Knockout Refactoring & Knockout for XAML Developers

# Lifecycle

http://www.pluralsight.com/training/player?author=rob-conery&name=play-by-play-knockout-refactor-m1-intro&mode=live&clip=0&course=play-by-play-knockout-refactor

http://github.com/tekpub/knockout-cart

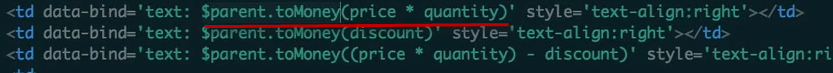
### Project

Knockout-cart

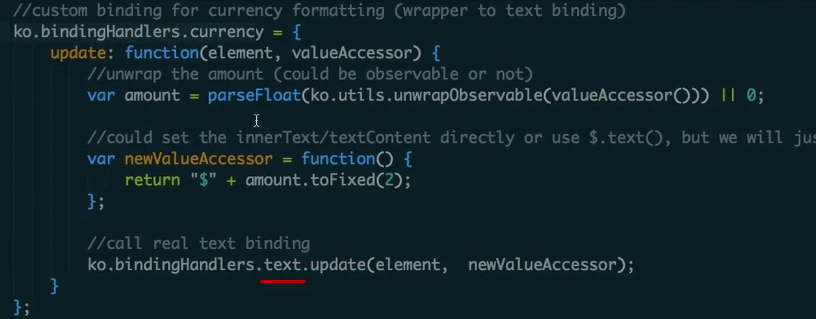
### 变量默认值

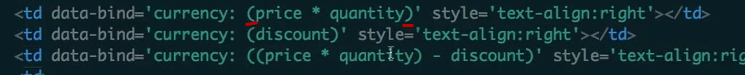


### 用Custom Binding处理显示—ko.bindingHandlers

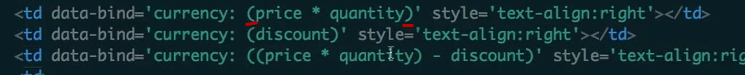


加上一个Currency的CustomBinding

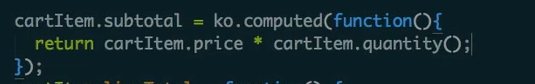


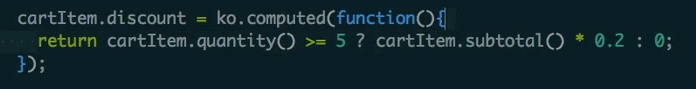


