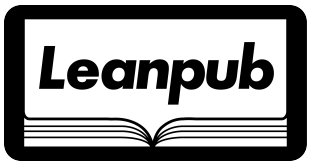
DevTyr

Knockout.js

Knapp und gut: Knockout.js

Einführung und Praxis

This version was published on 2012-07-20



This is a Leanpub book, for sale at:

http://leanpub.com/knockoutjs

Leanpub helps authors to self-publish in-progress ebooks. We call this idea Lean Publishing. To learn more about Lean Publishing, go to: http://leanpub.com/manifesto

To learn more about Leanpub, go to: http://leanpub.com

Tweet This Book!

Please help Norbert Eder by spreading the word about this book on Twitter!

The suggested hashtag for this book is #knockoutjsebook.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

https://twitter.com/search/#knockoutjsebook

Inhaltsverzeichnis

1 Einleitung

Bei Knockout.js[1] handelt es sich um eine JavaScript-Bibliothek, die Anleihen beim Model-View-ViewModel (MVVM) Pattern genommen hat. Die Hauptintention liegt darin, eine klare Trennung zwischen Benutzerschnittstelle und der Implementierung zu schaffen. Hierzu auch die durch das Pattern gedachte Aufteilung. Dieses Kapitel klärt alle Voraussetzungen, gibt einen Überblick der einzelnen Bestandteile des MVVM-Patterns und erklärt die Grundzüge von Datenbindungen.

1.1 Voraussetzungen

Um Knockout.js einsetzen zu können wird nicht viel benötigt. Das volle Paket wird auf github[2] gehostet. Es enthält sämtlichen Code und Build-Scripts. Dadurch können Sie sich selbst einen Überblick über die Implementierung verschaffen, als auch Kompatibilitätstests durchführen.

Um nun ein Projekt auf Basis Knockout.js umzusetzen benötigen Sie lediglich die aktuellste Version, die auf der Download-Seite[3] zu finden ist. Diese wird in zwei Varianten angeboten:

- Die komprimierte Variante dient dem Einsatz in Produktivumgebungen. Sie wurde so verkleinert, dass sie möglichst wenig Overhead beim Laden einer Webseite erzeugt.
- Eine Debug-Variante. Diese liegt in lesbarer Form vor und kann für das Debuggen verwendet werden.

Zusätzlich kann jquery.tmpl.js verwendet werden, eine Template-Engine, die standardmäßig von Knockout.js unterstützt wird. Diese ist ebenfalls im Download-Bereich zu finden, ist jedoch als optional anzusehen.

1.2 Browserunterstützung

Da bereits das Thema Browserunterstützung angesprochen wurde, die Information, welche Browser im Detail unterstützt werden. Nachfolgend findet sich die zum aktuellen Zeitpunkt gültige Liste.

- Apple Safari (Windows Version 5+, Mac OS X Version 3.1.2+)
- Google Chrome 5+
- Microsoft Internet Explorer 6+
- Mozilla Firefox 2+
- Opera 10+

[1] http://knockoutjs.com
[2] https://github.com/SteveSanderson/knockout
[3] https://github.com/SteveSanderson/knockout/downloads

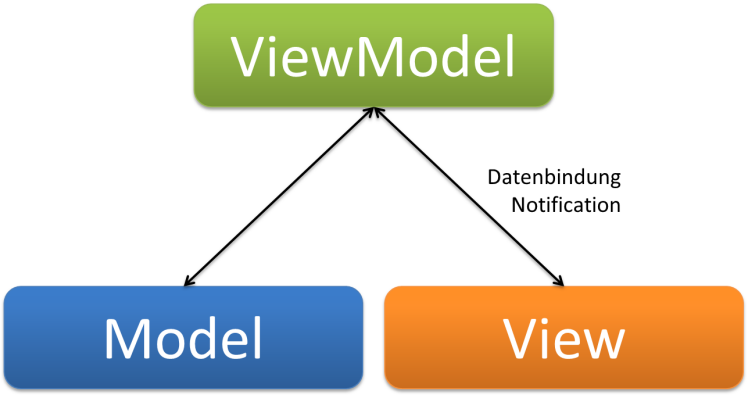
Wenn Sie den in den Voraussetzungen erwähnten Sourcecode herunter geladen haben, können Sie für spezifische Browser vorgefertigte Tests durchführen um ein Bild über die Lauffähigkeit von Knockout.js zu erhalten. Öffnen Sie dazu `/spec/runner.html` im Browser. Laufen alle Tests erfolgreich durch, kann Knockout.js eingesetzt werden.

