بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

شروع کار با Knockout.js

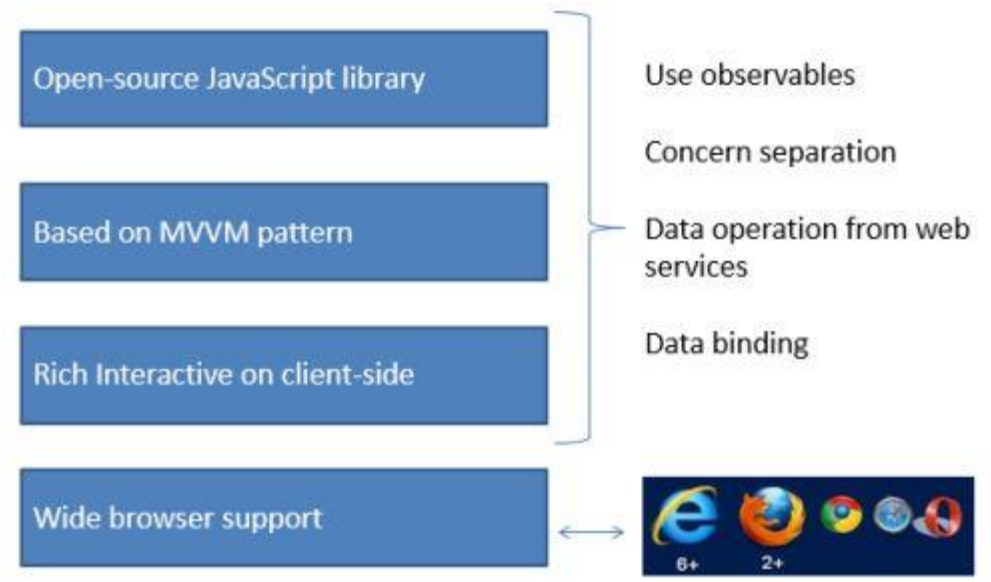
مدرس : مهندس افشین رفوآ

شروع کار با Knockout.js

در این مقاله قصد داریم مروری بر روی یکسری مفاهیم داشته باشیم. از آنجا که این مقاله یک مقدمه برای Knockout.js است، درباره این موضوع صحبت می کنیم که چرا باید از Knockout.js استفاده کنیم و اینکه چطور کار با Knockout را شروع کنیم. اینکه observableها چه هستند، چطور محاسبه می شوند و observableArray چیست؟ سپس مروری بر KO (KO نام دیگر Knockout است)، KO با MVVM خواهیم داشت.

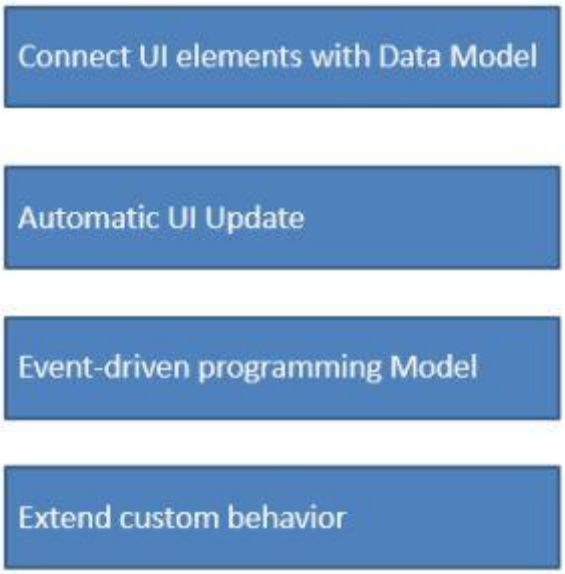
Knockout.js چیست؟

Knockout یک کتابخانه غنی شده در JavaScript است که بوسیله Steven Sanderson، یکی از کارکنان فعلی Microsoft، نوشته شده است. وی تا قبل از آن به استخدام Microsoft در نیامده بود.



KO از الگوی MVVM استفاده می کند. یعنی همان الگوی Model View View Model و باعث می شود یک تعامل client-side قوی فراهم شود. نگران پشتیبانی مرورگر نباشید. در صورتی که از IE 6 یا بالاتر استفاده می کنید، مرورگر شما Knockout را پشتیبانی می کند.

ممکن است این سوال برای شما پیش آمده باشد که استفاده از Knockout چه مزیتی می تواند داشته باشد؟



اول از همه اینکه، می توانید هر زمان که خواستید عناصر UI را با داده های Model ربط بدهید یا به اصطلاح Connect کنید.

