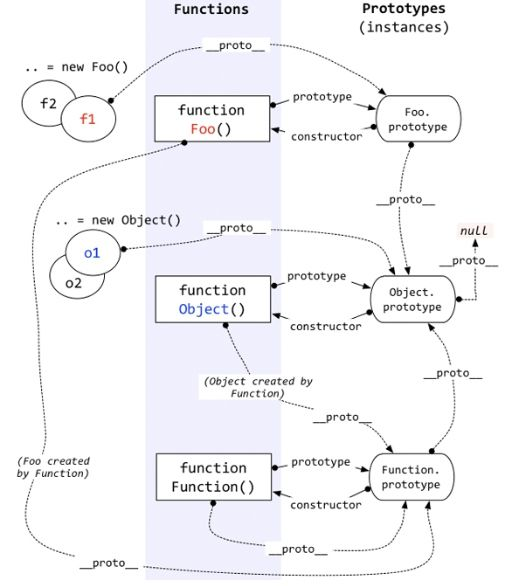
原型与原型链：



原型链是实现继承的主要方法，基本思想就是利用原型让一个引用类型继承另一个引用类型的属性和方法。

每当定义一个对象（函数也是对象）时候，对象中都会包含一些预定义的属性。其中每个****函数对象****都有一个prototype 属性，这个属性指向函数的****原型对象****。

要理解原型链就要明白原型对象、构造函数、实例，三者之间的关系。

我们先来梳理三者的关系：

每个构造函数都有一个原型对象，原型对象都包含一个指向构造函数的指针，实例也有一个指向原型对象的内部指针。

当这个原型对象是另一个类型的实例，那么这个原型对象就有一个内部指针指向另一个原型，以此类推就构成了一条原型链。原型链的根就是Object.prototype。

每个对象都有 \_\_proto\_\_ 属性，但只有函数对象才有 prototype 属性

所有函数对象proto都指向 Function.prototype，它是一个空函数（Empty function）

原型和原型链是JS实现继承的一种模型。

原型链的形成是真正是靠\_\_proto\_\_ 而非prototype

闭包

闭包就是能够读取其他函数内部变量的函数，只有函数内部的子函数才能读取局部变量。

作用：一个是前面提到的可以读取函数内部的变量，另一个就是让这些变量的值始终保持在内存中，不会在f1调用后被自动清除。

原因就在于f1是f2的父函数，而f2被赋给了一个全局变量，这导致f2始终在内存中，而f2的存在依赖于f1，因此f1也始终在内存中，不会在调用结束后，被垃圾回收机制（garbage collection）回收。

Ajax

new 的过程

对于 var o = new Foo();

1、var o = new Object();

2、o. \_\_proto\_\_ = A.prototype;//这里还记得之那个function里面的默认的属性么?

3、A.call(o)//由于这里this是指向o,可以把什么this.name/getName绑定到o上.

4、把这个o返回给a;//完成var a = new A()的过程.

new是在进行实例化，实例继承了原型对象的属性和方法。

首先先创建一个空对象o,将o的隐式原型执行构造函数的显式原型。

然后调用call方法将this指向对象o，

将对象o返回给a。

正则表达式：

作用域：

js的作用域分两种，全局和局部，在js作用域环境中访问变量的权利是由内向外的，内部作用域可以获得当前作用域下的变量并且可以获得当前包含当前作用域的外层作用域下的变量，反之则不能。

作用域链：函数对象和其它对象一样，拥有可以通过代码访问的属性和一系列仅供JavaScript引擎访问的内部属性。其中一个内部属性是[[Scope]]，该内部属性包含了函数被创建的作用域中对象的集合，这个集合被称为函数的作用域链，它决定了哪些数据能被函数访问。

当一个函数创建后，它实际上保存一个作用域链，并且作用域链会被创建此函数的作用域中可访问的数据对象填充。

闭包：在一个函数内部也有限权访问另一个函数内部的变量该怎么办呢？闭包就是用来解决这一需求的，闭包的本质就是在一个函数内部创建另一个函数。