Js 面向对象

一切皆对象

变量类型:字符串、数字、布尔、对象（数组，日式时间，json）、Null、Undefined

所有变量都是对象，除了两个例外 null 和 undefined。

变量赋值及比较

普通变量比较相等　== 值相同为true
对象比较相等　　== 值和引用都相同才为true

作用域

变量的作用域（scope）是程序中定义这个变量的区域。变量分为两类：全局（global）的和局部的

在变量解析过程中首先查找局部的作用域，然后查找上层作用域

this

在函数中this到底取何值，是在函数真正被调用执行的时候确定的，函数定义的时候确定不了。

this 出现的场景分为四类，简单的说就是

有对象就指向调用对象
没调用对象就指向全局对象
用new构造就指向新对象
通过 apply 或 call 或 bind 来改变 this 的所指。

创建对象

对象都是通过函数创建的

var obj=new 函数()

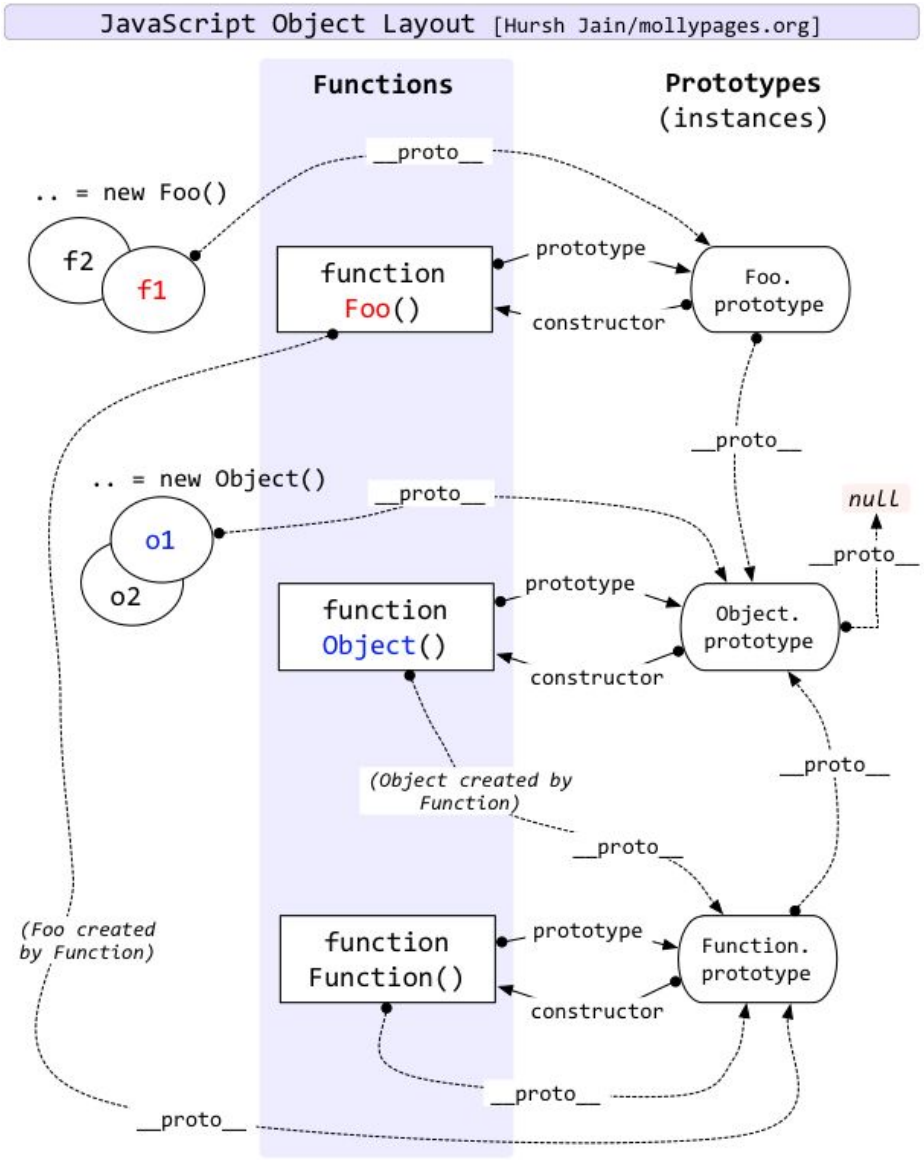
对象里面包含一个constructor属性指向构造函数

原型

Js所有的函数都有一个prototype属性，这个属性引用了一个对象，即原型对象，也简称原型

http://blog.csdn.net/kittyjie/article/details/4380918

原型链



//原型链：实例对象与原型之间的连接，叫做原型链

//原型链的最外层：Object.prototype

//属性在查找的时候是先查找自身的属性，如果没有再查找原型，再没有，再往上走，一直查找到Object的原型上

//图片链接http://laruence-wordpress.stor.sinaapp.com/uploads/javascript_object_layout.jpg

对象实例　原型　构造函数关系

对象都是通过函数创建的

我们知道每个函数都有一个原型，这个原型是个对象，并且对象里面包含一个constructor属性。constructor又指回自身。

hasOwnProperty用来判断一个对象是否有你给出名称的属性或对象

封装

属性,方法

带this为public 否则为private

prototype定义的是静态属性，静态方法

继承

属性继承: Bar.call(this,'张三','18');

方法继承: Bar.prototype = new Foo();

对象拷贝

浅拷贝

浅拷贝对象A时，对象B将拷贝A的所有字段，如果字段是内存地址，B将拷贝地址，若果字段是基元类型，B将复制其值。

浅拷贝的缺点是如果你改变了对象B所指向的内存地址，你同时也改变了对象A指向这个地址的字段

深拷贝

深拷贝，这种方式会完全拷贝所有数据，优点是B与A不会相互依赖（A,B完全脱离关联），缺点是拷贝的速度更慢，代价更大

实例