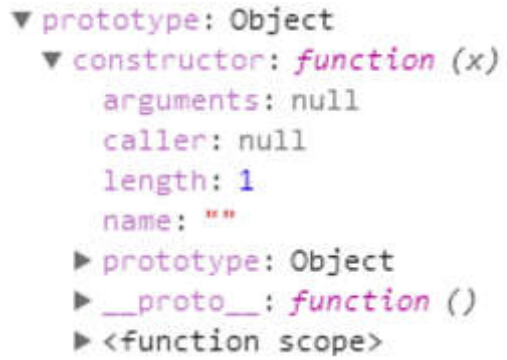
javascript学习--2017-11-9：第九章：类与模块

笔记本：	javascript		
创建时间：	2017/11/9 9:33	**更新时间：**	2017/11/9 16:33
作者：	xiethan		

- 每个javascript 对象都是一个属性集合，相互之间没有任何联系
- 在javascript中，类的实现是 基于其原型继承机制，如果两个实例都是从同一个原型对象上继承了属性，我们说它们是同一个类的实例
- 在javascript中，可以模拟出好多类的特性
- 其类一个重要的特性：动态可继承

类与原型
类与构造函数



构造函数：是用来初始化新创建的对象，使用关键字new来调用；新对象的prototype
从某种意义上讲：定义构造函数既是定义类，并且类名首字母要大写。而且普通的函数和方法都是首字母小写
检测对象是否属性某个类时
r instanceof Range

constructor属性
将Range。prototype定义为一个新对象，这个对象包含所需要的方法。任何javascript函数都可以用作构造函数，并且调用构造函数需要用到一个prototype属性。因此每个javascript函数都自动拥有一个prototype属性，这个属性值是一个对象，这个对象包含一个不可枚举属性constructor。constructor属性是一个函数对象

```
//构造函数
var F=function(){};
var p=F.prototype;
var c=p.constructor;
console.log(c===F);//true
F.prototype.constructor==F //true
```

可以看出构造函数的原型中存在预先定义好的constructor属性，意味着对象通常继承的是constructor，均值代它们的的构造函数，由于构造函数是类的“公共标识”，因此这个constructor属性为对象提供了类。

```
var o=new F();
o.constructor===F;//true;
```

javascript中的java式类继承

在javascript中定义类的步骤为三步

1. 先定义一个构造函数，并设置初始化新对象的实例属性
2. 给构造函数的prototype对象定义实例的方法
3. 给构造函数定义类字段和类属性

```
//javascript定义类的三步法
//complex.js，来表示复数的类
//第一步，这个构造函数为它所创建的每个实例定义了实例r和i
function Complex(real,imaginarty){
if(isNaN(real) || isNaN(imaginarty))
throw new TypeError;
this.r=real;
this.i=imaginarty;
}
```

```
//第二步：当前复数对象加上另外一个复数，并返回一个新的计算和值后的复数对象
Complex.prototype.add=function(that){
return new Complex(this.r+that.r,this.i+that.i);
}

//第三步：类字段和类方法 直接定义为构造函数的属性类
//类字段， ，大写代表常量
Complex.ZERO=new Complex(0,0);
//类方法
Complex.parse=function(s){
try{
var m=Complex._format.exec(s);//利用正则表达式进行匹配,方法在下面
return new Complex(parseFloat(m[1]),parseFloat(m[2]));
}catch (x){
throw new TypeError("匹配出错");
}
}
//"私有字段"，下划线代表它在类内部使用，不属于类的公有api的部分
Complex._format=/^\{([^,],[^,])\}$/;
//运行
var o=new Complex(1,2);
var o2=new Complex(1,2);
console.log(o.add(o2));
```

```
true
The Window
▼ Complex {r: 2, i: 4}
    i: 4
    r: 2
  ▼ __proto__: Object
    ▶ add: function (that)
    ▼ constructor: function Complex(real,imaginarty)
        arguments: null
        caller: null
        length: 2
        name: "Complex"
      ▼ prototype: Object
        ▶ add: function (that)
        ▶ constructor: function Complex(real,imaginarty)
        ▶ __proto__: Object
      ▶ __proto__: function ()
      ▶ <function scope>
    ▼ __proto__: Object
      ▶ __defineGetter__: function __defineGetter__()
      ▶ __defineSetter__: function __defineSetter__()
      ▶ __lookupGetter__: function __lookupGetter__()
      ▶ __lookupSetter__: function __lookupSetter__()
      ▶ constructor: function Object()
      ▶ hasOwnProperty: function hasOwnProperty()
      ▶ isPrototypeOf: function isPrototypeOf()
      ▶ propertyIsEnumerable: function propertyIsEnumerable()
      ▶ toLocaleString: function toLocaleString()
      ▶ toString: function toString()
      ▶ valueOf: function valueOf()
      ▶ get __proto__: function get __proto__()
      ▶ set __proto__: function set __proto__()
```

在java中可以使用fina声明字段为常量，并且可以将字段和方法声明private，用以表示它们是私有成员且在类外面是不可见的。在javascript中没有这些关键字，使用一些写法上的约定来暗示。如上面代码中的大写，下划线

类的扩充

如可以给ECMAScript3中，的函数类添加一个bind()方法，

```
if(!Function.prototype.bind){
Function.prototype.bind=function(o){
}
}
```

类与对象：

javascript的数据类型：null ,undefined ,布尔值，数字，字符串，函数，对象（typeof 可以得出值的类型）

如果o继承自c.prototype，则，o instanceof c的值是 true

//isprototypeOf();