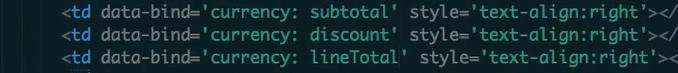
### 用ko.computed处理计算



原始版本直接绑定了属性的计算，加括号，要换掉

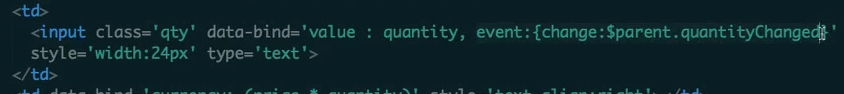




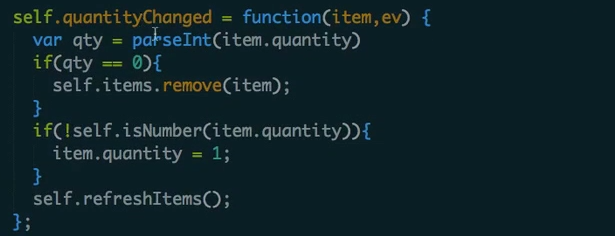


### 改变变量的读写行为—ko.observable().fn read write

原始版本上每次Change的时候，都要做quantityChanged



quantityChanged的作用是对输入做检验，如果购买量是0，就移除这个物品，如果输入对就刷新购买列表

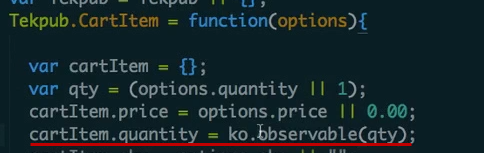


刷新的方式是先删除所有item，然后再重新给items设值，这样items就会刷新了

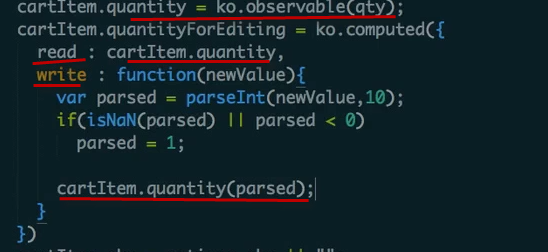


需求是在给变量写值的时候，会引发一系列其他操作

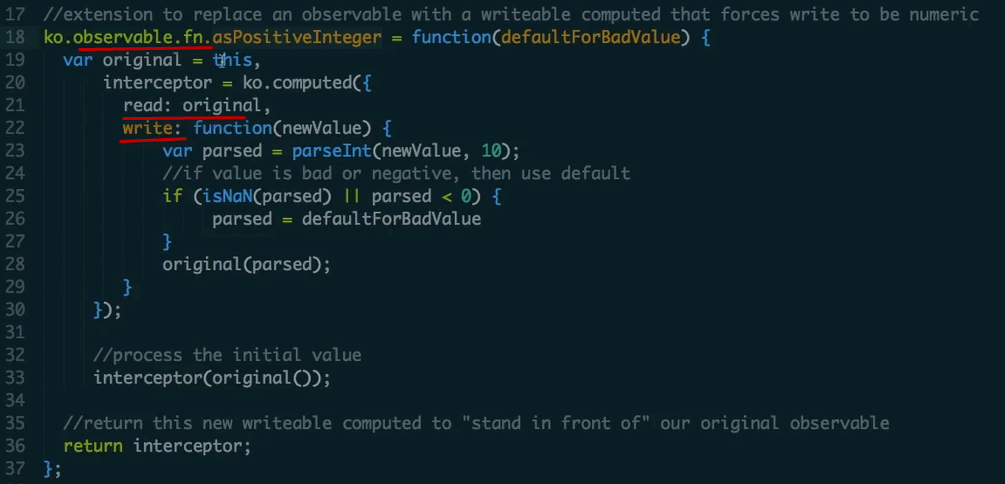
解决方法：首先把quantity做成了Observable



然后加一个新的computed变量



