

中文网<http://www.aizhengli.com/knockoutjs>

http://www.javaarm.com/file/www/javascript/fwk/knockoutjs/site-back/documentation\_cn/binding-context.html

Knockout是javascript一个类库

Model-View-View Model (MVVM)是一种创建用户界面的设计模式实现视图与业务逻辑分离。通过它只要将UI界面分成以下3个部分

https://blog.csdn.net/binglan520/article/details/71107807

Model: 用于**存储你应用程序数据**（Model 的数据模型，和我们的业务需求或者说业务实体（Entity）是一一映射关系）, 当使用KO的时候，通常是使用Ajax向服务器请求数据来读写这个数据模型

View Model: 用于描述数据内容和页面操作的数据模型, 使用KO时，你的View Model（数据模型）是纯粹的不包含HTML知识的JavaScript对象

View，代表View Model状态的一个可见、互动的UI界面。它主要用于显示View Model的数据信息、发送用户命令（例如，当用户点击按钮）以及在View Model发生变化时保持自动更新。

使用KO时，你的View层主要就是简单的将HTML文档声明式的绑定到View Model，将它们关联起来。另外，你也可以利用模版从View Model获取数据动态生成HTML。

使用KO创建一个View Model，仅仅只需要声明一个JavaScript对象

## Knockout是建立在以下3个核心功能之上的：属性监控与依赖跟踪 ；声明式绑定 ；模版机制

ko.applyBindings(myViewModel,document.getElementById('someElementId'))

applyBindings的参数说明

第一个参数是你想激活KO时用于声明式绑定的ViewModel对象；

第二个参数（可选），设置要使用data-bind属性的HTML元素或容器；

ko.applyBindings(myViewModel, document.getElementById('checkId'))

它限制了只有ID为checkId的元素才能激活使用KO功能，当你在一个页面中声明了多个View Model来绑定不同的界面区域时，这样限制是很有好处的

Observables 方法

KO最重要的一个属性就是当View Model发生变化时能够自动的更新UI界面

KO是如何能够知道你的View Model已经发生改变了呢？答案是：你需要将你的model属性声明成observable的, 因为它是非常特殊的JavaScript 对象，能够通知用户它的改变以及自动检测依赖关系

并不是所有的浏览器都支持JavaScript的getters and setters (比如IE)，所以，为了兼容，ko.observable监控的对象都是方法。

1、读取当前监控的属性值，只需要直接调用observable（不需要参数），在这个例子当中，调用myViewModel.personName()将会返回'Bob'，调用myViewModel.personAge() 将会返回'123'

2、写一个新值到监控属性上，调用observable方法，将要写的值作为一个参数传入即可。例如，调用myViewModel.personName('Mary') 将会把名称改变成 'Mary'。

3、一次性改变Model对象监控的多个属性值，你可以使用链式方法。例如：myViewModel.personName('Mary').personAge(50) 将会把名称改变成'Mary'将年龄设置为 50.

监控属性最重要的一点就是可以随时监控，也就是说，其他代码可以告诉它哪些是需要通知发生变化的。这就是为什么KO会有如此多的内置绑定语法。所以，当你在页面中编写data-bind="text: personName"，text 会绑定注册到自身，当personName发生变化时，它能够立即得到通知。

当你通过调用 myViewModel.personName('Mary') 将名称的值改变成为'Mary'时，text绑定会自动更新新值到其对应的DOM对象元素上,这就是为什么改变数据模型能够自动刷新View页面。

监控属性数组（Observables Arrays ）: 如果你想发现并响应一个对象的改变，就应该用监控属性（observables）。如果你想发现并响应一个集合的变化，就该用监控属性数组（observableArray）。监控属性数组在显示或编辑多个值以及对界面的一部分重复显示或隐藏（如添加删除条目）等多种情况下 都是很有用的。

如果我们想要使用Knockoutjs表示多个值的话我们可以使用数组监控(Observable Arrays)来表示

属性数组的赋值

Push方法

初始化赋值

var myObservableArray = ko.observableArray();

myObservableArray.push('Some value');myObservableArray.push('Some value');

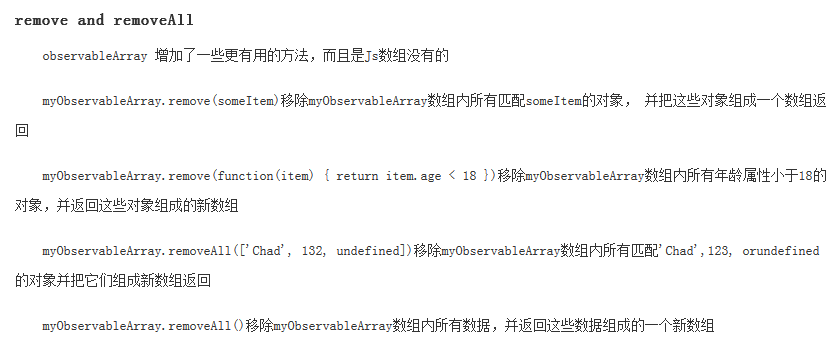
var anotherObservableArray = ko.observableArray([

{name: "Bungle", type: "Bear" },

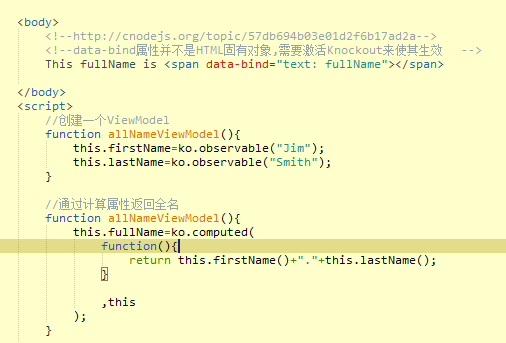
{name: "George", type: "Hippo" },

{name: "Zippy", type: "Unknown" }

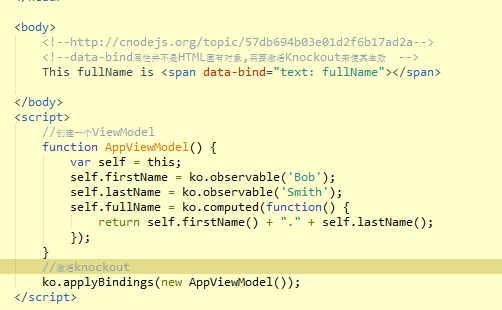
]);



Computed Observable：计算属性



当你需要全程跟踪this时，下面的写法是一种很流行的惯例：如果你将你的 viewmodel's结构this作为一个变量复制一份(传统称之为self)，在以后你可以使用self来代表viewmodel而不必担心它重定义或指别的东西



applyBinding & Observables

## 数据绑定的形式：One-Time，One-Way，Two-way

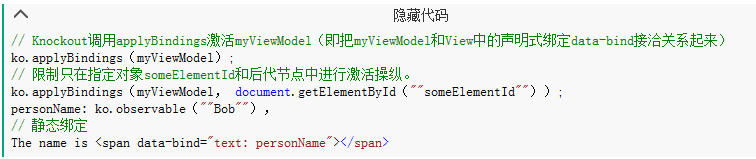
## One-Time：绑定模式的意思即为从viewModel绑定至UI这一层只进行一次绑定，程序不会继续追踪数据的在两者中任何一方的变化，这种绑定方式很使用于报表数据，数据仅仅会加载一次

