再谈继承

在阅读本文之前，请先查看继承与原型链一文以了解关于JavaScript继承和构造器原型的知识。

在JavaScript中一直在使用继承机制，但是本页面使用了一些ECMAScript5中引入的方法。请查看这些方法的不同之处，并考虑它们是否可以效仿。

例子

B继承于A:

```
function A(a){
  this.varA = a;
}

A.prototype = {
  varA : null,
  doSomething : function(){
     }
}

function B(a, b){
  A.call(this, a);
  this.varB = b;
}
B.prototype = Object.create(A.prototype, {
  varB : {
    value: null,
    enumerable: true,
    configurable: true,
    writable: true
  },
  doSomething : {
    value: function(){      A.prototype.doSomething.apply(this, arguments);
},
    enumerable: true,
    configurable: true,
    writable: true
  }
});

var b = new B();
```

```
b.doSomething();
```

最重要的两个地方是:

- 类型定义在了prototype属性中
- 使用了Object.create()来实现继承

原型和Object.getPrototypeOf

对于那些从Java或c++转来的程序员来说,JavaScript肯定会让他们感到困惑,因为JavaScript是动态的,完全运行时的,而且它竟然没有类,只有实例(对象).有些人嘴里的"类"也仅仅是一个用来模拟类的函数对象.

你可能已经注意到了,在上例中,函数A有一个特殊的属性prototype.这个属性主要是和new运算符一起工作的.当使用new运算符生成对象实例的时候,实例会有一个称之为[[Prototype]]的内部属性,指向了构造函数的prototype属性指向的那个对象.比如,当你执行var a1 = new A()的时候,JavaScript引擎内部会执行类似a1.[[Prototype]] = A.prototype这样的操作(在创建了a1指向的那个对象之后,但在运行A()之前).

当你访问一个对象的属性时,JavaScript会首先看这个属性是否存在于该对象自身上,如果没有,就会到它的内部属性[[Prototype]]指向的那个对象的身上找.这也就实现了,构造函数的prototype属性指向的那个对象上的所有属性被该构造函数的所有实例所共享,如果你改变了prototype属性指向的那个对象身上的某个属性的值,则所有的实例都会受到影响.

比如,基于上的那个例子,如果你执行了var a1 = new A(); var a2 = new A();那么a1.doSomething实际上访问的就是Object.getPrototypeOf(a1).doSomething,也就是你一开始定义的A.prototype.doSomething.更清楚点,也就是Object.getPrototypeOf(a1).doSomething == Object.getPrototypeOf(a2).doSomething == A.prototype.doSomething.

原型链的遍历是递归进行的,也就是说,查找过程是这样的:a1.doSomething,
Object.getPrototypeOf(a1).doSomething,
Object.getPrototypeOf(Object.getPrototypeOf(a1)).doSomething等等,一直到找到那个属性,或者到达原型链顶层(Object.getPrototypeOf方法返回null的时候).

所以,当你在执行:

```
var o = new Foo();
```

的时候,JavaScript实际上给你做了类似这样的操作:

```
var o = new Object();
o.[[Prototype]] = Foo.prototype;
o.Foo();
```

这是,如果你访问

```
o.someProp;
```

它会首先查看o对象本身是否有属性someProp.如果没有,则检查

Object.getPrototypeOf(o).someProp是否存在,如果仍不存在,就继续检查

Object.getPrototypeOf(Object.getPrototypeOf(o)).someProp,依次类推.