**最好的方法是给observable加上一个函数**

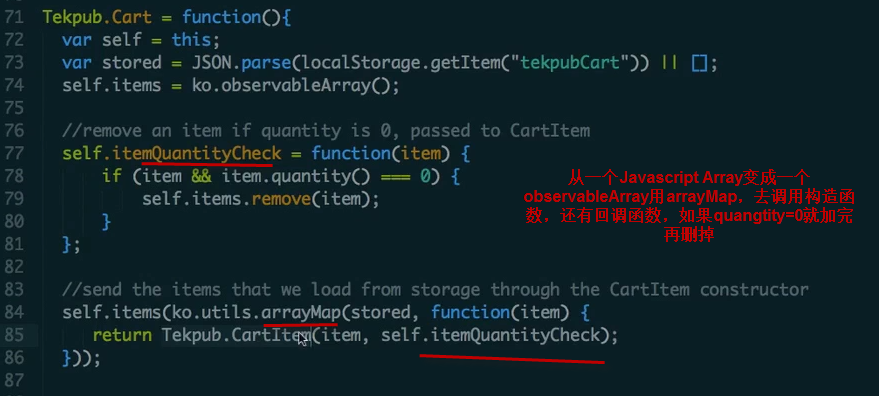


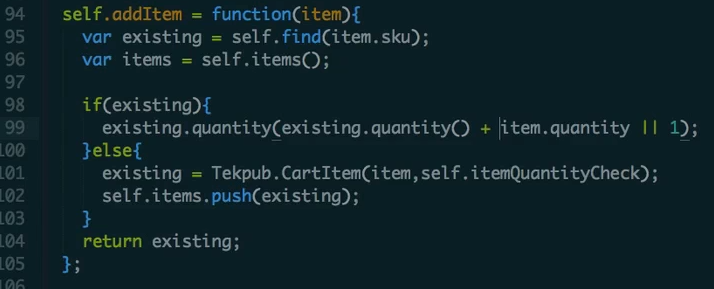
然后就不需要新加一个变量了，直接在原变量上去用

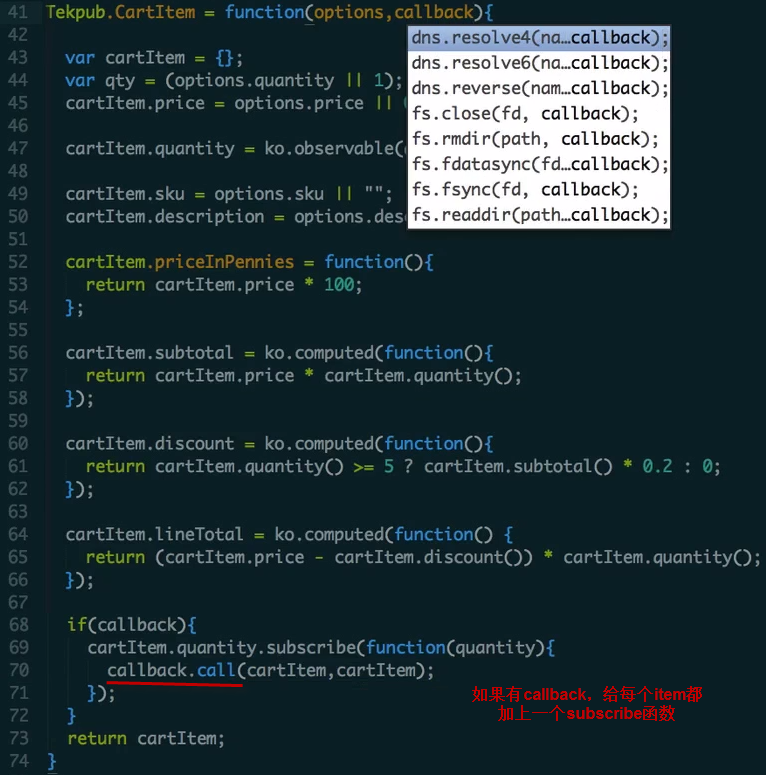


### Observable Array的构造—ko.utils.arrayMap

这里需要的是在一个array中，新增一个对象时要调用构造函数，要判断值合不合法，其实就是自己手写item的构造函数，然后在创建时mapping到构造函数。Add时也调用构造函数。合不合法的函数在构造时被加到每个item作为其成员函数，在之后调用。







### 保存编辑历史—localStorage

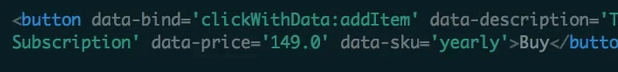
每次编辑了购物车列表，之前的版本都会被保存



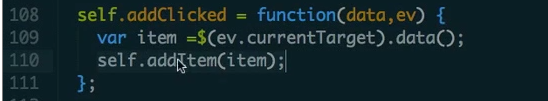
在有些浏览器里要先removeItem才能在set

### 在binding里用html的data-属性当参数

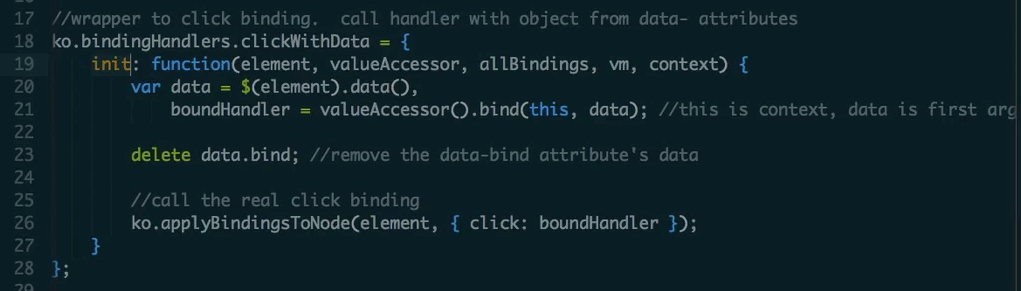
最后的效果是点击之后，会被data-属性组成Object当参数传给addItem函数。ko默认会把绑定到html上对象当参数。