## One-Way绑定模式即为单向绑定，即object-UI的绑定，只有当viewModel中数据发生了变化，UI中的数据也将会随之发生变化，反之不然

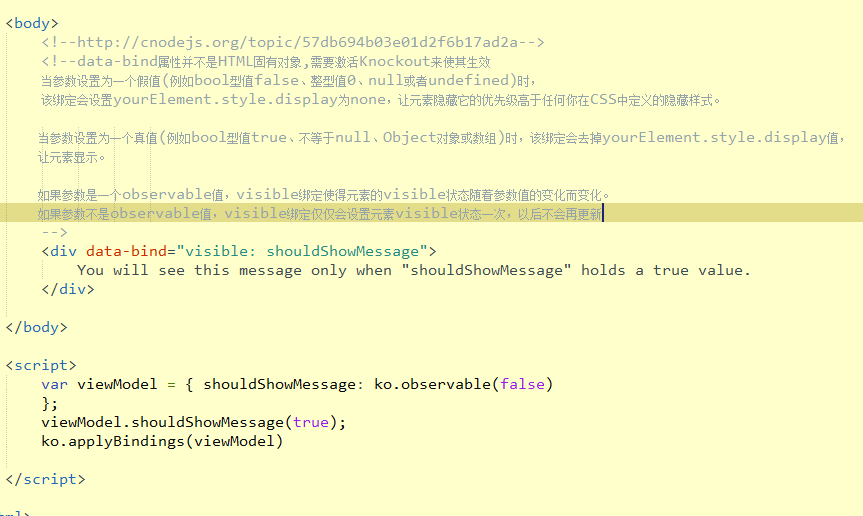
## Two-Way绑定模式为双向绑定，无论数据在Object或者是UI中发生变化，应用程序将会更新另一方，这是最为灵活的绑定方式，同时代价也是最大的。数据绑定只是作为元素的自定义属性写上标签内，并不能决定它是何种绑定

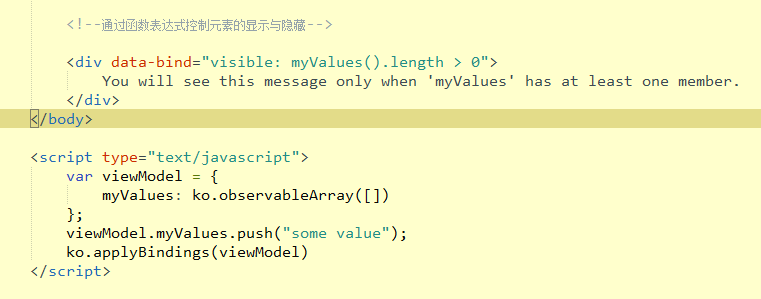
## 如果值是通过ko.observable定义的说明是双向绑定，否则为One-Time绑定，在knockout不存在单向绑定

applyBindings



Visible绑定：通过绑定一个值来确定DOM元素显示或隐藏





text绑定主要是让DOM元素显示参数值

通常<span>和<em>这样的元素上非常有用，而实际上你可以绑定任何元素



<span data-bind="text: price() >50 ? 'expensive' : 'affordable'"></span>

text绑定HTML代码

viewModel.myMessage("<i>Hello, world!</i>")输出的结果按html代码来解析

text 在不使用多余的元素的情况下通过text绑定来设置文本内容

<select data-bind="foreach: items"> <option>Item<span data-bind="text: name"></span></option></select>

无效绑定：因为<option>中不容许出现其他标签

为了解决这个问题，你可以使用容器语法，它基于一个注释元素

<select data-bind="foreach: items">

<option>Item<!--ko text: name--><!--/ko--></option>

</select>

<!--ko-->和<!--/ko-->注释标记作为起始和结束符，定义一个“虚拟元素”，里面包含了标记，Knockout能够识别这种虚拟元素语法和绑定作为你需要的容器元素而存在

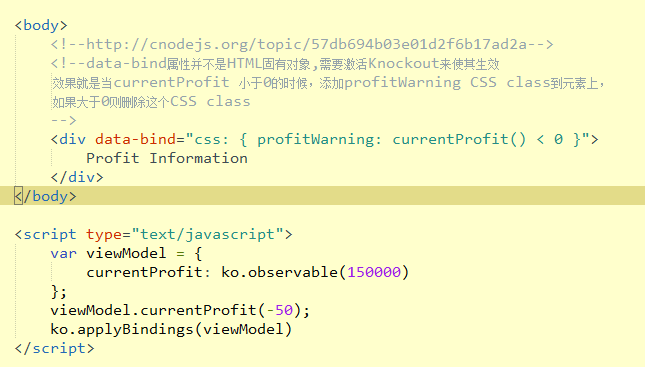
HTML绑定



CSS类名绑定

ko内置的基础绑定之一 。 用于控制DOM元素的class属性( 会根据所绑定的bool值（或者可以返回bool值的js表达式），给元素 移除/添加 对应的class名称 如果与observable或者computed属性绑定，则会产生双向绑定效果。

css绑定是添加或删除一个或多个CSS class到DOM元素上



参数说明：该参数是一个JavaScript对象，属性是你的CSS class名称，值是true或false，用来决定是否应该使用这个CSS class，你可以一次设置多个CSS class

<div data-bind="css: { profitWarning: currentProfit() < 0, majorHighlight: isSevere }">

非布尔值会被解析成布尔值。例如， 0和null被解析成false，21和非null对象被解析成true。

如果参数是监控属性observable的，那随着值的变化将会自动添加或者删除该元素上的CSS class。如果不是，那CSS class将会只添加或者删除一次并且以后不在更新

注意：应用的CSS class的名字不是合法的JavaScript变量命名

如下写法是错误的：<div data-bind="css: { my-class: someValue }">...</div>

但可以通过加‘’的方法去解决

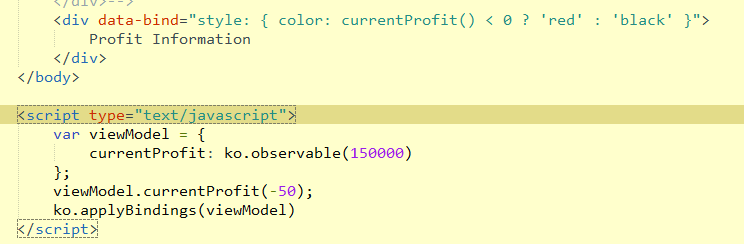
<div data-bind="css: { 'my-class': someValue }">...</div>

Style属性绑定

同样是用于控制样式，不过该绑定用于控制样式的细节：

控制DOM元素的Style属性，比如color,width等 会根据所绑定的bool值（或者可以返回bool值的js表达式），给元素 移除/添加 对应的Style属性 如果与observable或者computed属性绑定，则会产生双向绑定效果

style绑定是添加或删除一个或多个DOM元素上的style值

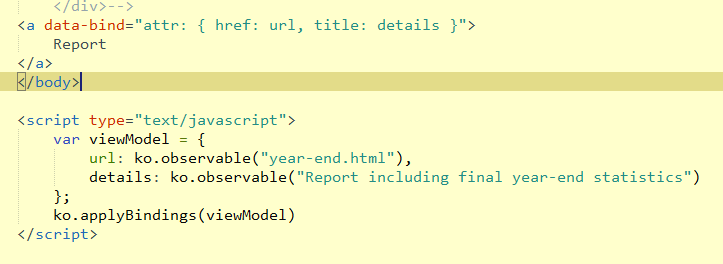


该参数是一个JavaScript对象，属性是你的style的名称，值是该style需要应用的值。

多个值的设定

<div data-bind="style: { color: currentProfit() < 0 ? 'red' : 'black', fontWeight: isSevere() ? 'bold' : '' }"> </div>

attr属性绑定：给DOM元素添加自定义属性，或绑定到已有的属性，attr 绑定提供了一种方式可以设置DOM元素的任何属性值。你可以设置img的src属性，连接的href属性。使用绑定，当模型属性改变的时候，它会自动更新