می توانید به راحتی یک مدل داده پویا و پیچیده ایجاد کنید.

هر زمان که Data Model تغییر کرد، UI به صورت خودکار بروزرسانی می شود. یا بر عکس هر زمان که UI تغییر می کند، Data Model بروزرسانی می شود.

پشتیبانی از مدل برنامه نویسی رخداد گرا (event-driven)

قابلیت توسعه رفتار دلخواه

حالا باید با سه بخش اصلی Knockout آشنا شوید تا متوجه شوید Knockout چطور به شما کمک خواهد کرد.

اولین جزء Observable

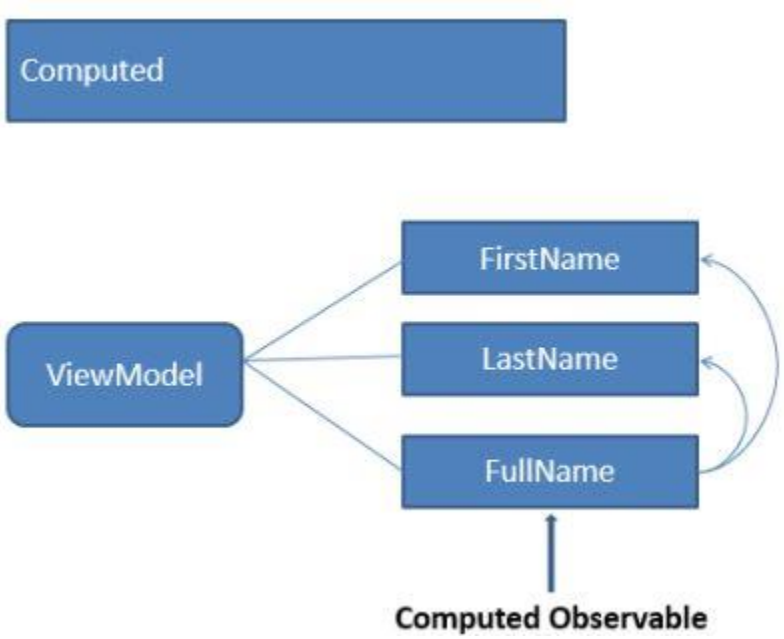
شما باید به Knockout درباره بروزرسانی بخش view model بگوئید و برای این کار باید ویژگی های model تان را به صورت Observable تعریف کنید. مانند مثال زیر:

```
var myViewModel = {
  FirstName: ko.observable("FirstName"),
  LastName: Ko.observable("LastName")

};
```

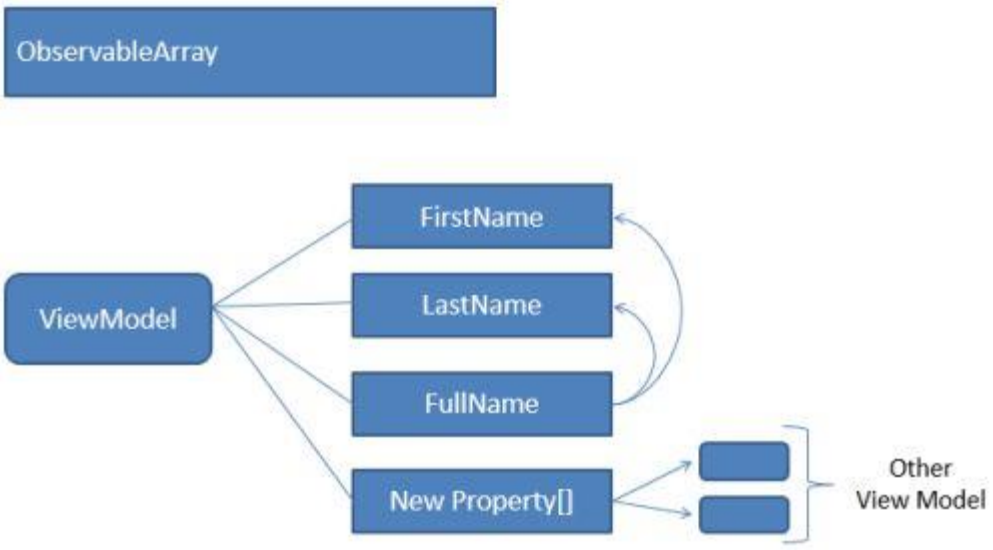
ما یک interface داریم که خصوصیت تغییر I-notify را فراخوانی می کند و می توان با اعمال بر روی کلاس های C# یا VB ، بین خصوصیات این کلاس ها و UI ارتباط برقرار کرد. یعنی زمانی که مقداری را در جعبه متن تایپ می کنیم، این مقدار به صورت خودکار در model و View model پشتیبان ذخیره می شود و به صورت خودکار و همزمان model و UI بروزرسانی می شوند.

