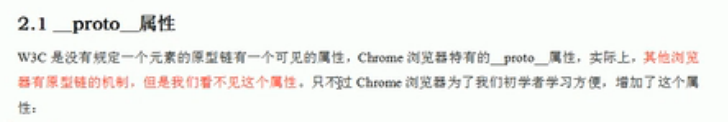
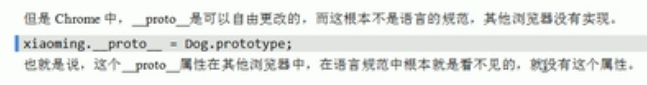
17原型对象属性



node也有\_\_proto\_\_机制（node基于chrome的v8引擎）但是看不到实际的内容

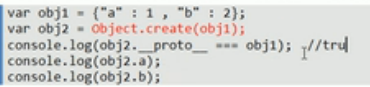
IE、FireFox等浏览器中可能没有。









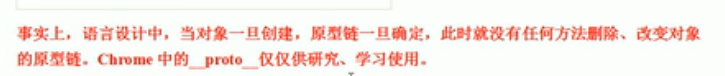


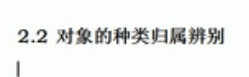
instanceof比较的是\_\_proto\_\_属性

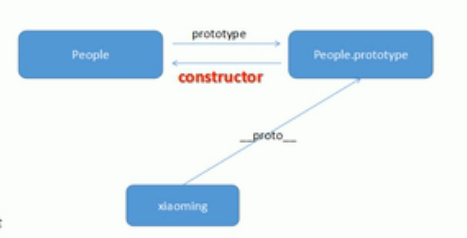


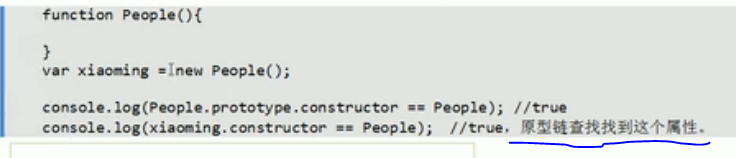
显然任何一个对象都有原型对象，原型链的终点就是Object.prototype











原型链查找

任何一个构造函数的prototype身上都有一个constructor属性，指向构造函数;

任何一个构造函数实例身上也有一个constructor属性，也指向构造函数

function Person(name,age,sex){

this.name=name;

this.age=age;

this.sex=sex;

}

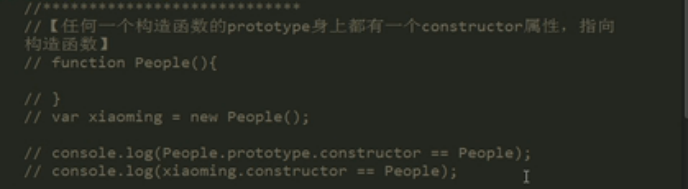
var a=new Person('Daniel',12,1);

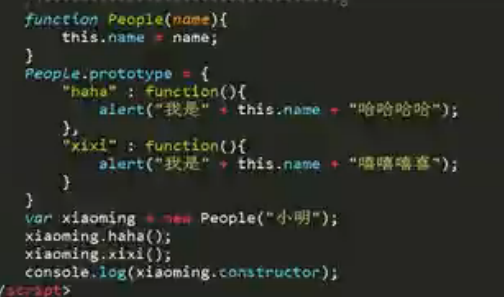
console.log(Person.prototype.constructor);

console.log(a.constructor);

console.log('Person===Person.prototype.constructor? '+(Person===Person.prototype.constructor));//true