该参数是一个JavaScript对象，属性是你的attribute名称，值是该attribute需要应用的值

如果属性名称是data-something的话，如下写法是错误的需要加’ data-something ’

<div data-bind="attr: { ‘data-something’: someValue }">...</div>

foreach绑定: 使用此功能可以方便我们循环遍历输出某个数组、集合中的内容



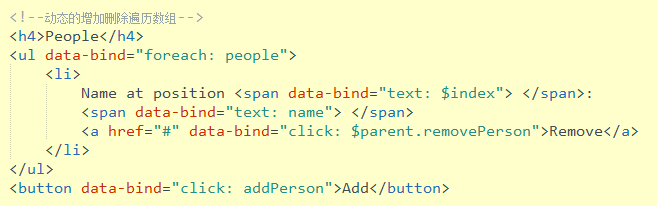
动态增加删除遍历数组

foreach绑定

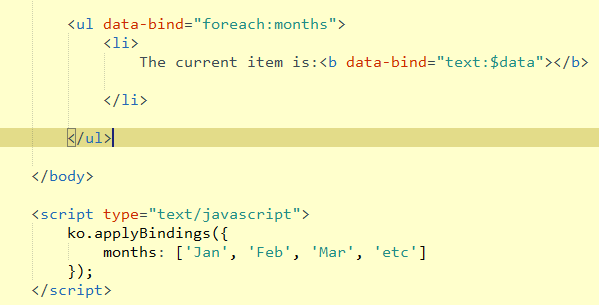
要求的数组格式是[{key:value1},{key:value2}] ，这可能与你数据库保存的数组格式不同。于是，你需要个转化函数。

如果你想绑定[value1,value2,value3]这种数组格式，要用$data。但是，这样只能显示数据，你在view里修改了数据，不会影响到view-model，因为这是值传递。

$parent概念，当前对象所在集合了，经常用于remove。$root概念，就是集合a包集合b包集合c，集合c的$root就是集合a，集合c的$parent就是集合b。



$data的使用

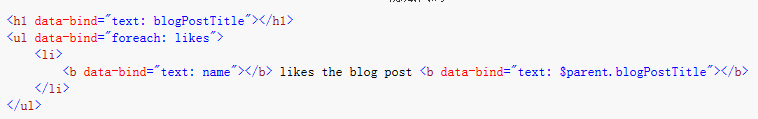


我们也可以使用$data来调用数组中具体的元素，比如我们要使用$data调用例1中的firstName的话，我们可以使用$data.firstName来输出firstName的具体内容

$index、$parent属性说明

$index来表示我们数组的下标，$index是Knockoutjs为我们提供的属性，我们可以直接引用，它会随着数组等下标的变化动态变化的，比如如果数组的长度减少了1，$index的值也会跟着减少1

使用$parent来使用foreach元素之外的属性



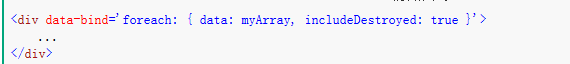
使用"as"为foreach中的元素定义别名

我们可以使用$data来代替数组中的元素，同时我们也可以使用as来为我们要遍历的元素起一个别名，在使用的时候注意别名要加’ ’

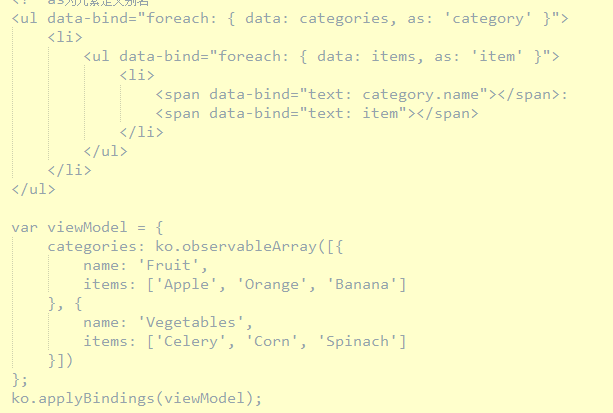
在没有绑定属性的情况下使用foreach

<!--ko--><!--/ko-->：虚拟标签的使用

默认不显示被标示为删除的元素

有的时候我们要跳过数组中的一些元素，此时这些元素已经被标示为删除，但并没有被真实删除，这些元素当我们使用foreach输出的时候是默认不显示的，如果我们想要这些元素显示的话，我们可以使用includeDestroyed这个选项





If binding/if not binding

使用if binding可以控制某个组件动态显示，类似我们之前接触到的visible属性，不过此属性绑定过以后就不能更改， if binding可以根据相应的条件控制组件是否显示

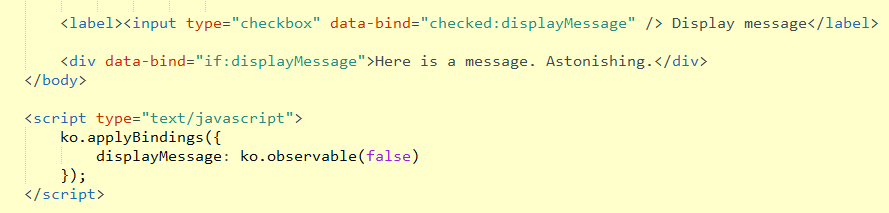
与visible binding类似，都可以控制一段内容是否出现在页面中。

与visible不同的是，if binding是真正的控制Html标签是否出现在DOM中，如果绑定的值为false,则Html标签不会出现在页面中。

if绑定会修改DOM结构，所以出于性能考虑，不应该频繁修改if的绑定值。（这种情况应该使用visible binding）

如果与observable或者computed属性绑定，则会产生双向绑定效果。

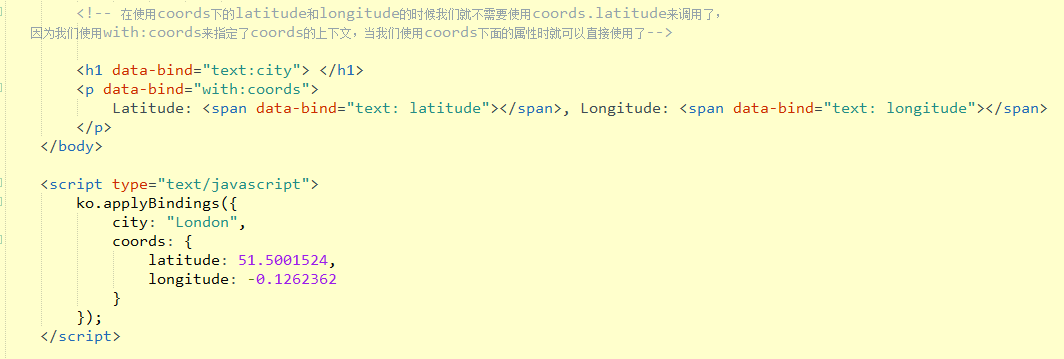
根据checkbox是否勾选来控制是否显示下面的一个<div>



我们也可以使用if来判断某个元素是否为null，如果为null则不会显示



with binding来重新定义一个上下文绑定



click绑定: click绑定在DOM元素上添加事件句柄以便元素被点击的时候执行定义的JavaScript 函数。大部分是用在button，input和连接上，但是可以在任意元素上使用