دومین جزء Computed



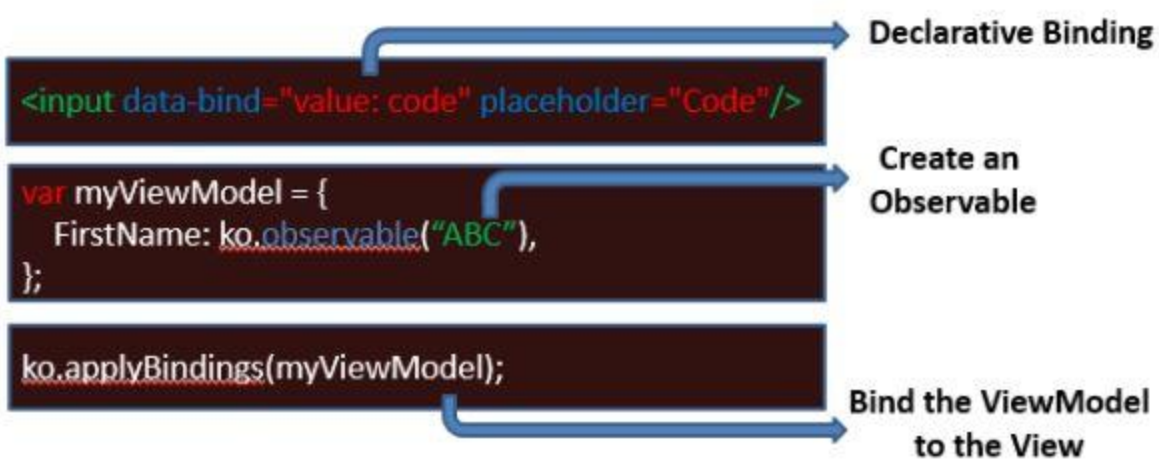
Computed Observableها، یکسری توابع وابسته به یک یا چند observable دیگر هستند و هر زمان که این وابستگی ها تغییر کند، به صورت خودکار بروزرسانی می شوند.

مثلاً شما یک observable برای firstName و یکی دیگر برای lastname دارید و می خواهید نام کامل را نمایش دهید.



ObservableArray زمانی استفاده می شود که بخواهیم تغییرات یک مجموعه را تشخیص داده و به آن پاسخ دهیم. اساساً زمانی استفاده می شود که می خواهید یکسری مقادیر را پس از نمایش ویرایش کنید. به این ترتیب بعضی از بخش های UI، با هر بار حذف یا اضافه شدن عناصر، مکرراً نمایش داده می شوند و یا ناپدید می شوند.

حالا سوال این است که چطور این کارها را با Knockout.js انجام دهیم؟ به سادگی و فقط در سه مرحله:



اول از همه در HTML باید اتصال را اعلان کنیم (declarative binding).

سپس در کد JavaScript، observable را ایجاد میکنیم.

در آخر هم ViewModel را به View متصل می کنیم.

بیایید بفهمیم الگوی MVVM چیست. فهم این موضوع به شما کمک می کند تا متوجه شوید چطور Knockout.js data binding بین JavaScript و HTML انجام می شود.

در Model داده های برنامه نگهداری می شود و View وظیفه نمایش داده های model به کاربر را به عهده دارد. در آخر ViewModel ارتباط بین View و Model را فراهم میکند.

Knockout.js با استفاده از observableها، فرآیند اتصال داده MVVM را ساده تر می شود. به این صورت که خصوصیات Viewmodel بسته به تغییرات view، به صورت خودکار تغییر میکنند.

اجازه دهید تا با هم محیط توسعه برای کار با Knockout.js را نصب کنیم و یک demo کوچک از آن داشته باشیم.