原来为了使用data属性，要加一个函数，手动去获得data()，然后调用addItem

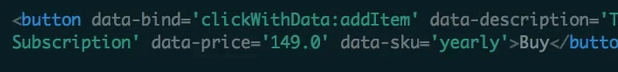


先在加一个新的binding，在binding里获得data，然后将data bind到valueAccessor上。



然后html直接用AddItem即可，会传data-sku进去

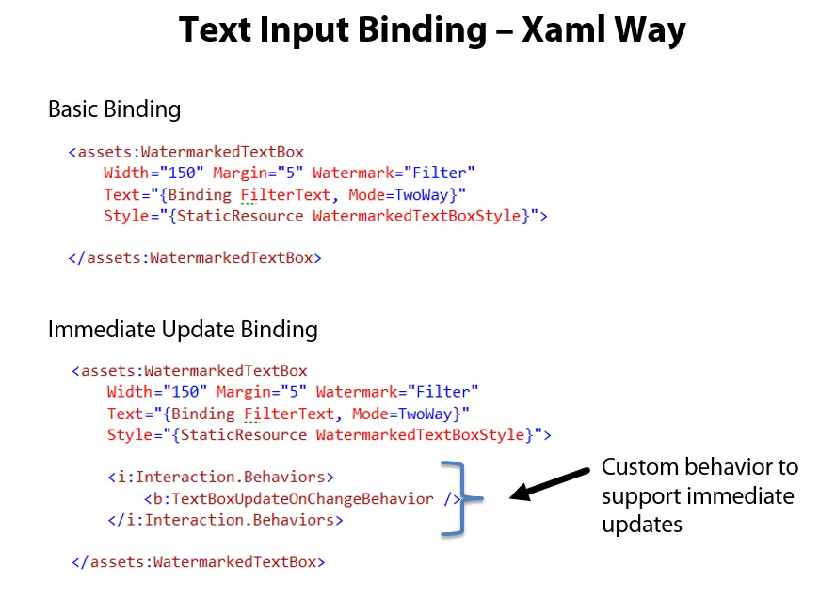


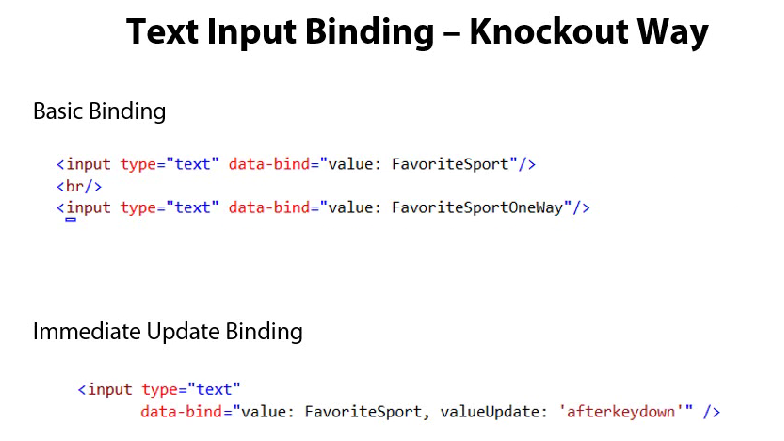


# Input Controls

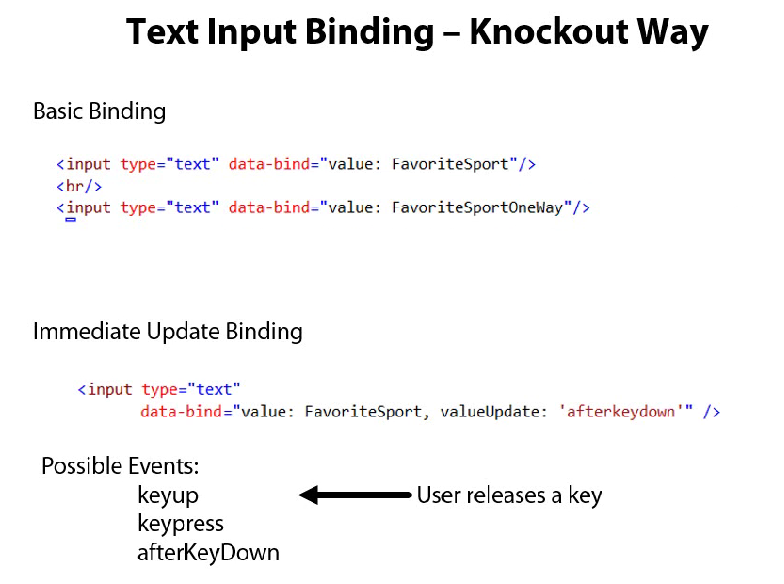
http://www.pluralsight.com/courses/knockout-xaml-developer