计数案例



Click点击事件时所执行的函数。

你可以声明任何JavaScript函数 – 不一定非要是view model里的函数。你可以声明任意对象上的任何函数，例如：someObject.someFunction

参数传递

<button data-bind="click: function() { viewModel.myFunction('param1', 'param2') }">

Click me

</button>

点击安缪KO就会调用这个匿名函数，里面会执行viewModel.myFunction()，并且传进了'param1' 和'param2'参数

访问事件源对象



如果你需要的话，可以使用匿名函数的第一个参数传进去

<button data-bind="click: function(event) { viewModel.myFunction(event, 'param1', 'param2') }">

Click me

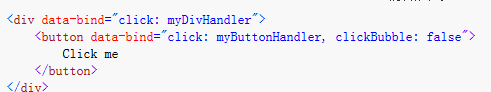
</button>

允许执行默认事件

默认情况下，Knockout会阻止冒泡，防止默认的事件继续执行。例如，如果你点击一个a连接，在执行完自定义事件时它不会连接到href地址。这特别有用是因为你的自定义事件主要就是操作你的view model，而不是连接到另外一个页面。

当然，如果你想让默认的事件继续执行，你可以在你click的自定义函数里返回true。

防止冒泡事件的执行

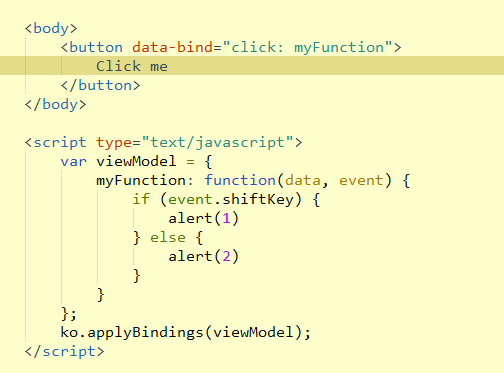
默认情况下，Knockout允许click事件继续在更高一层的事件句柄上冒泡执行。例如，如果你的元素和父元素都绑定了click事件，那当你点击该元素的时候两个事件都会触发的。如果需要，你可以通过额外的绑定clickBubble来禁止冒泡

默认情况下，myButtonHandler会先执行，然后会冒泡执行myDivHandler。但一旦你设置了clickBubble为false的时候，冒泡事件会被禁止

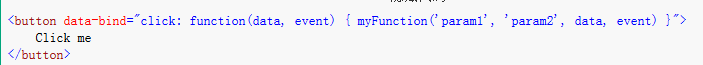
我们也可以把当前元素作为参数传递给所要调用的方法，如下例子：我们点击按钮，然后删除按钮对应的元素



定义事件



多个参数传递：如果我们想要传递很多参数给调用的方法的话，我们可以在标签中设定对应方法要传递的参数

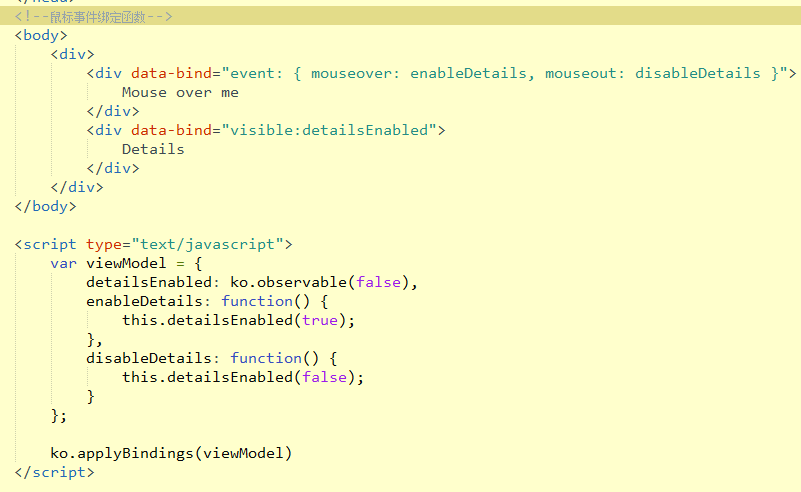


如果不想在标签中出现那么多内容我们可使用KO的bind方法进行参数传递



event绑定

event绑定在DOM元素上添加指定的事件句柄以便元素被触发的时候执行定义的JavaScript 函数



参数说明

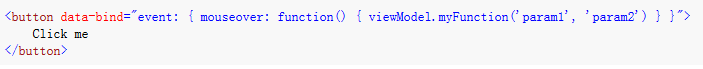
事件的值是一个JavaScript对象，他的属性名是事件名称，值是你所需要执行的函数。

你可以声明任何JavaScript函数 – 不一定非要是view model里的函数。你可以声明任意对象上的任何函数，例如：event: { mouseover: someObject.someFunction }。

View model上的函数在用的时候有一点点特殊，就是不需要引用对象的，直接引用函数本身就行了，比如直接写event: { mouseover: enableDetails }就可以了，而无需写成：event: { mouseover: viewModel.enableDetails }（尽管是合法的）。

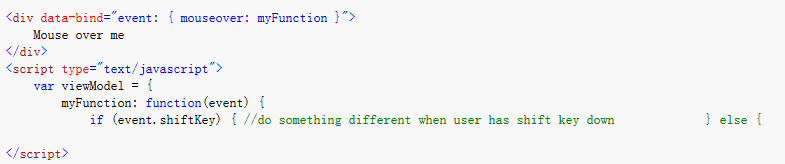
参数传递

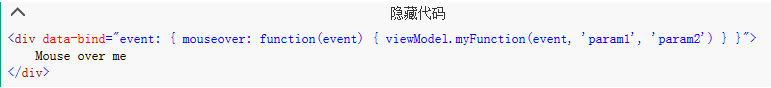
简单的办法是传一个function包装的匿名函数



鼠标滑过时调用匿名函数并设定传递的参数是param1,param2

访问事件源对象: 有些情况，你可能需要使用事件源对象，Knockout会将这个对象传递到你函数的第一个参数



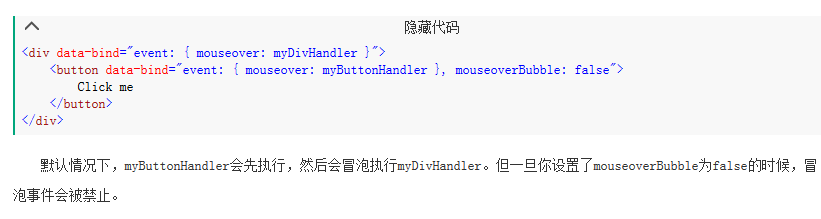
如果你需要的话，可以使用匿名函数的第一个参数传进去，然后在里面调用：

允许执行默认事件

默认情况下，Knockout会阻止冒泡，防止默认的事件继续执行。例如，如果在一个input标签上绑定一个keypress事件，当你输入内容的时候，浏览器只会调用你的函数而不是天价你输入的值。另外一个例子click绑定，当你点击一个a连接，在执行完自定义事件时它不会连接到href地址。因为你的自定义事件主要就是操作你的view model，而不是连接到另外一个页面。当然，如果你想让默认的事件继续执行，你可以在你event的自定义函数里返回true

