法。

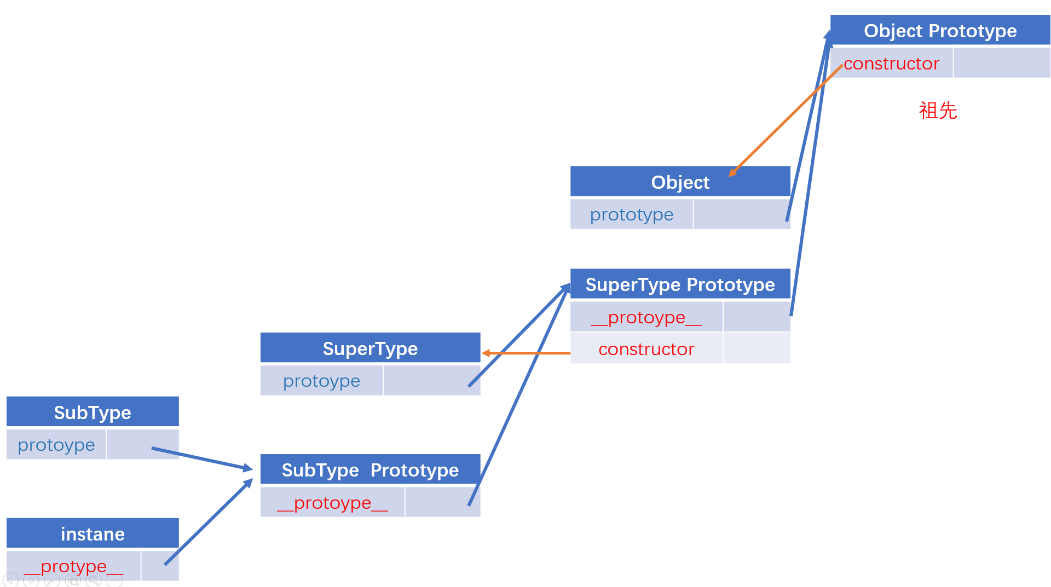
prototype（原型），我们所创建的每一个函数，解析器都会向函数中添加一个属性prototype，这个属性所对应的就是原型对象。

\_\_proto\_\_，当函数通过构造函数形式调用时，他所创建的函数中都会有一个隐含属性\_\_proto\_\_来指向原型对象。

#### 1.1.2原型链

层层递进，原型的原型便形成了链条。

当访问对象的方法或属性时，先在对象自身中寻找，如果有则直接使用，如果没有则会去原型对象中寻找。原型对象中找不到，就继续往原型对象的原型对象中继续寻找，直到找到Object对象的原型（祖先，没有原型）如果在Object原型中依然没有找到，则返回undefined



        function SuperType(){

        }

        function SubType(){

        }

        SubType.prototype = new SuperType()

        var instance = new SubType()

#### 1.1.2原型链的问题

通过原型实现继承的时候，原型会变成另一个对象的实例，其所有属性都会被共享，当修改一个，其他的都会改变。