### Text Input Binding——Value Binder

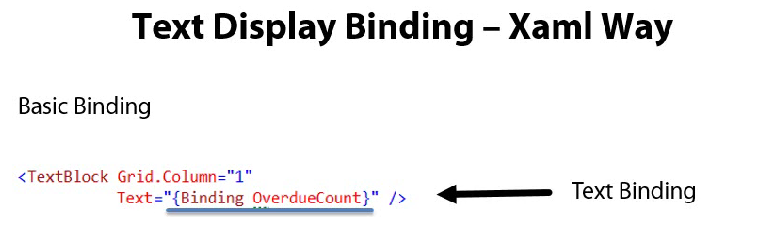


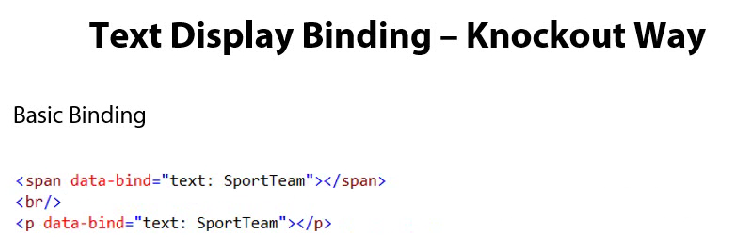


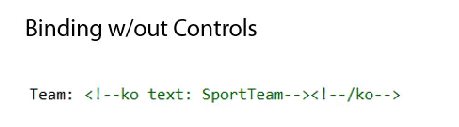
### Text Input Binding——Value Binder



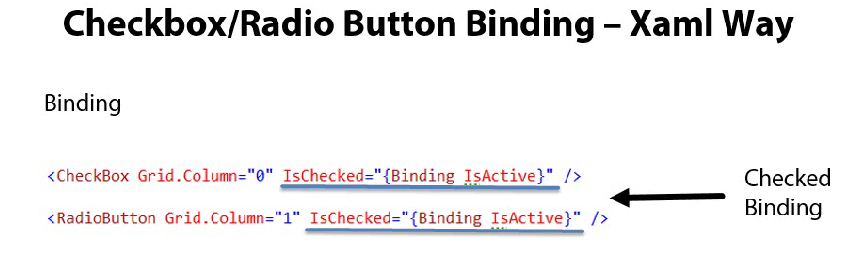
### Text Display Controls——Text Binder

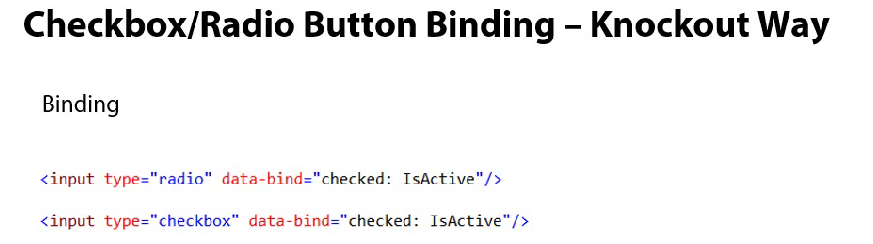




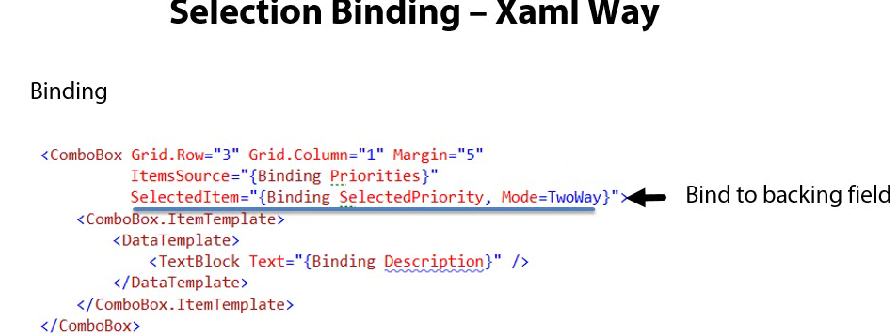


