1. 在ES6之前，对象的属性名只能是字符串，甚至可以是空字符串的任意字符串。对象的值可以是任意JS值，或者是getter/setter。
2. 属性除了key，value之外，还有一些“属性特性”。  
   可写（writable）：表明是否可设置该属性的值  
   可枚举（enumerable）：表明是否可以通过for/in遍历  
   可配置（configurable）：表明是否可以删除以及是否可以修改其他特性。  
   ES5之前，所有的对象都是可枚举、可写、可配置的。如对象直面量{“o”：1}，属性“o”是可枚举、可写、可配置。
3. 除了包含属性外，对象还有以下3个特性：  
   原型（prototype）  
   类（class）  
   扩展标记（extensible flag），指明了是否可以向对象添加新的属性
4. 创建对象
5. 对象直接量，是一个表达式，这个表达式的每次运算都会创建并初始化一个新的对象。也就是说，如果在一个循环中使用对象直接量，那么它将创建很多对象。
6. New（）
7. 原型：  
   每一个对象都从原型继承属性。  
   Object.create（）是静态函数，是Object构造函数的一个属性方法。
8. 给对象属性赋值  
   属性复制首先检查原型链，判断是否允许赋值。如果o继承一个只读属性x，那么就不允许赋值。如果允许赋值，那么它也总是会在原始对象o上对已有属性赋值，而不会去修改原型链。属性赋值要么失败，要么创建一个新属性，要么在原始对象中设置。
9. 查询一个不存在的属性不会报错，只是会返回undefined。而查询一个不存的对象的属性会报错，就相当于查询undefined和null的属性会报类型错误异常。在ES5严格模式中，任何失败的属性设置都会抛出一个类型错误异常。
10. delete：  
    delete只能删除自由属性，不能删除继承的属性。这是肯定的，因为如果允许删除继承属性的话，那么会改变其他对象。除此之外，delete只是断开属性与属猪对象的连接。  
    delete成功删除或者没有副作用的时候会返回true，但是当delete删除一个不可配置的属性的时候，ES5的非严格模式返回false，严格模式会抛出类型错误异常。
11. 检测属性  
    任何对象都从Object.prototype那里继承了一个方法，hasOwnProperty（）检测是否是对象的自有属性。  
    任何对象都从Object.prototype那里继承了一个方法，propertyIsEnumerable()检测改属性同时是自有属性，同时可枚举。  
    in可以检测对象中有没有这个属性，如“x” in o，有的话返回true，另外使用o[“x”]!==undefined也可以
12. getter和setter  
    又称“存取器属性”。
13. 属性的特性  
    ES3中对象所有的属性都是可写、可枚举、可配置的，且无法修改。  
    ES5中可以对这些进行设置。  
    静态方法：Object.defineProperty();  
    例如Object.defineProperty(o,”x”,{value:1,writable:true,enumerable:false,configurable:true})。没有设置的默认为undefined或者false。  
    还有一个Object.defineProperties();当然，这两个函数是ES5的，如果操作失败，会抛出类型错误异常，而不是简单地返回false。  
    对应的返回属性描述符对象的API是Object.getOwnPropertyDecriptor();  
    如果数据属性是不可配置的，不能将它的可写性从false修改为true，但是却可以从true修改为false。
14. 对象的原型属性  
    ES5可通过静态函数Object.getPrototypeOf()获得对象的原型。这个方法无论你是通过Object.create（）创建还是字面量还是new都会返回正确的。  
    ES3没有这个函数，所以通常使用o.constructor.prototype来检测，但是这个不保险，因为有时候，你会忘了设置对象的constructor属性。除此之外，使用Object.create（）创建的对象使用此方法会不正确。Object.create（）创建的对象它的内置的\_proto\_属性是正确的，所以你使用Object.getPrototypeOf()永远正确，但是你用constructor的时候，他会指向最开始的创建函数，所以会不正确。  
    对应的检查函数，所有对象都从Object.proytotype上继承了一个isPrototypeOf的方法，可以使用p. isPrototypeOf（o）来检查p是不是o的原型

类属性      
function classof(o){

      if(o===null) return "Null";

      if(o===undefined) return "Undefined";

      return Object.prototype.toString.call(o).slice(8,-1);

    }

1. instanceOf操作符实际上就是在检测原型链，左操作数是一个对象，右操作数是一个构造函数。
2. 判断对象是否可扩展 Object.isExtensible()  
   使对象不可扩展Object.preventExtensions(),不可挽回的。  
   Object.seal(),使对象不可扩展之外，使得所有的自由属性不可配置。  
   Object.isSealed()检测对象是否是封闭的。  
   Object.freeze()使对象不可扩展之外，使得所有的自由属性不可配置，还将所有的数据属性设置为只读的。但是不影响setter。  
   Object.isFrozen()检测对象是否被冻结。