防止事件冒泡

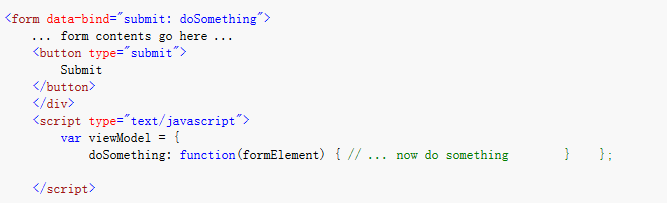
默认情况下，Knockout允许event事件继续在更高一层的事件句柄上冒泡执行。例如，如果你的元素和父元素都绑定了mouseover事件，那么如果你的鼠标在该元素移动的时候两个事件都会触发的。如果需要，你可以通过额外的绑定youreventBubble来禁止冒泡



Submit绑定

submit绑定在form表单上添加指定的事件句柄以便该form被提交的时候执行定义的JavaScript 函数。只能用在表单form元素上。

当你使用submit绑定的时候， Knockout会阻止form表单默认的submit动作。换句话说，浏览器会执行你定义的绑定函数而不会提交这个form表单到服务器上。可以很好地解释这个，使用submit绑定就是为了处理view model的自定义函数的，而不是再使用普通的HTML form表单。如果你要继续执行默认的HTML form表单操作，你可以在你的submit句柄里返回true

这个例子里，KO将把整个form表单元素作为参数传递到你的submit绑定函数里。 你可以忽略不管

不在submit上绑定click事件的原因：你可以使用click绑定代替submit绑定。不过submit可以handle其它的submit行为，比如在输入框里输入回车的时候可以提交表单。

你可以声明任何JavaScript函数 – 不一定非要是view model里的函数。你可以声明任意对象上的任何函数，例如： submit: someObject.someFunction。

View model上的函数在用的时候有一点点特殊，就是不需要引用对象的，直接引用函数本身就行了，比如直接写submit: doSomething就可以了，而无需写成： submit: viewModel. doSomething（尽管是合法的）

Value绑定

value绑定是关联DOM元素的值到view model的属性上。主要是用在表单控件<input>，<select>和<textarea>上。

当用户编辑表单控件的时候， view model对应的属性值会自动更新。同样，当你更新view model属性的时候，相对应的元素值在页面上也会自动更新。

注：如果你在checkbox或者radio button上使用checked绑定来读取或者写入元素的 checked状态，而不是value 值的绑定

Value：绑定

### 

使用valueUpdate参数，那就是意味着KO将使用自定义的事件而不是默认的离开焦点事件。下面是一些最常用的选项：

“change”（默认值） - 当失去焦点的时候更新view model的值，

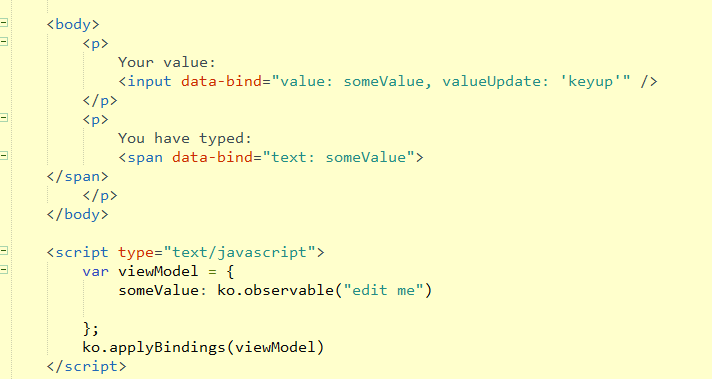
或者是<select> 元素被选择的时候。

“keyup” – 当用户敲完一个字符以后立即更新view model。

“keypress” – 当用户正在敲一个字符但没有释放键盘的时候就立即更新view model。不像 keyup，这个更新和keydown是一样的。

“afterkeydown” – 当用户开始输入字符的时候就更新view model。主要是捕获浏览器的keydown事件或异步handle事件。

上述这些选项，如果你想让你的view model进行实时更新，使用“afterkeydown”是最好的选择



模板绑定

模板绑定的两种方式

原生模板 支持foreach 与其他控制流绑定的机制在内部，这些控制流绑定捕获元素中包含的HTML标记，并将其用作模板来针对任意数据项进行渲染。 此功能内置于Knockout中，不需要任何外部库。

基于字符串的模板

是将Knockout连接到第三方模板引擎的一种方式。 Knockout会将您的模型值传递给外部模板引擎，并将生成的标记字符串插入到文档中。 请参阅下面有关使用jquery.tmpl和Underscore模板引擎的示例。

主要参数

如果您只提供字符串值，则KO会将其解释为要呈现的模板的ID。 它提供给模板的数据将是您当前的模型对象。

要获得更多控制权，请传递一个JavaScript对象，并结合以下属性

Name:可视为模板的ID

Nodes: 直接传递一个DOM节点数组作为模板。 这应该是一个不可观察的数组，并且请注意，如果元素有一个元素，那么这些元素将从其当前父级元素中移除。 如果您还为名称传递了非空值，则此选项将被忽略。

Data：作为要呈现的模板的数据提供的对象。 如果省略此参数，KO将查找一个foreach参数，或者将回退使用当前的模型对象。

If：如果提供此参数，则只有在指定的表达式计算为真（或真实值）时才会呈现模板。 这可以用于防止null观察值在模板被填充之前与模板绑定。

Foreach：指示KO在“foreach”模式下呈现模板

As：当与foreach结合使用时，为每个正在呈现的项目定义一个别名

afterRender, afterAdd, or beforeRemov:”回调函数针对渲染的DOM元素进行调用

命名模板

通常使用控制流绑定（foreach，with，if等）时，不需要为模板指定名称：它们是由DOM元素中的标记隐式匿名定义的。你也可以将模板分解成一个单独的元素，然后通过名称引用它们

看如下实例



执行结果



说明：在此示例中，人员模板标记会使用两次：一次是买方，一次是卖方。模板标记包装在<script type =“text / html”>中 - 为了确保标记不作为JavaScript执行，必须使用虚拟类型属性，这样Knockout不会尝试将绑定应用于该标记，除非它被用作模板。

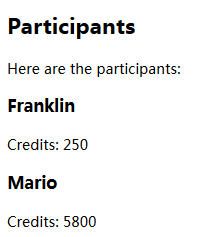
通常不需要使用命名模板，但有时它可以帮助最大限度地减少重复标记

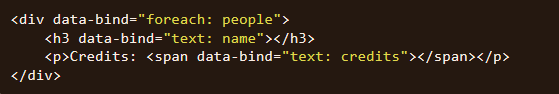
命名模板中使用foreach

如果你想要一个已命名的模板与foreach绑定 ，你可以按照以下方式进行



执行结果

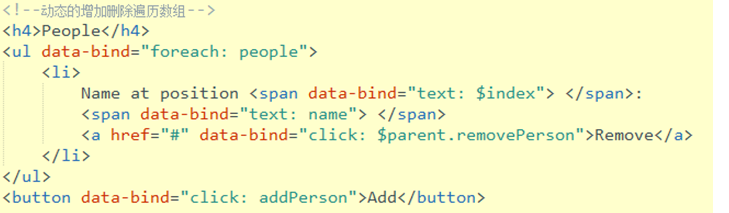


这给出了与使用foreach的元素直接嵌入匿名模板达到与命名模板相同的效果

使用“as”为“foreach”项目提供别名

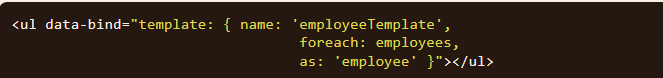
嵌套foreach模板时，引用层次结构中较高级别的项目通常很有用。

一种方法是是使用$ parent绑定上下文变量。



简单的一种是使用as为迭代变量声明一个名称。

注意与as关联的字符串值'employee'。 现在，在此foreach循环内的任何位置，您的子模板中的绑定将能够使用employee访问员工对象



别名对多层嵌套的foreach很有用的，它给你提供了一个明确的方式来引用层次结构中更高级别声明的任何命名项。 下面的例子，展示了如何在渲染一个月的同时引用季节

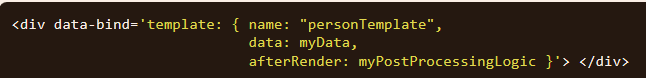




使用 “afterRender”, “afterAdd”, and “beforeRemove”