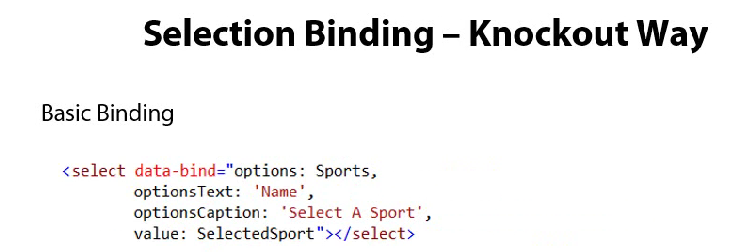
### Boolean Controls——checked Binder





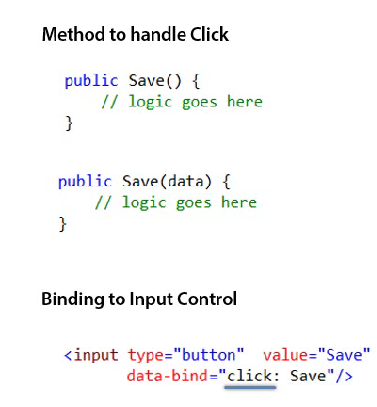
### Selection Binding——Text Binder

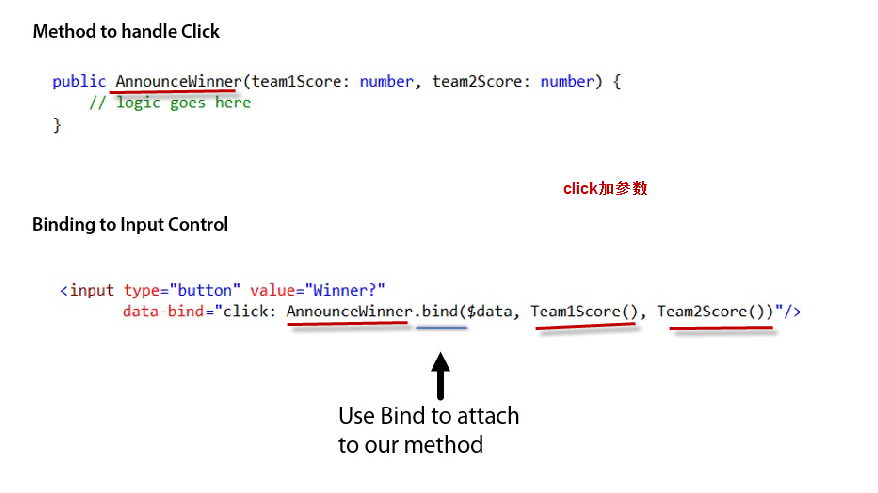




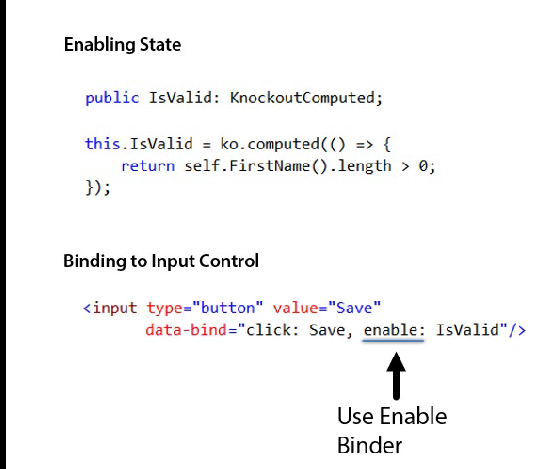
# Events

### Click

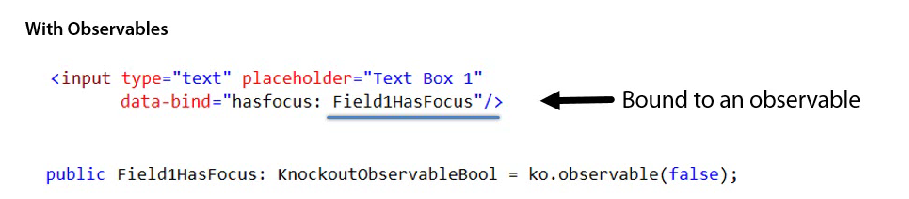




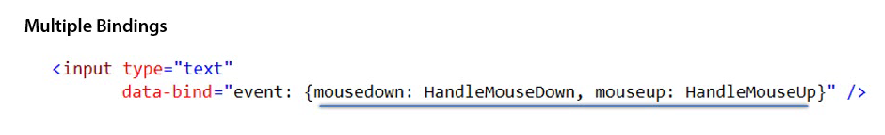
