JS中的原始值是不可更改的；

原始值的比较是值的比较，只有在它们值相等的时候它们才相等；

对象的比较时引用的比较，两个单独的对象永不相等，当且仅当它们引用同一个基对象时才相等；

通常称对象为引用类型；

var a = [];

var b = a;

如果将一个对象赋值给另一个对象，赋的是引用，即a和b指向的是同一个引用，对象并没有复制一次；

如果想得到一个对象的副本，那么就要显示的将原对象中的变量、属性、方法复制到另一个对象中；

**转换：**

注意：一个值可以转换为另一个值，但并不意味着这两个值相等；比如，undefined会在某些情况下转换为false，但是undefined不等于false；

**显示转换**：最简单的方法就是使用包装对象；

除了undefined和null之外的任何值都具有toString（）方法；

JS中的某些运算符会做**隐式类型转换**：

如果“+”有一个操作数是字符串，那么另一个会被转换为字符串，a + ""等价于String（a）；

一元“+”会把操作数转换为数字，+a等价于Number（a）；

“！”把操作数转换为布尔值然后取反；

转换为数字的三种方法：

Number（）--只能基于十进制转换；

     toFixed（）--参数指定小数位数；

     toExponential（）--科学计数法，参数是要保留的小数位数；

     toPrecision（）--科学计数法，参数是有效数字位数；

parseInt（）--可接受两个参数，第一个是要转换的变量，第二个是进制；

                      若第一个字符不是数字，则转换为NaN；

parseFloat（）--同上

（1）原始值-->原始值

简单；

转换为数字：以数字表示的字符串可以直接转换为数字，并且允许在开始和结尾处带有空格；

（2）原始值-->对象

原始值调用String（）、Number（）、Boolean（）构造函数，转换为各自的包装类型；

undefined和null除外，他们用在期望是一个对象的地方会造成一个类型错误，不会执行正常的转换；

（3）对象-->原始值

所有的对象继承了两个转换方法：toString（）和valueOf（）

toString（）--返回一个反映这个对象的字符串；

valueOf（）--如果存在任意原始值，他就默认将对象转换为表示他的原始值，但是对象是复合值，大多数对象无法真正的表示为一个原始值，因此，默认的valueOf（）简单的返回对象本身；

对象转换为字符串：

首先调用toString（）方法（如果有的话），如果返回的是一个原始值，则将这个原始值转换为字符串；

如果没有toString（），或者返回的不是一个原始值，那么就调用valueOf（）（如果有的话），如果返回一个原始值，那么就将它转换为字符串；

如果无法从toString（）或valueOf（）获得一个原始值，就会抛出一个类型错误的异常；

对象转换为数字：

首先调用valueOf（），如果返回一个原始值，就将它转换为数字；

如果没有valueOf（），或者返回的不是原始值，那么就调用toString（），如果返回了一个原始值，将它转换为数字；

如果不能从valueOf（）或toString（）获得一个原始值，就抛出错误；

对于所有非日期对象来说，对象到原始值的转换基本上都是到数字的转换（首先调用valueOf（））；

日期对象使用对象到字符串的转换模式，并且通过valueOf（）或toString（）得到的原始值会被直接使用，不会被强制转换为数字或字符串（+、==、！=、关系运算符的情况下） ；

如果将一个对象和原始值比较，转换会遵照对象-->原始值的转换方式进行；