有时，您希望对由模板生成的DOM元素运行自定义后处理逻辑。如使用JavaScript小部件库（如jQuery UI），需要拦截模板的输出，以便可以在其上运行jQuery UI命令时将某些呈现的元素转换为日期选择器，滑块或其他结果集。通常，对DOM元素执行这种后处理的最佳方式是编写自定义绑定

如果您真的只想访问由模板发出的原始DOM元素，则可以使用afterRender。  
  
传递一个函数或传递视图模型上的函数的名称，Knockout将在渲染或重新渲染模板之后立即调用它。



viewModel.myPostProcessingLogic = function(elements) {

//“元素”是刚刚由模板呈现的DOM节点的数组

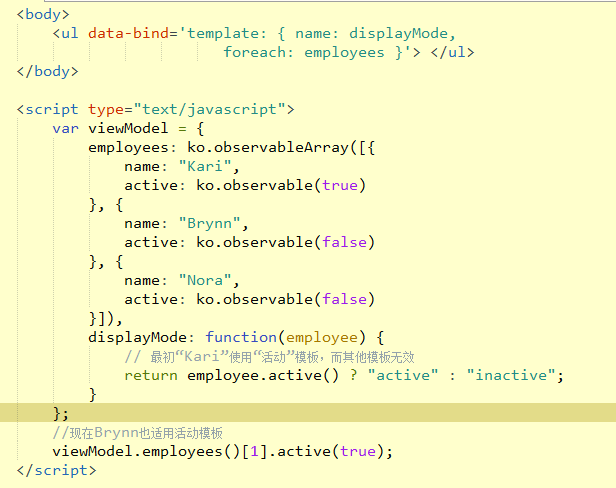
//你可以在这里添加自定义的后期处理逻辑

}

如果您使用的是foreach，并且想在添加或删除的元素时执行一段逻辑可以使用afterAdd and beforeRemove具体可以查看foreach绑定

动态选择调用模板

如果您有多个命名模板，则可以将名称选项的observable传递给它。 随着observable的值更新，元素的内容将使用适当的模板重新呈现。 或者可以传递回调函数以确定要使用的模板。 如果您正在使用foreach模板模式，Knockout将评估数组中每个项目的函数，并将该项目的值作为唯一参数传递。 否则，返回该函数将被赋予数据选项的值或提供整个当前模型对象。



如果你的函数引用了可观察值，那么只要这些值有变化，绑定就会更新。 这将导致数据使用适当的模板重新呈现。

如果您的函数接受第二个参数，那么它将接收整个绑定上下文。 您可以在动态选择模板时访问$ parent或任何其他绑定上下文变量。 例如，你可以修改前面的代码片段，如下所示

displayMode: function(employee, bindingContext) {

    // 现在根据employee或bindingContext的属性返回一个模板名称字符串

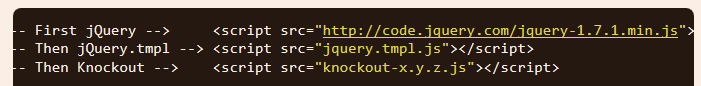
}

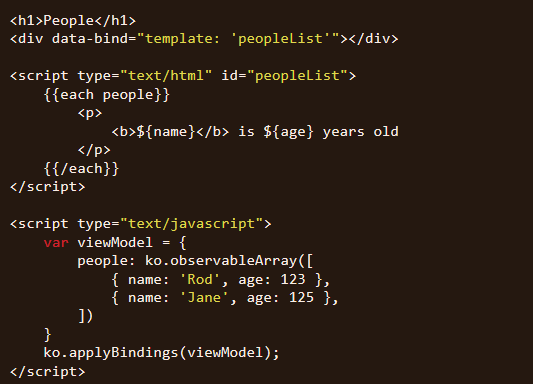
## 使用jQuery.tmpl，一个基于字符串的外部模板引擎

## 在绝大多数情况下，Knockout的原生模板和foreach（如果使用其他控件流绑定）将成为构建任意复杂UI的唯一需求。 但是，如果您想要与外部模板库（如Underscore模板引擎或jquery.tmpl）集成，Knockout提供了一种方法。

## 默认情况下，Knockout支持jquery.tmpl。 要使用它，您需要按以下顺序引用以下库：

## 





这是有效的，因为{{each ...}}和$ {...}是jQuery.tmpl语法。 更重要的是，嵌套模板很简单：因为您可以使用模板内部的数据绑定属性，所以您可以简单地在模板中放置一个data-bind =“template：...”来呈现嵌套模板。

请注意，截至2011年12月，jQuery.tmpl不再处于积极的发展状态。 我们建议使用Knockout的原生基于DOM的模板（即，foreach，if，with等绑定）而不是jQuery.tmpl或任何其他基于字符串的模板引擎。

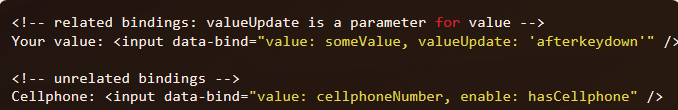
数据绑定的语法

Knockout的声明式绑定系统提供了一种将数据链接到UI的简洁且强大的方法， 绑定到简单的数据属性或使用单个绑定通常很简单明显。 对于更复杂的绑定，它有助于更好地理解Knockout绑定系统的行为和语法

基本组成：绑定名称 ： 绑定的值

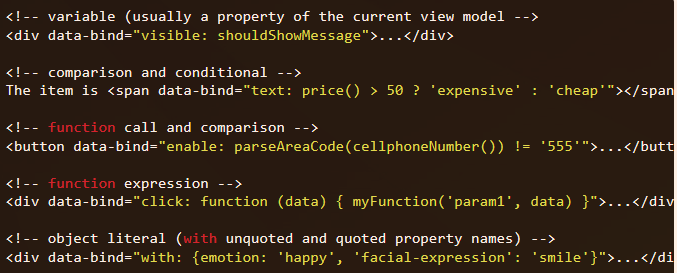


一个元素可以包含多个绑定（相关或不相关），每个绑定用逗号分隔



绑定名称 通常应该与已注册的绑定处理程序（内置或自定义）匹配， 如果名称不匹配，Knockout将忽略它（没有任何错误或警告）。 因此，如果绑定不起作用，请首先检查名称是否正确。

