所有的方法创建的都是浅不变形，也就是说，它们只会影响目标对象和它的直接属性。如果目标对象引用了其他对象（数组、对象、函数，等），

其他对象的内容不受影响，仍然是可变的。

结合 writable:false 和 configurable:false 就可以创建一个真正的常量属性（不可修改、重定义或者删除）。

如果你想禁止一个对象添加新属性并且保留已有属性，可以使用 Object.preventExtensions(..)。

Object.seal(..) 会创建一个“密封”的对象，这个方法实际上会在一个现有对象上调用Object.preventExtensions(..)

并把所有现有属性标记为 configurable:false 。

所以，密封之后不仅不能添加新属性，也不能重新配置或者删除任何现有属性（虽然可以修改属性的值）。

Object.freeze(..) 会创建一个冻结对象，这个方法实际上会在一个现有对象上调用Object.seal(..)

并把所有“数据访问”属性标记为 writable:false ，这样就无法修改它们的值。这个方法是你可以应用在对象上的级别最高的不可变性，

它会禁止对于对象本身及其任意直接属性的修改。

给一个属性定义 getter、setter 或者两者都有时，这个属性会被定义为“访问描述符”。

in 操作符会检查属性是否在对象及其 [[Prototype]] 原型链中。

for..in 循环可以用来遍历对象的可枚举属性列表。

多态的另一个方面是在继承链的不同层次中一个方法名可以被多次定义，当调用方法时会自动选择合适的定义。

对于默认的[[Get]]操作来说，如果无法在对象本身上找到需要的属性，就会继续访问对象的[[Protorype]]链。

所有的函数默认都会拥有一个名为protorype的公有并且不可枚举的属性。

JS默认不会复制对象属性。JS会在两个对象之间创建一个关联，这样一个对象可以通过委托访问另一个对象的属性和函数。

默认情况下，对象并不会像差异继承暗示的那样通过复制生成。因此，差异继承也不适合用来描述 JavaScript 的 [[Prototype]] 机制。

constructor并不表示被构造，.constructor并不是一个不可变属性，它是不可枚举的，但它的值是可被修改的。

委托行为意味着某些对象（ XYZ ）在找不到属性或者方法引用时会把这个请求委托给另一

个对象（Task）。禁止互相委托。

在委托设计模式中，除了建议使用不相同并且更具描述性的方法名之外，还要通过对象关联避免丑陋的显式伪多态调用（ Widget.call 和 Widget.prototype.render.call ），代之以简单的相对委托调用 this.init(..) 和 this.insert(..) 。

ES6 的 class 语法可以简洁地定义类方法。唯一的区别是对象的字面形式仍然需要使用“,”来分隔元素，而 class 语法不需要。这个区别对于整体的设计来说无关紧要。

匿名函数没有 name 标识符，这会导致：1. 调试栈更难追踪；2. 自我引用（递归、事件（解除）绑定，等等）更难；3. 代码（稍微）更难理解。