### Click with Enable State



### Focus



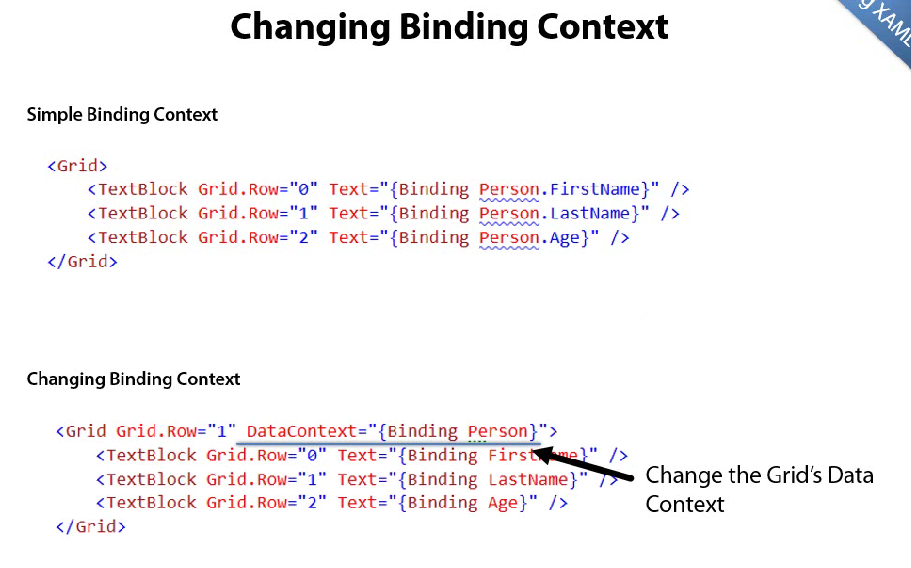
### DOM Events

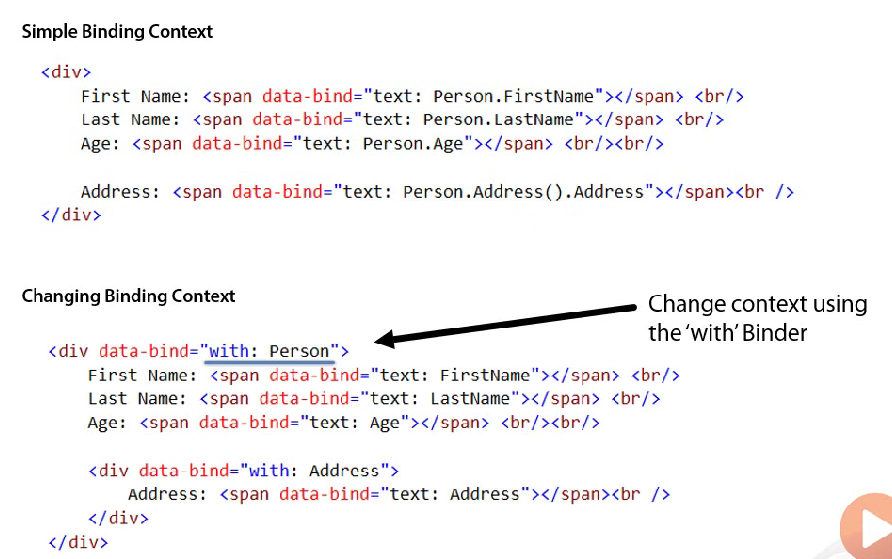


# Binding Context

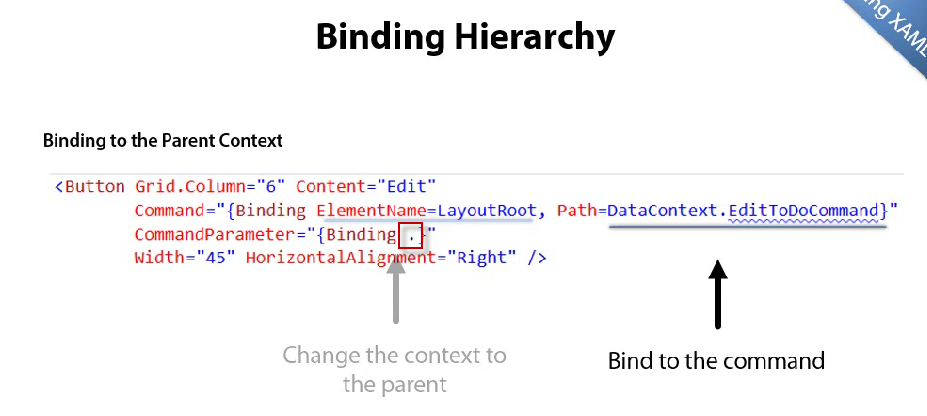
### With

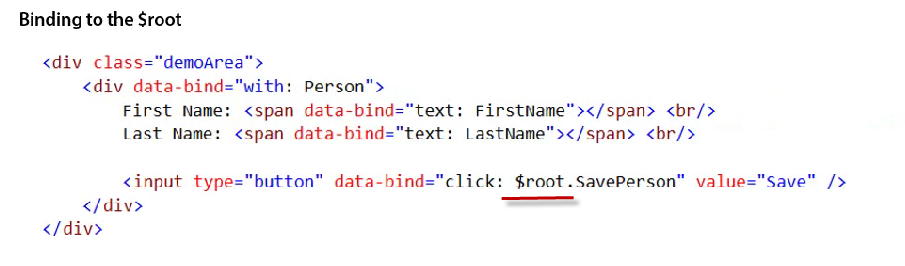


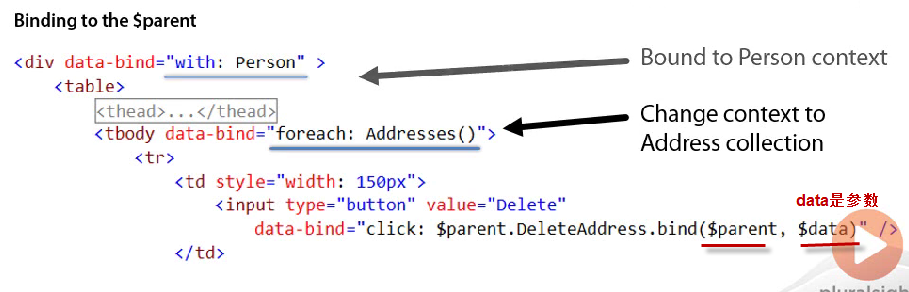




### Root Parent Data

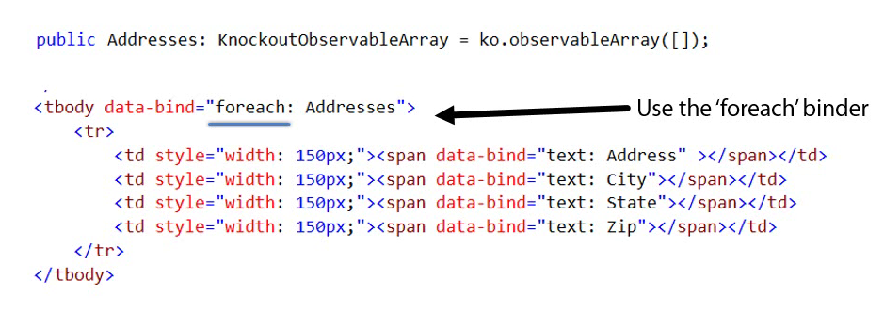