绑定值可以是单个值，变量或文字或几乎任何有效的JavaScript表达式



绑定值可以包含任意数量的空格，制表符和换行符因此您可以随意使用它来根据需要格式化绑定内容



从Knockout 3.0开始，您可以指定没有值的绑定，这会给绑定一个未定义的值。 例如：与绑定预处理（可为绑定指定默认值）配对时，此功能特别有用。



binding context

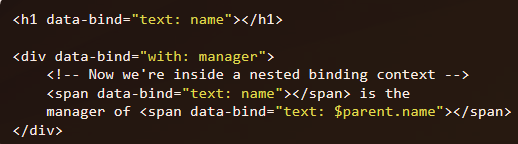
绑定上下文(binding context)是一个对象，它持有一些数据，可供你在你的绑定中引用。在使用绑定的时候，Knockout自动地创建并管理绑定上下文的层次结构。该层次结构的根级别指向 你传递给ko.applyBindings(viewModel)的viewModel参数。

然后，每当你使用一个‘诸如 [with](http://www.javaarm.com/file/www/javascript/fwk/knockoutjs/site-back/documentation_cn/with-binding.html)或者[foreach](http://www.javaarm.com/file/www/javascript/fwk/knockoutjs/site-back/documentation_cn/foreach-binding.html)之类’的控制流绑定时，它就会创建一个 子级绑定上下文，指向被嵌入(到该上下文之内)的视图模型的数据。

绑定上下文提供了以下特殊的属性，你可以在任何绑定中引用它们

$parent

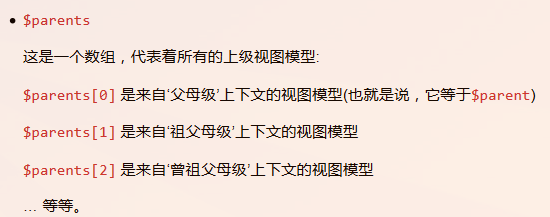
这是上级上下文中的视图模型对象，与当前上下文紧接着的外层上下文。在根上下文中，它是undefined

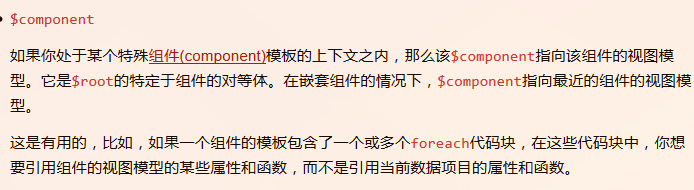
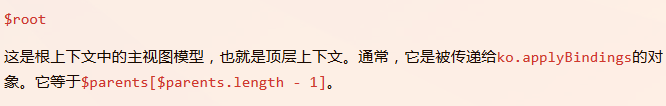


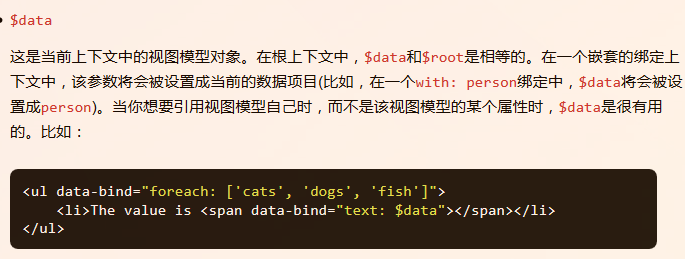
#### 视图模型：指的viewModel

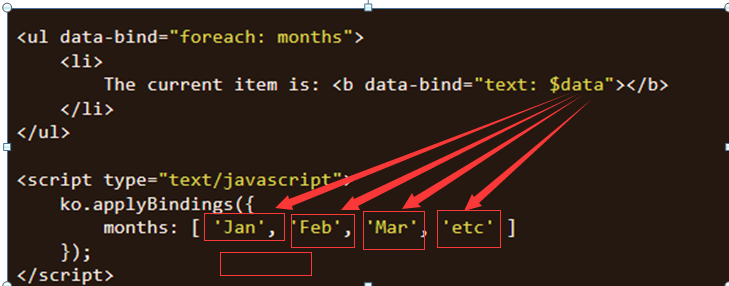


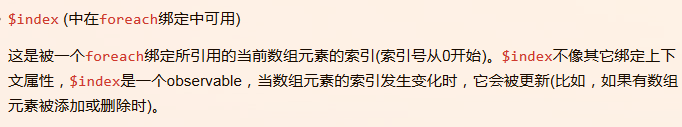
说明：$root经常是相同的，而$parent是不同的，而这种不同主要取决于你嵌套的深度



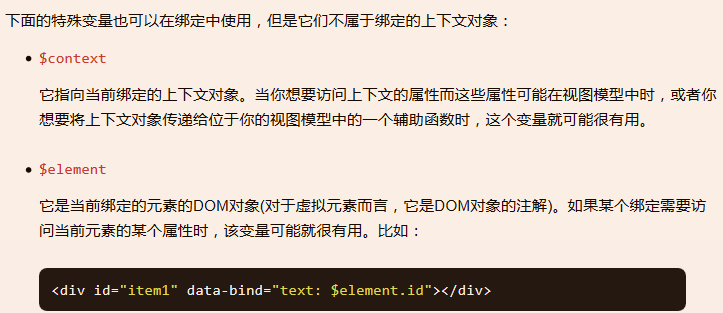
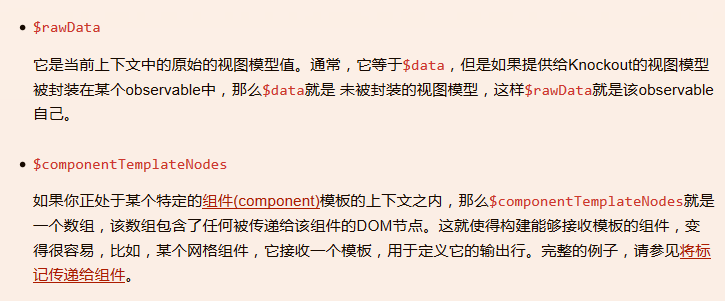
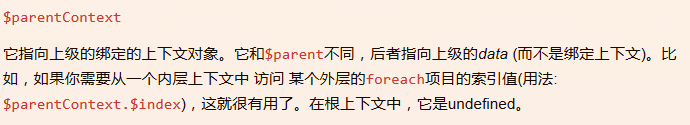






