آموزش KnockoutJS ـ آموزش event binding در KnockoutJS

هدف

این binding این امکان را برای شما فراهم می کند تا برای یکی از رویدادهای مشخص، event handler ای را اضافه کنید. به صورتی که زمانی که این رویداد در المان مربوطه ی DOM فعال شود، تابع جاوا اسکریپت انتخاب شده ی شما احضار خواهد شد. از event binding می توان برای مقید شدن به تمامی event ها مانند keypress, mouseover یا mouseout استفاده کرد.

مثال:

```
<div>
  <div data-bind="event: { mouseover: enableDetails, mouseout: disableDetails }">
    Mouse over me
  </div>
  <div data-bind="visible: detailsEnabled">
    Details
  </div>
</div>

<script type="text/javascript">
  var viewModel = {
    detailsEnabled: ko.observable(false),
    enableDetails: function() {
      this.detailsEnabled(true);
    },
    disableDetails: function() {
      this.detailsEnabled(false);
    }
  };
  ko.applyBindings(viewModel);
</script>
```

حالا اگر موس خود را بر بروی اولین المان ببرید یا از روی آن بردارید، در view model متدهایی احضار می شود که باعث فعال شدن detailsEnabled observable می شود. المان دوم با نشان دادن یا مخفی کردن خود به تغییرات مقدار detailsEnabled واکنش نشان می دهد.

پارامترها

- پارامتر اصلی

شما باید شیء جاوا اسکریپتی را وارد کنید که نام مشخصه های آن با نام event ها مطابقت داشته باشد. علاوه بر این مقادیر نیز باید با تابعی مطابقت داشته باشند که می خواهید آن تابع را به این رویداد مقید کنید.

هیچ محدودیتی در اشاره به توابع جاوا اسکریپت برای شما وجود ندارد. لزومی ندارد که حتما تابعی در view model شما وجود داشته باشد. می توانید با نوشتن event { mouseover: someObject.someFunction } در هر شیئی به یک تابع اشاره کنید.

- پارامترهای اضافی
- ندارد

نکته 1 : تحویل یک "current item" به تابع handler به عنوان یک پارامتر

زمانی که تابع handler را فراخوانی می کنید، KnockoutJS مقدار مدل فعلی را به عنوان اولین پارامتر ارائه می کند. این کار مخصوصا زمانی که بخواهید برخی از بخش های رابط کاربری را به ازای هر یک از آیتم ها در مجموعه ای نمایش دهید، و لازم باشد که بدانید که رویداد به کدام آیتم تشاره می کند، کمک شایانی به شما می کند.
برای مثال:

```
<ul data-bind="foreach: places">
  <li data-bind="text: $data, event: { mouseover: $parent.logMouseOver }"> </li>
</ul>
<p>You seem to be interested in: <span data-bind="text: lastInterest"> </span></p>

 <script type="text/javascript">
   function MyViewModel() {
     var self = this;
     self.lastInterest = ko.observable();
     self.places = ko.observableArray(['London', 'Paris', 'Tokyo']);

     // The current item will be passed as the first parameter, so we know which place was
hovered over
     self.logMouseOver = function(place) {
       self.lastInterest(place);
     }
   }
   ko.applyBindings(new MyViewModel());
</script>
```

دو نکته در این مثال وجود دارد که باید به آن ها توجه کنید:

- اگر داخل یک binding context تو در تو قرار دارید (مثلا اگر داخل یک بلوک with یا foreach باشید) اما تابع handler شما در view model ریشه (root viewmodel) یا در parent context دیگری قرار داشته باشد، در این صورت برای شناسایی موقعیت تابع handler باید از پیشوندی مانند $parent یا $root استفاده کنید.
- اغلب اعلان کردن self (یا متغیر دیگری) به عنوان نام مستعاری برای this در view model تان می تواند مفید باشد.
در صورتی که این کار را انجام دهید، از مشکل تعریف مجدد this که باعث می شود در event handler ها یا Ajax request callback ها معنی دیگری بدهد، جلوگیری می کنید.