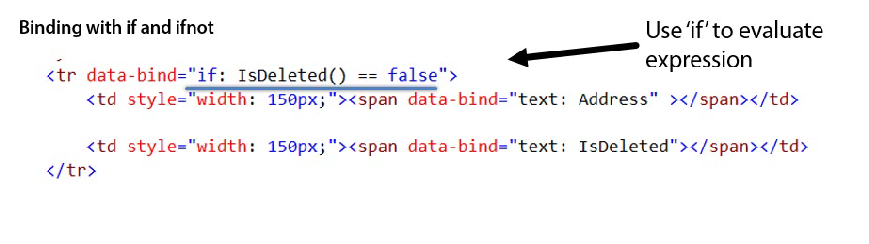


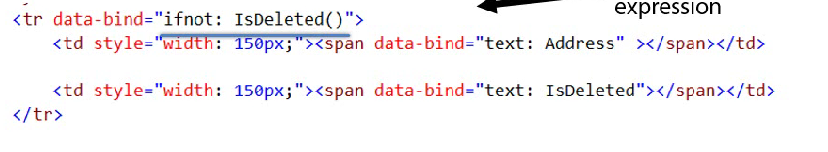
### Foreach





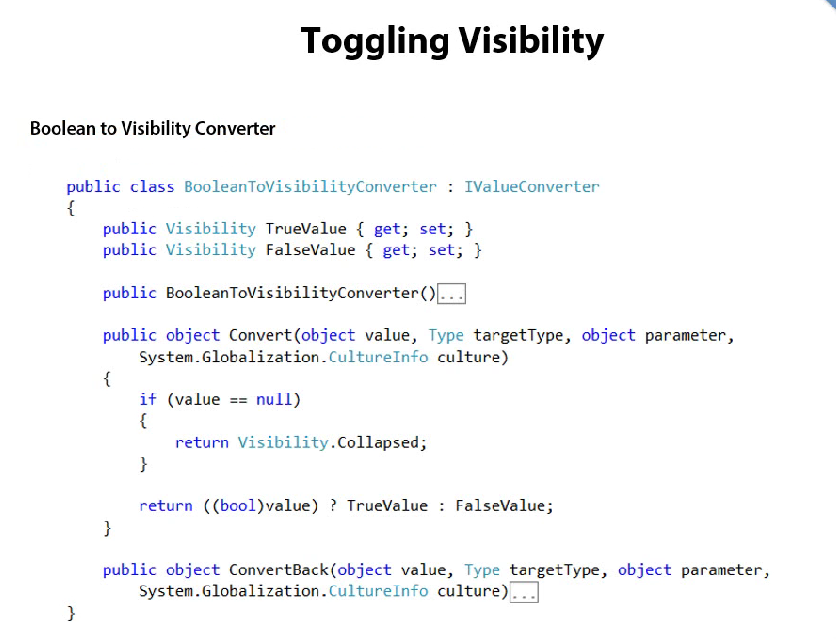
### If ifnot

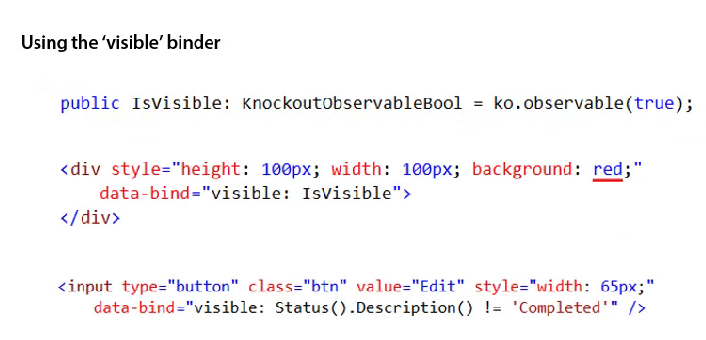




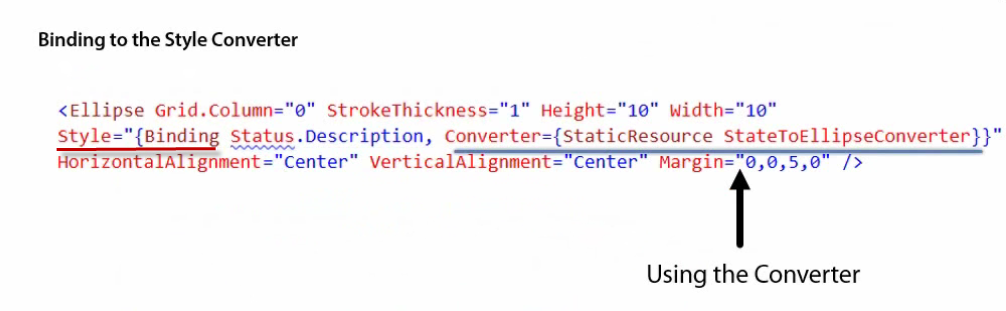
### Visible



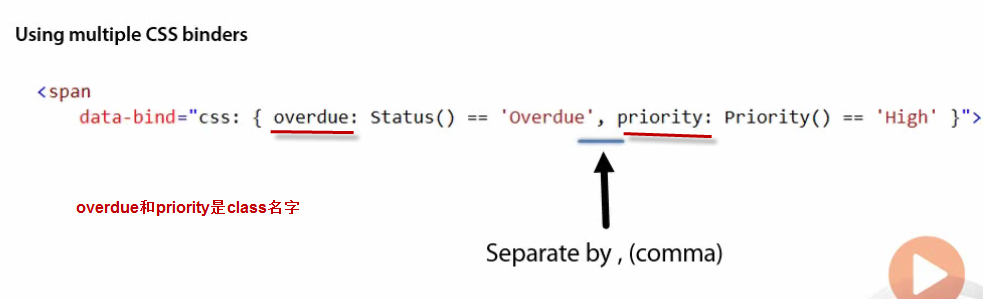




### Css——改Class







### Style

