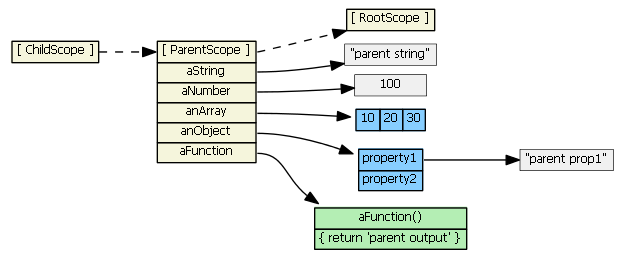
假设父类 parentScope 有如下成员属性 aString, aNumber, anArray, anObject, 以及 aFunction。子类 childScope 原型继承父类 parentScope，于是我们有：

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance1.png)

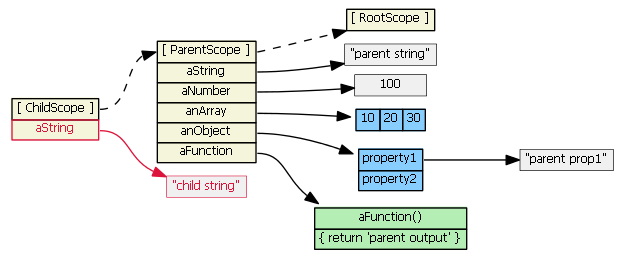
**如果子 Scope 尝试去访问 parentScope 中定义的属性，JavaScript 会先在子 Scope 中查找，如果没有该属性，则找它继承的 scope 去获取属性，如果继承的原型对象 parentScope 中都没有该属性，那么继续在它的原型中寻找，从原型链一直往上直到到达 rootScope。**所以，下面的表达式结果都是 ture：

|  |
| --- |
| childScope.aString === 'parent string'  childScope.anArray[1] === 20  childScope.anObject.property1 === 'parent prop1'  childScope.aFunction() === 'parent output' |

假设我们执行下面的语句

|  |
| --- |
| childScope.aString = 'child string' |

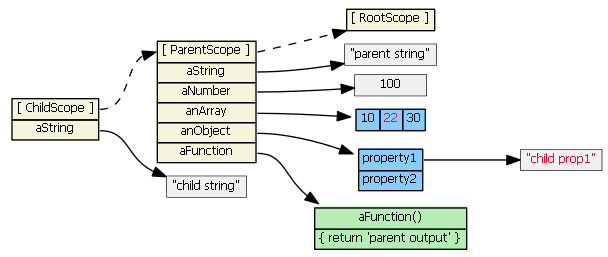
原型链并没有被查询，反而是在 childScope 中增加了一个新属性 aString。这个新属性隐藏（覆盖）了 parentScope 中的同名属性。在下面我们讨论 ng-repeat 和 ng-include 时这个概念很重要。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance2.png)

假设我们执行这个操作：

|  |
| --- |
| childScope.anArray[1] = '22'  childScope.anObject.property1 = 'child prop1' |

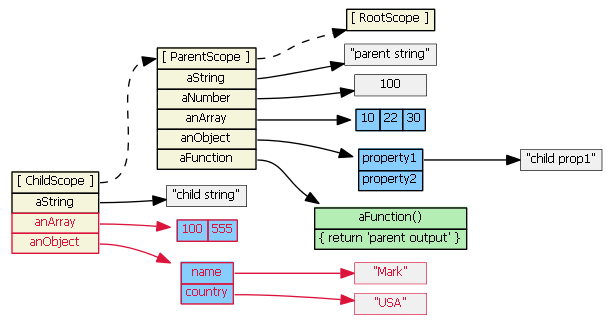
原型链被查询了，因为对象 anArray 和 anObject 在 childScope 中没有找到。它们在 parentScope 中被找到了，并且值被更新。childScope 中没有增加新的属性，也没有任何新的对象被创建。（注：在 JavaScript 中，array 和 function 都是对象）

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance3.png)

假设我们执行这个操作：

|  |
| --- |
| childScope.anArray = [100, 555]  childScope.anObject = { name: 'Mark', country: 'USA' } |

原型链没有被查询，并且子 Scope 新加入了两个新的对象属性，它们隐藏（覆盖）了 parentScope 中的同名对象属性。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance4.png)

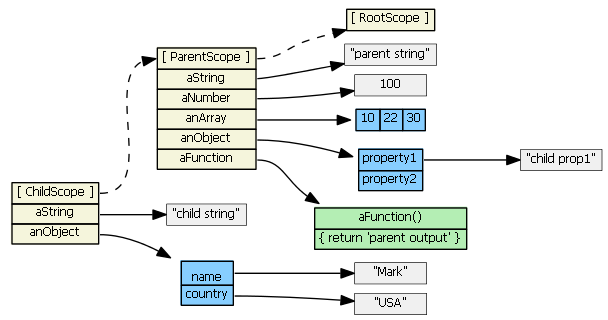
应该可以总结

* 如果读取 childScope.propertyX，并且 childScope 有属性 propertyX，那么原型链没有被查询。
* 如果设置 childScope.propertyX，原型链不会被查询。

最后一种情况，

|  |
| --- |
| **delete** childScope.anArray  childScope.anArray[1] === 22 *// true* |

我们从 childScope 删除了属性，则当我们再次访问该属性时，原型链会被查询。删除对象的属性会让来自原型链中的属性浮现出来。

[](http://cdn.lovelucy.info/wordpress/wp-content/uploads/2013/07/angularjs-inheritance5.png)