برای نصب محیط توسعه در Visual Studio :

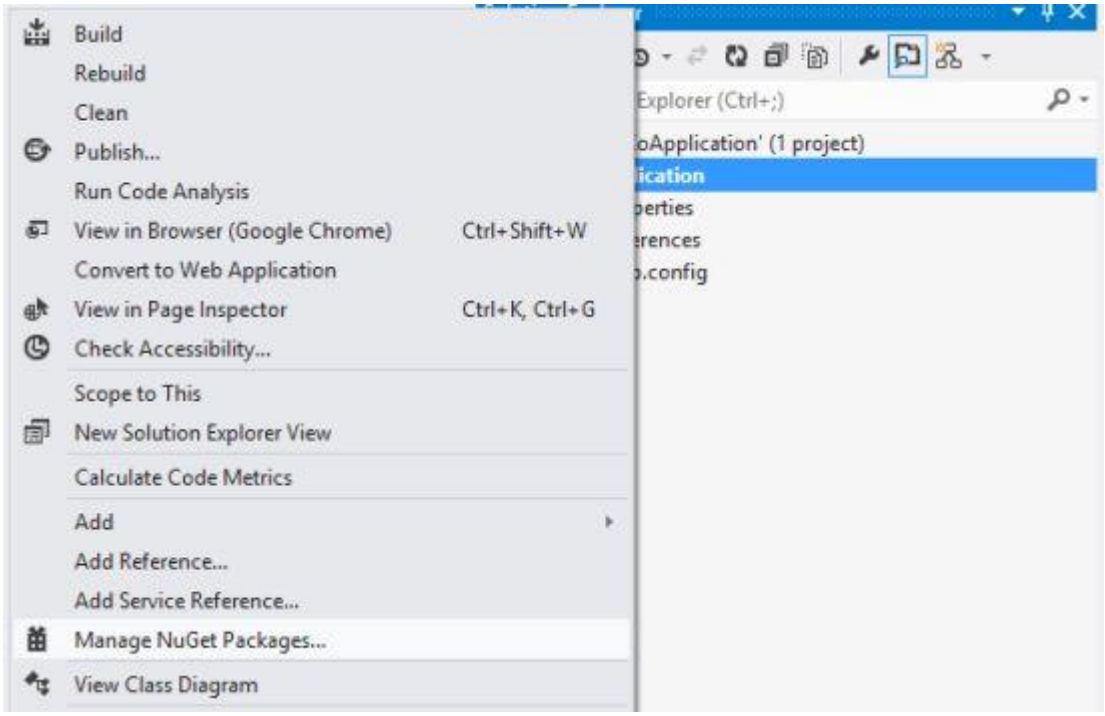
یک ASP.NET Empty Web Application ایجاد کنید.

کتابخانه Knockout را به پروژه اضافه کنید:

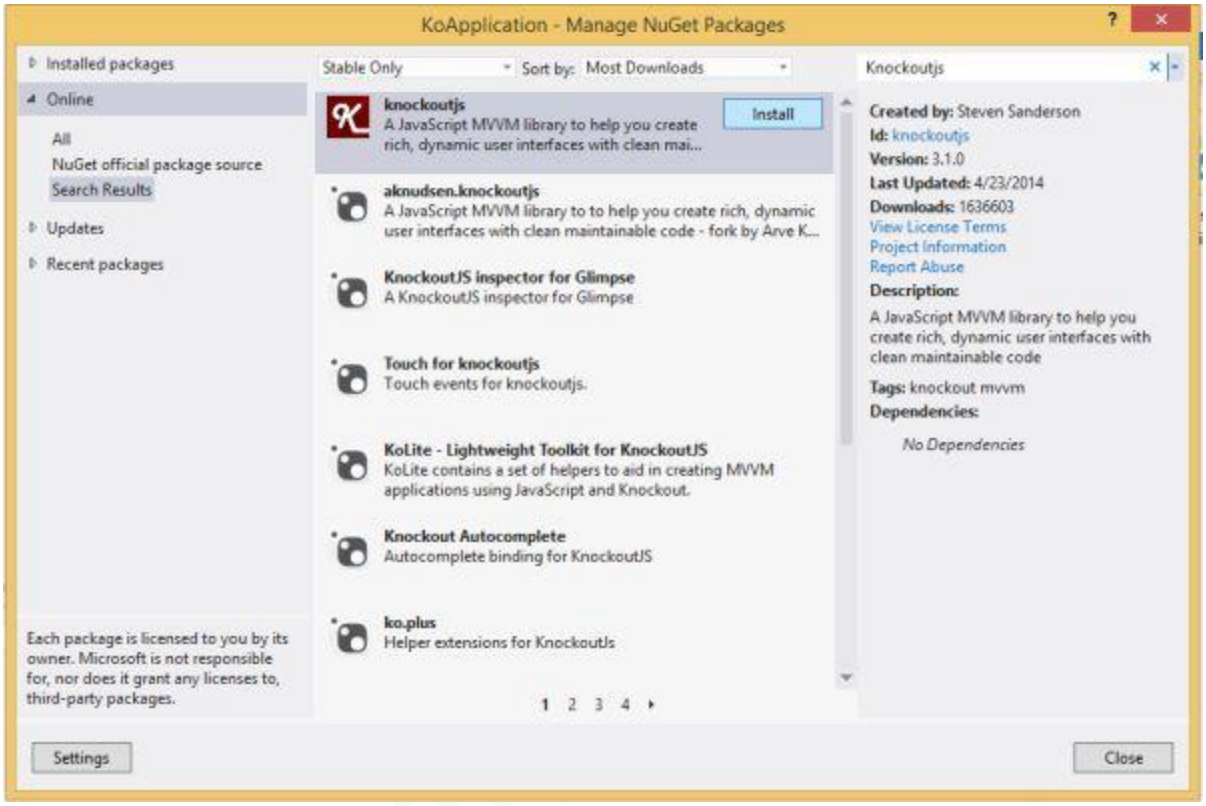
می توانید این کار را به صورت دستی با اضافه کردن فایل های KO JS به پروژه انجام دهید.