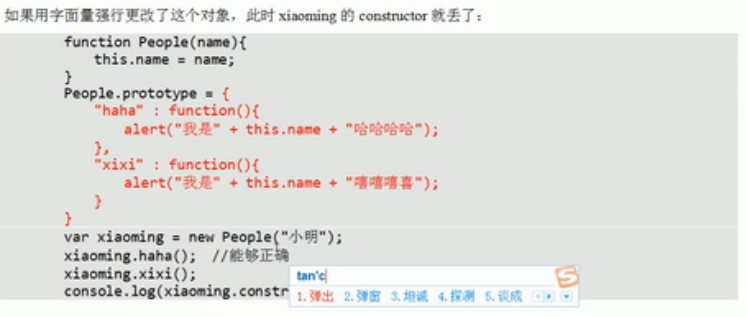
console.log('a.constructor==Person? '+(a.constructor==Person));//true





直接修改了prototype对象，丢失了constructor属性











Instanceof运算符用来判断某个对象是不是某个构造函数的实例

本质上：

对象 instanceof 函数

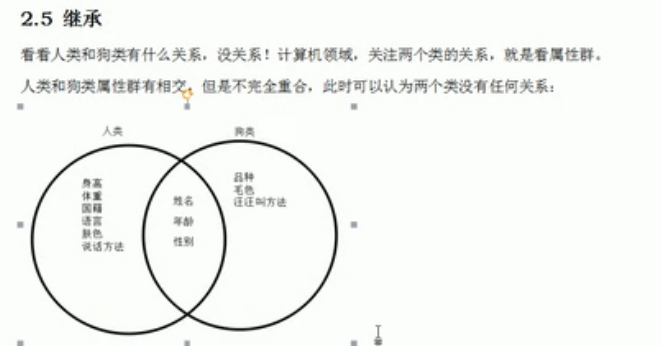
如果函数的prototype在对象的原型链上，此时返回true，反之返回false

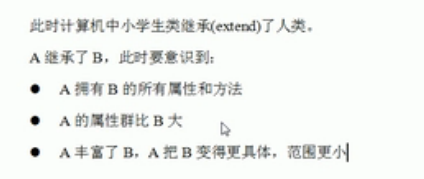
换句话说，如果对象的原型链上，恰好有某个元素是函数的prototype，此时返回true，返回false。

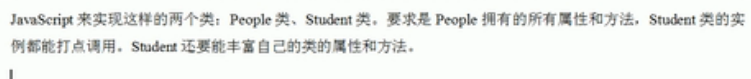
因为Object.prototype是所有对象原型链的终点，所以任何的对象都会有：

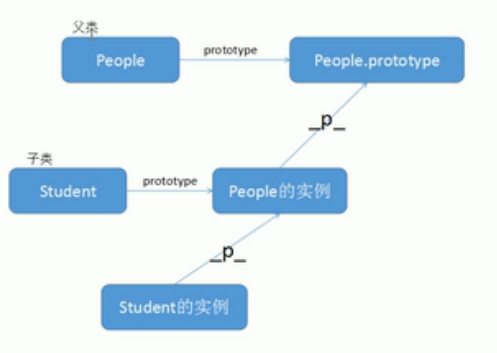
对象 instanceof Object 返会true.

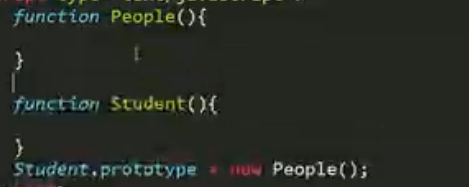
总结，在js中，我们没有可靠的办法来探寻一个元素是不是一个构造函数的实例。











function People(){

this.sayHello=function(){

return 'People say hello in People';

}

People.prototype.sayHello=function(){

return 'People say hello in People.prototype';

}

function Student(){

this.sayHello=function(){

return 'Student say hello in Student';

