点击

```
        //6.使用方法hasOwnProperty,属性只有存在于实例中才会返回true
        console.log("person1.has Property name:",person1.hasOwnProperty("name"))
//true
        console.log("person1.has Property age:",person1.hasOwnProperty("age"))
//false
        //7.in操作符 前面提到hasOwnProperty方法可用于检测属性是否是实例属性，in则会遍历所有属性，不管是实例上的，还是原型上的
        //8.Object.keys() 此方法可以获取对象的所有可枚举的属性的名字
        console.log("keys:",Object.keys(person1)) //keys: ["name"]
        /*
         Person.prototype就是原型对象，也就是实例person1和person2的原型。原型对象也是对象，所以它也有proto属性，连接它的原型，
         原型对象Person.prototype的原型就是Object.prototype这个大boss，所有原型对象都是Object构造函数生成的
        */
        /*
        正是因为所有的原型最终都会指向Object.prototype，所以对象的很多方法其实都是继承于此，比如toString()、valueOf()，前面用到的hasOwnProperty，甚至是.constructor、proto
        */

//console.log("Person.__proto__====Object.prototype",Person.__proto__===Object.prototype) //false
        console.log("Person.__proto__:",Person.__proto__)
        console.log("Person.__proto__:",Person.__proto__.constructor)
        console.log("Object.prototype:",Object.prototype)
//constructor,hasOwnProperty,toString,toLocaleString,valueOf,isPrototypeOf
        console.log("Object:",Object)
        //Object.prototype有原型吗？
        console.log("Object.prototype没有:",Object.prototype.__proto__) //null，所以它就是前面所提到的尽头
    }
```