آموزش KnockoutJS ـ آموزش click binding در KnockoutJS

هدف

کار این binding اضافه کردن یک event handler است به گونه ای که بعد از اینکه المان DOM مربوطه کلیک می شود، تابع جاوا اسکریپت احضار می شود. کاربرد این binding بیشتر در المان هایی مانند button, input و a ظهور پیدا می کند. اما در واقع در کنار همه ی المان های قابل دیدن DOM کار می کند.

مثال

```
<div>
  You've clicked <span data-bind="text: numberOfClicks"></span> times
  <button data-bind="click: incrementClickCounter">Click me</button>
</div>

<script type="text/javascript">
  var viewModel = {
    numberOfClicks : ko.observable(0),
    incrementClickCounter : function() {
      var previousCount = this.numberOfClicks();
      this.numberOfClicks(previousCount + 1);
    }
  };
</script>
```

کلیک کردن شما بر روی دکمه باعث احضار شدن incrementClickCounter() در view model می شود و در ادامه این کار باعث تغییر دادن حالت view model و آپدیت شدن رابط کاربری می شود.

پارامترها

- پارامتر اصلی

  پارامتر اصلی تابعی است که میخواهید به رویداد click المان مقید کنید.

  محدودیتی در اشاره به توابع جاوا اسکریپت وجود ندارد (الزامی در بودن تابع در view model شما وجود ندارد). با نوشتن click: someObject.someFunction می توانید در تمامی شیء ها 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 بر روی کدام یک از بخش های رابط کاربری آیتم کلیک شده است، کمک شایانی به شما می کند. برای مثال:

```
<ul data-bind="foreach: places">
  <li>
    <span data-bind="text: $data"></span>
    <button data-bind="click: $parent.removePlace">Remove</button>
  </li>
</ul>

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

     // The current item will be passed as the first parameter, so we know which place to
remove
     self.removePlace = function(place) {
       self.places.remove(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 رویداد کلیک تان دسترسی داشته باشید. KnockoutJS این رویداد را به عنوان پارامتر دوم به تابع شما تحویل می دهد. مانند مثال زیر:

```
<button data-bind="click: myFunction">
  Click me
</button>

 <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:

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

حالا KnockoutJS داده ها و اشیاء رویداد را به تابع لفظی تان تحویل می دهد. که بعد از این، این دو آماده ی تحویل به handler تان می شوند.
به عنوان جایگزین، اگر دوست ندارید توابع لفظی را در view خود به کار ببرید، می توانید از function bind استفاده کنید. این تابع مقادیر مشخصی از پارامتر را به یک function reference متصل می کند:

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

نکته 3 : مجاز کردن click action پیش فرض

به صورت پیش فرض KnockoutJS از اینکه رویداد کلیک هر کار پیش فرضی را انجام دهد جلوگیری می کند. این یعنی اگر شما در یک تگ a (یک لینک) از click binding استفاده کنید، در این صورت مرورگر تنها تابع handler شما را فراخوانی می کند. و دیگر به href 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 کار پیش فرض کلیک ادامه داشته باشد، تنها کافیست که 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 click 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 رویداد click هستند، در این صورت click handler مربوط به هر دوی این المان ها فعال می شود. در صورت لزوم می توانید با لحاظ کردن یک binding اضافی به نام clickBubble و قرار دادن مقدار آن بر روی false از حبابی شدن رویداد خود جلوگیری کنید. مانند مثال زیر:

```
<div data-bind="click: myDivHandler">
  <button data-bind="click: myButtonHandler, clickBubble: false">
    Click me
  </button>
</div>
```

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

نکته 5 : ایجاد تعامل با جی کوئری

جی کوئری در صورتی که موجود باشد، KnockoutJS از آن برای مدیریت رویدادهای رابط کاربری مانند click استفاده می کند. برای این که این رفتار را غیرفعال کنید و به KnockoutJS بفهمانید که همیشه از event handling های 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.