نکته 2 : دسترسی به event object یا تحویل دادن پارامترهای بیشتر

در برخی از حالات نیاز است که به شیء رویداد DOM (DOM event object) مربوط به رویدادتان دسترسی داشته باشید. KnockoutJS این رویداد را به عنوان پارامتر دوم به تابع شما تحویل می دهد. مانند مثال زیر:

```
<div data-bind="event: { mouseover: myFunction }">
  Mouse over me
</div>

 <script type="text/javascript">
  var viewModel = {
    myFunction: function(data, event) {
      if (event.shiftKey) {
        //do something different when user has shift key down
      } else {
        //do normal action
      }
    }
  };
  ko.applyBindings(viewModel);
</script>
```

اگر می خواهید پارامترهای بیشتری را به تابع خود تحویل دهید، یکی از راه های انجام آن قرار دادن handler خود در یک تابع لفظی است که یک پارامتر را مانند مثال زیر قبول میکند:

```
<div data-bind="event: { mouseover: function(data, event) { myFunction('param1', 'param2',
data, event) } }">
  Mouse over me
</div>
```

حالا KnockoutJS این رویداد را به تابع لفظی تان تحویل می دهد. که بعد از این، این رویداد آماده ی تحویل به handler تان می شود.

به عنوان جایگزین، اگر دوست ندارید توابع لفظی را در view خود به کار ببرید، می توانید از bind function استفاده کنید. این تابع مقادیر مشخصی از پارامتر را به یک function reference متصل می کند:

```
<button data-bind="event: { mouseover: myFunction.bind($data, 'param1', 'param2') }">
  Click me
</button>
```

نکته 3 : مجاز کردن default action

به صورت پیش فرض KnockoutJS از اینکه رویداد هر کار پیش فرضی را انجام دهد جلوگیری می کند. این یعنی اگر شما برای capture کردن رویداد keypress مربوط به یک تگ input از event binding استفاده کنید، در این صورت مرورگر تنها تابع handler شما را فراخوانی می کند. و دیگر مقدار key را به مقدار المان input اضافه نخواهد کرد. یک مثال معمول استفاده از click binding است که در این binding از event binding به صورت درونی استفاده می شود و تابع handler شما فراخوانی می شود. اما مرورگر به href مربوط به لینک نخواهد رفت. این پیش فرض به این دلیل مفید است که شما زمانی از click binding استفاده می کنید که می خواهید از این لینک به عنوان بخشی از رابط کاربری استفاده کنید که view model شما را کنترل کند، نه به عنوان یک hyperlink معمولی که به یک صفحه ی اینترنتی دیگر متصل شده است.

با این حال اگر بخواهید که default action ادامه داشته باشد، تنها کافیست true را از click handler function خود برگشت دهید.

نکته 4 : جلوگیری رویداد از حبابی شدن (bubbling)

به صورت پیش فرض KnockoutJS این امکان را برای event فراهم می کند تا مانند حباب به سمت یکی از event handler های سطح بالاتر برود. برای مثال اگر المان شما و parent آن المان در حال مدیریت mouseover event باشند، در این صورت event handler مربوط به هر دوی این المان ها فعال می شود. در صورت لزوم می توانید با لحاظ کردن یک binding اضافی به نام youreventBubble و قرار دادن مقدار آن بر روی false از حبابی شدن رویداد خود جلوگیری کنید. مانند مثال زیر:

```
<div data-bind="event: { mouseover: myDivHandler }">
  <button data-bind="event: { mouseover: myButtonHandler }, mouseoverBubble: false">
    Click me
  </button>
</div>
```

به طور معمول در حالت بالا myButtonHandler ابتدا فراخوانی می شود. سپس رویداد به صورت حبابی به سمت myDivHandler حرکت می کند. هرچند که mouseoverBubble binding ای که ما با مقدار false وارد کردیم باعث شد رویداد کلیک از myButtonHandler بالاتر نرود.

نکته 5 : آموزش ایجاد تعامل با جی کوئری در KnockoutJS

جی کوئری در صورتی که موجود باشد، KnockoutJS از آن برای مدیریت رویدادهای رابط کاربری استفاده می کند. برای این که این رفتار را غیرفعال کنید و به KnockoutJS بفهمانید که همیشه از event handling بومی استفاده کند، می توانید پیش از فراخوانی ko.applyBindings آپشن زیر را در کدتان قرار دهید:

```
ko.options.useOnlyNativeEvents = true;
```

وابستگی ها

به غیر از کتابخانه ی اصلی KnockoutJS وابستگی دیگری ندارد.