با استفاده از NuGet، فایل های reference را به پروژه اضافه کنید:

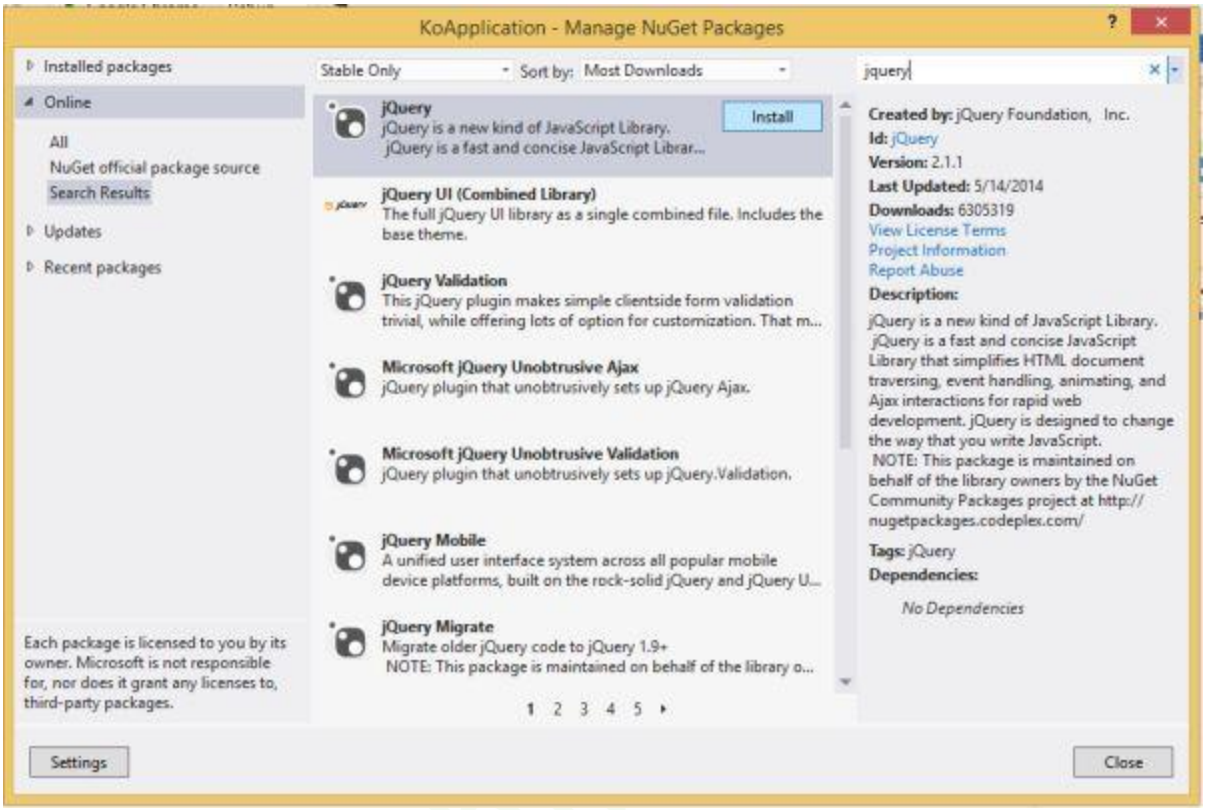
من روش دوم را انتخاب کرده ام. با راست کلیک بر روی پروژه، گزینه Manage NuGet Package را انتخاب کنید.



