Js原型链：

在js中，类概念模糊，创建对象通过clone基本对象Object.prototype得出，修改原型所指向的对象，从而使继承关系由单一的Object到其他不同的对象。

即：

|  |  |  |
| --- | --- | --- |
| var a=new Object()  a.name=”1”;  a.mothed=function(){  return this.name;  } | var b={  name:1,  mothed : function(){  return this.name;  }  } | var c=function（）{  This.name=”1”}  c.prototype.mothed=function(){  return this.name;  }  var c\_c=new c(); |

上述三种创建对象的方法中，第二种是第一种的简写方式，第三种是第一种的实际形式，第一种a.mothed等价于c.prototype.mothed。且c.prototype===Object,并非Object.prototype;

如果将c.prototype指向对象更改成其他的自定义对象d，按照链型，c\_c就继承于d,而非继承于Objcet，

原型链中如果c\_c对象的调用方法不存在，就会依照链型向上寻址，直至Object,对于Object.prototype===undefined，此时原型链结束。

原型链仅能保证单继承，对于多继承，如果在原型链中赋值其他的对象，会被覆盖。

This关键字

在js中this是动态绑定的，它指向的是一个对象，而这个对象在普通函数中指向的是window对象，在构造器中则是指向当前实例对象。

在js中方法名可以直接赋值给变量，此时的函数就变成普通的函数，如果该方法中使用了this，这this指向的就是该变量的所属对象，而不是该函数的所属对象，就会产生指向错误，为保证this能够正确指向，可以通过apply和call将该方法中的this进行具体替换。

由于this指向替换，在实际的函数体中，通过冒用对象的方法，给想要的对象增加冒用对象的一些属性，从而到达继承的特性。

为保证能够多继承