عنوان: **آموزش Knockout.Js #4**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۲/۰۷/۰۲ ۸:۳۰
آدرس: www.dotnettips.info
گروه‌ها: KnockoutJS, JavaScriptFrameWork

مقید سازی رویداد کلیک

Click Binding روشی است برای اضافه کردن یک گرداننده رویداد در زمانی که قصد داریم یک تابع جاوااسکریپتی را در هنگام کلیک بر روی المان مورد نظر فراخوانی کنیم. از این مقید سازی عموما در عناصر button و input و تگ a استفاده می‌شود. اما در حقیقت در تمام عناصر غیر پنهان صفحه مورد استفاده قرار می‌گیرد.

```
<div>
    Number Of Clicks <span data-bind="text: numberOfClicks"></span> times
    <button data-bind="click: clickMe">Click me</button>
</div>

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

رویداد کلیک button در کد بالا به تابعی با نام clickMe مقید شده است. این تابع در viewModel جاری صفحه تعریف شده است و در بدنه آن تعداد کلیک‌های قبلی را به علاوه یک خواهد کرد. از آنجا که تگ span در بالای صفحه به تعداد کلیک‌ها مقید شده است در نتیجه همواره مقدار این تگ به روز خواهد بود.

***نکته اول:** اگر قصد داشته باشیم که عنصر جاری در viewModel را به گرداننده رویداد پاس دهیم چه باید کرد؟
هنگام فراخوانی رویدادها، KO به صورت پیش فرض مقدار جاری مدل را به عنوان اولین پارامتر به این گرداننده پاس می‌دهد. این روش مخصوصا در هنگامی که قصد اجرای عملیاتی خاص بر روی تک تک عناصر یک مجموعه را داشته باشید(مثل حلقه foreach) بسیار مفید خواهد بود.

```
<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(['Tehran', 'Esfahan', 'Shiraz']);

         self.removePlace = function(place) {
             self.places.remove(place)
         }
     }
     ko.applyBindings(new MyViewModel());
</script>
```

در تابع removePlace می‌بینید که مقدار آیتم جاری در لیست به عنوان اولین آرگومان به این تابع پاس داده می‌شود، در نتیجه می‌دانیم که کدام عنصر را باید از لیست مورد نظر حذف کنیم. برای به دست آوردن آیتم جاری در لیست از $parent یا $root می‌توان استفاده کرد.
همان طور که پست قبل توضیح داده شد؛ برای اینکه بتوانیم از یک viewModel به مجموعه از عناصر در یک حلقه foreach مقید کنیم امکان استفاده از اشاره گر this میسر نیست. در نتیجه بهتر است در ابتدای viewModel مقدار این اشاره گر را در یک متغیر معمولی (در اینجا به نام self است) ذخیره کنیم و از این پس این متغیر را برای اشاره به عناصر viewModel به کار بریم. در اینجا self به عنواتن یک alias برای this خواهد بود.

***نکته دوم:** دسترسی به عنصر رویداد

در بعضی مواقع نیاز است در حین فراخوانی رویداد ،عنصر رویداد DOM به عنوان فرستنده در اختیار تابع گرداننده قرار گیرد. خبر خوش این است که KO به صورت پیش فرض این عنصر را نیز به عنوان پارامتر دوم به توابع گرداننده رویداد پاس می‌دهد. برای مثال:

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

 <script type="text/javascript">
    var viewModel = {
        myFunction: function(data, event) {
            if (event.shiftKey) {

            } else {
            }
        }
    };
    ko.applyBindings(viewModel);
</script>
```

تابع myFunction در مثال بالا دارای دو پارامتر است. پارامتر دوم در این تابع به عنوان عنصر فرستنده رویداد مورد استفاده قرار خواهد گرفت. بدین ترتیب در توابع event Handlerها می‌توان به راحتی اطلاعات مورد نیاز درباره آبجکت رویداد را به دست آورد.

***نکته سوم:** به صورت پیش فرض KO از اجرای عملیات پیش فرض رویدادها جلوگیری به عمل می‌آورد. این به این معنی است که اگر برای رویداد کلیک تگ a بک تابع گرداننده تعریف کرده باشید، بعد از کلیک بر روی این المان؛ مرورگر فقط این تابع تعریف شده توسط شما را فراخوانی خواهد کرد و دیگر عملیات راهبری به صفحه مورد نظر در خاصیت href صورت نخواهد گرفت. اگر به هر دلیلی قصد داشته باشیم که این رفتار صورت نگیرد کافیست در انتهای تابع گرداننده رویداد مقدار true برگشت داده شود.

***نکته چهارم:** مفهوم clickBubble

ابتدا به کد زیر دقت کنید:

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

همان طور که مشاهده می‌کنید در کد بالا برای عنصر button یک رویداد کلید تعریف شده است. از طرف دیگر این button درون تگ div قرار دارد که برای این تگ نیز این رویداد کلیک با تابع گرداننده متفاوتی تعریف شده است. نکته این جاست که به صورت پیش فرض بعد از فراخوانی رویداد کلیک عنصر داخلی، رویداد کلیک عنصر خارجی نیز فراخوانی خواهد شد. به این رفتار event bubbling می‌گویند. اگر قصد داشته باشیم که این رفتار را غیر فعال کنیم(یعنی با کلیک بر روی button، رویداد کلیک تگ div اجرا نشود باید مقدار خاصیت clickBubble رویداد عنصر داخلی را برابر false قرار دهیم) به صورت زیر:

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

نظرات خوانندگان

نویسنده: مهرداد اشکانی
تاریخ: ۱۳۹۲/۰۷/۰۳ ۱۵:۵۷

عالی بود دوست عزیز خیلی استفاده کردیم

نویسنده: سعید
تاریخ: ۱۳۹۲/۱۰/۱۶ ۲۳:۵۸

سلام ضمن تشکر از ارایه آموزش فارسی مفیدتون که باعث میشه زمان کمتری برای درک مفهوم صرف بشه اما لطفا درصورت امکان روند را بایک فرآیند عملی قابل درک توضیح دهید چون بعضی اصطلاحات فارسی نمیتونه گویا باشد مثلا در نکته دوم جملات برای من واضح نبود و چون مثال عملی نیست مجبورم که به مطالب زبان اصلی مراجعه کنم تا مفهوم را بهتر درک کنم. من آموزش قبلیتون که با مثال بود را بخوبی درک کردم و از کاراتون تشکر میکنم