}

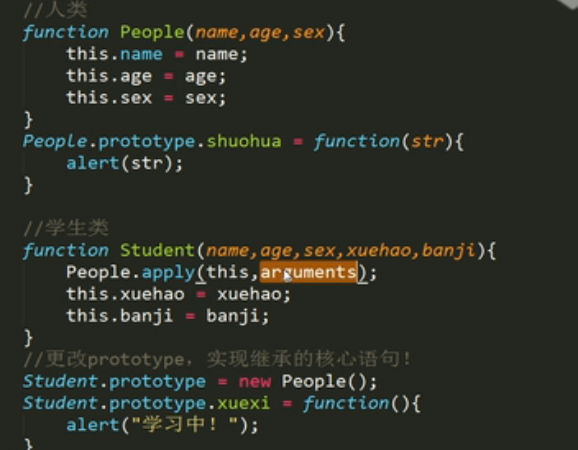
}

Student.prototype=new People(); //prototype指向父类的实例，就是将student对象的构造函数constructor改变了

var s=new Student();

console.log(s);

console.log(s.sayHello());

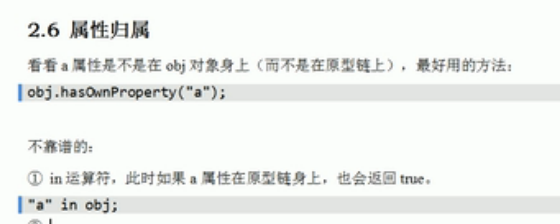




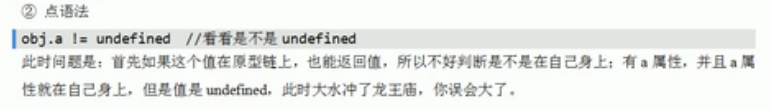
Js很少用继承



20原型对象属性



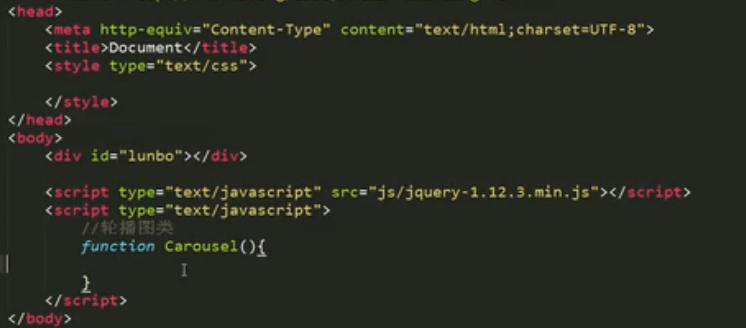
hasOwnProperty是检查属性是否在对象自身上而不是在原型链上；而in检测对象否包含属性

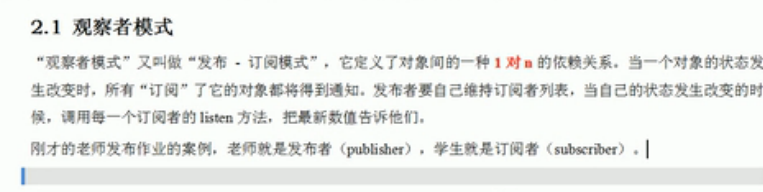


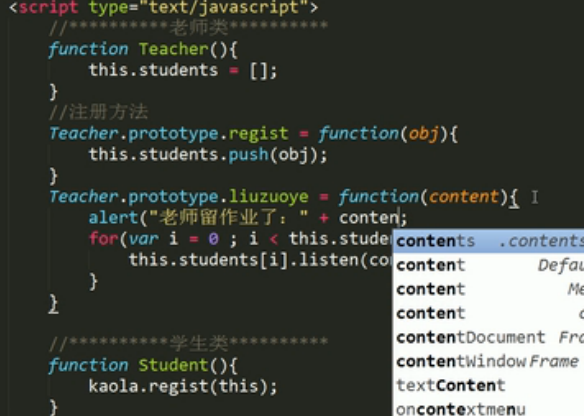


21原型对象属性

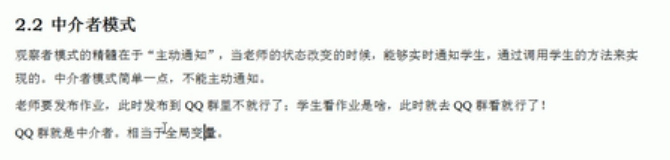








23中介者模式



//===============20171010