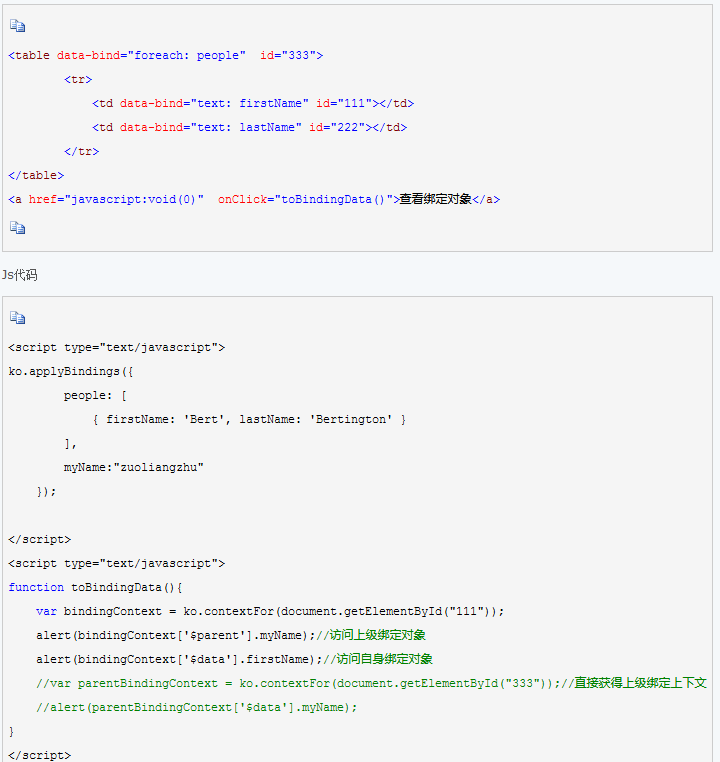


在foreach中$index与$data可分别代表角标和元素



每个具有声明式绑定的dom元素，ko都会为其设置一个绑定上下文(bindingContext)和一个绑定对象。而每个绑定上下文中都包含$parent和$data俩个属性，$data属性就是该dom元素的绑定对象，$parent属性是上级元素的绑定对象。也就是说如果我们要向访问绑定对象，首先要获得绑定上下文，在通过绑定上下文来访问绑定对象：bingingContext.$data.属性名、bingingContext.$parent.属性名。而我们以前的例子中都是只用$data.属性名、$parent.属性名就可以访问，这是因为在ko内部(准确说是在binding组件)已经帮我们获得了bingingContext。如果我们想在ko外部直接访问绑定对象或绑定上下文，可以使用：ko.dataFor(element)、ko.contextFor(element)。







1.值更改触发时间

更改view后，焦点离开当前dom元素，view-model的值就改变。你可以设置valueUpdate: "afterkeydown"使得view值改变，view-model值立马改变（类似于onkeyup事件）。

更改view-model后，view的值立马改变。

2.注意引用传递和值传递

值传递，最常见的就是你model类型是string和number，于是你改了model的值，view-model不变。

引用传递，最常见的就是Array类型，你的Model类型Array数据，shift()了，你的view-model改变，你的view也改变了。

3.select元素的单击事件

select元素，你绑定了click事件，其实是双击事件，它的单击事件是change。

4.foreach绑定

一，要求的数组格式是[{key:value1},{key:value2}] ，这可能与你数据库保存的数组格式不同。于是，你需要个转化函数。

二，如果你想绑定[value1,value2,value3]这种数组格式，要用$data。但是，这样只能显示数据，你在view里修改了数据，不会影响到view-model，因为这是值传递。

三，$parent概念，当前对象所在集合了，经常用于remove。$root概念，就是集合a包集合b包集合c，集合c的$root就是集合a，集合c的$parent就是集合b。

5.with绑定

foreach绑定的是数组，with绑定的则是对象。

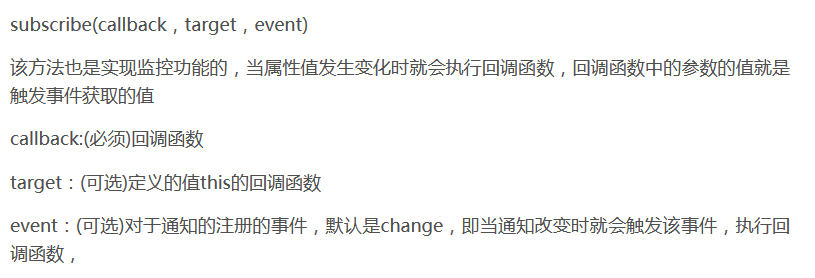
6.自定义绑定

这下你牛B了，value绑定，text绑定都只是knockout自带的，你也写出yourname绑定。

组件（实现特定功能的UI包）和自定义元素

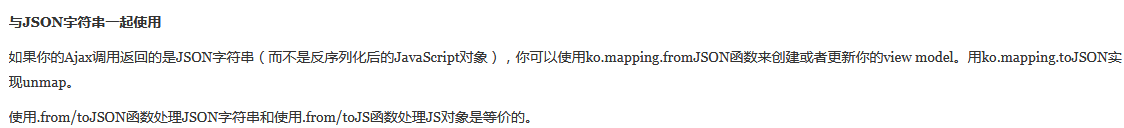
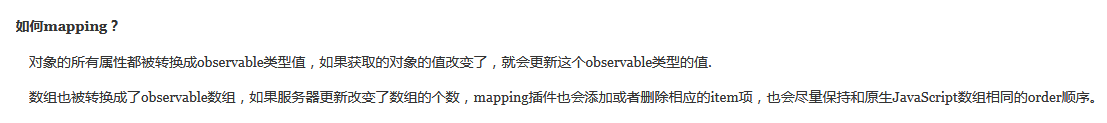
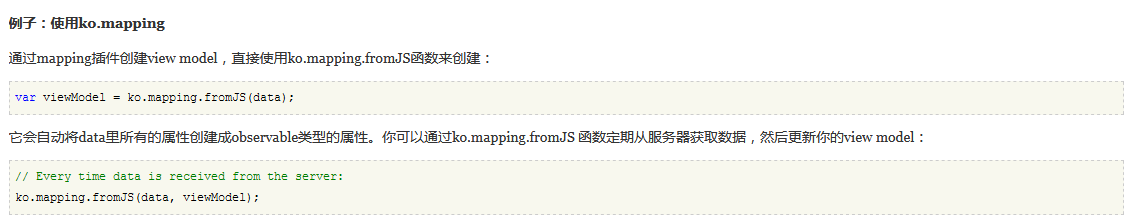
要开始，您可以使用ko.components.register注册一个组件（技术上，注册是可选的，但它是最简单的入门方式）。 组件定义指定了一个viewModel和模板。 例如

knockout中subscribe()方法



Mapping插件

Knockout设计成允许使用任何JavaScript对象作为view model。必须view model的一些属性是observable的，你可以使用KO绑定他们到你的UI元素上，当这些observable值改变的时候，这些UI元素就会自动更新。

绝大多数程序都需要从服务器端获取数据，由于服务器不知道observable的概念是什么，它只支持简单的JavaScript对象（通常是序列化以后的JSON），mapping插件可以让你很方便地将简单JavaScript对象map到带到有observable值的view model（相当于数据转换）。当然你也可以自己写JavaScript代码将从服务器获取的数据来构建 view model，mapping插件只是一种很好的替代而已

更多细节访问https://www.cnblogs.com/TomXu/archive/2011/11/29/2257054.html

JQuery的extend扩展方法

Jquery的扩展方法extend是我们在写插件的过程中常用的方法，该方法有一些重载原型

方法原型：extend(dest,src1,src2,src3...);

它的含义是将src1,src2,src3...合并到dest中,返回值为合并后的dest,由此可以看出该方法合并后，是修改了dest的结构的

如果想要得到合并的结果却又不想修改dest的结构，可以如下使用

var newSrc=$.extend({},src1,src2,src3...)//也就是将"{}"作为dest参数。

这样就可以将src1,src2,src3...进行合并，然后将合并结果返回给newSrc了。如下例

var result=$.extend({},{name:"Tom",age:21},{name:"Jerry",sex:"Boy"})

合并后的结果：result={name:"Jerry",age:21,sex:"Boy"}

也就是说后面的参数如果和前面的参数存在相同的名称，那么后面的会覆盖前面的参数值。

省略dest参数  
      上述的extend方法原型中的dest参数是可以省略的，如果省略了，则该方法就只能有一个src参数，而且是将该src合并到调用extend方法的对象中去

**1、$.extend(src)** 　　该方法就是将src合并到jquery的全局对象中去，如：

$.extend({  
hello:function(){alert('hello');}  
});

Jquery的extend方法还有一个重载原型

extend(boolean,dest,src1,src2,src3...)

第一个参数boolean代表是否进行深度拷贝，其余参数和前面介绍的一致，什么叫深层拷贝，我们看一个例子