Knockout را جستجو کنید. گزینه knockoutjs در بالا نمایش داده می شود. بر روی Install کلیک کنید.



پس از نصب knockout، اینبار به دنبال jQuery بگردید و آن را نیز در پروژه خود نصب کنید.

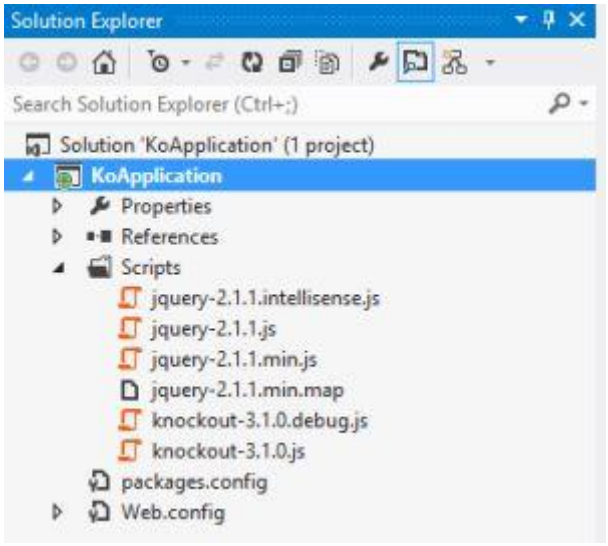


محیط برای پروژه ما آماده شده است.

برای مشاهده چگونگی عملکرد Knockoutjs:

یک فایل HTML به پروژه خود اضافه کنید.

در بخش Head فایل ایجاد شده یک ارجاع به فایل های knockoutjs و jQuery بدهید.




```
<head>
    <title>My first Ko Application</title>
    <script src="Scripts/knockout-3.0.0.js"></script>
    <script src="Scripts/jquery-2.0.3.js"></script>
</head>
```

View چیزی جز عناصر HTML نیست. برای بخش View:

اول، مانند زیر، دو فیلد برای نمایش مقادیر ایجاد کنید.

```
<p>First Name :<strong data-bind="text:initialName"></strong></p>
<p>Last Name :<strong data-bind="text:lastName"></strong></p>
```

سپس دو کنترل HTML input برای ورود مقادیر در نظر می گیریم. باید با استفاده از ویژگی data-bind (در HTML 5) مقادیر را به داده های منبع داده متصل کنیم.

```
<input data-bind="value:initialName" />
<input data-bind="value:lastName" />
```

برای بخش ViewModel :

باید یک تابع JavaScript بنویسیم تا خصوصیت اتصال دوطرفه یک شی را فعال کنیم. خصوصیات model را به صورت observable اعلان میکنیم. به این ترتیب داده ها در زمان اجرا هم خاصیت اتصال خود را حفظ کرده و ویرایش و بروزرسانی می شوند.

```
function koapp() {
  this.initialName = ko.observable();
  this.lastName = ko.observable();

};
```

سپس با استفاده از applyBinding، این تابع را فراخوانی می کنیم.

```
ko.applyBindings(new koapp());
```

کد کامل شده:

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <title>My first Ko Application</title>
  <script src="Scripts/knockout-3.0.0.js"></script>
  <script src="Scripts/jquery-2.0.3.js"></script>
</head>
<body>
  <p>First Name :<strong data-bind="text:initialName"></strong></p>
  <p>Last Name :<strong data-bind="text:lastName"></strong></p>
  <input data-bind="value:initialName" />
  <input data-bind="value:lastName" />
  <script type="text/ecmascript">
    function koapp() {
      this.initialName = ko.observable();
      this.lastName = ko.observable();
    };
    ko.applyBindings(new koapp());
  </script>
</body>
</html>
```

با اجرای برنامه خواهیم داشت:

زمانی که شما مقدار هر یک از فیلد های ورودی را تغییر دهید، مقدار برچسب بلافاصله تغییر می کند.

بنابراین دیدیم که اتصال دو طرفه و تازه سازی خودکار UI به راحتی بوسیله Knockoutjs قابل پیاده سازی است.