1.3 Bestandteile MVVM

Wie schon eingangs erwähnt, beruht Knockout.js auf dem MVVM-Pattern. Dieses kommt ursprünglich aus der Welt der Windows Presentation Foundation[4] (kurz WPF). Das Pattern besteht aus drei voneinander getrennten Teilen:

- Model
- View
- ViewModel

Bevor die Einzelteile genauer beschrieben werden, werfen Sie einen Blick auf die nachfolgende Abbildung, welche die Zusammenhänge der einzelnen Teile grafisch veranschaulichen soll.



Übersicht Model-ViewModel-View

Sehen wir uns die Bestandteile des MVVM-Patterns genauer an. Der erfolgreiche Einsatz von Knockout.js beruht auf ein klares Verständnis für dieses Pattern.

1.3.1 Model

Das Model ist für gewöhnlich als reines Datenobjekt zu sehen. Darin werden die Daten gehalten, mit denen das ViewModel und die View arbeiten. Im Falle von Knockout.js steht das Model hauptsächlich am Server zur Verfügung oder wird als einfaches Data Transfer Object (DTO)

[4] http://msdn.microsoft.com/de-de/library/ms754130.aspx

durch einen AJAX[5]-Aufruf übertragen und als JavaScript Object Notation (kurz JSON[6]) (de-)serialisiert. Mit diesen Daten wird in weiterer Folge das ViewModel befüllt. Nachfolgendes Listing zeigt ein einfaches JSON-serialisiertes Model.

```
{
  "person": {
    "firstName": "Norbert",
    "lastName": "Eder"
  }
}
```

1.3.2 View

Die View ist der Part, der für den Besucher der Website sichtbar ist, im Endeffekt also HTML-Markup. Sie enthält sämtliches Markup und verweist auf alle notwendigen JavaScript-Dateien, als auch alle Stylesheets. Sie selbst sollte so wenig Logik als möglich enthalten. Lediglich Logik, die sich rein auf die View bezieht ist an dieser Stelle zu empfehlen. Zusätzlich enthält die View Bindungen an das ViewModel.

1.3.3 ViewModel

Das ViewModel stellt sowohl die Daten, als auch alle möglichen Aktionen zur Verfügung. Kenner des Patterns MVC werden diesen Teil als Controller identifizieren, womit sie auch nicht falsch liegen. An die zur Verfügung gestellten Daten und Aktionen bindet die View. Dies wird zur Gänze von Knockout.js übernommen, es besteht jedoch genügend Eingriffsmöglichkeit. Dadurch entsteht eine klare Trennung der Zuständigkeiten. Nachfolgendes Listing zeigt ein einfaches Beispiel eines ViewModels mit zwei Attributen und einer Methode.

Info: Im gezeigten Beispiel scheint noch nichts Spezifisches aus Knockout.js auf.

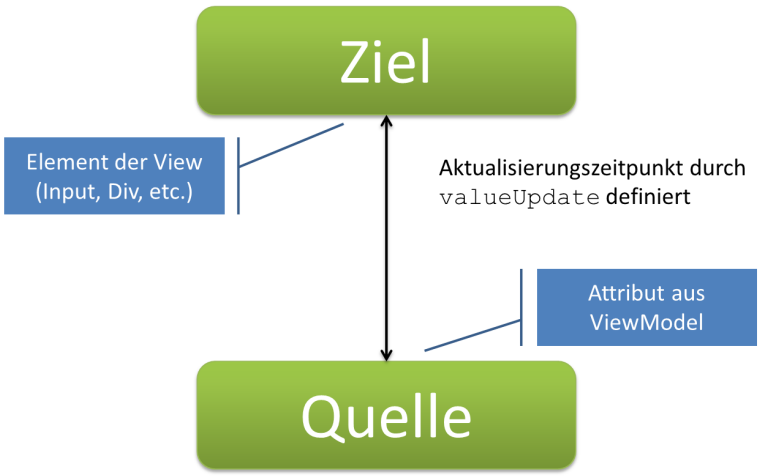
```
var personViewModel = {
    firstName: '';
    lastName: '';

    doSomething: new function() {
      alert('I did something!');
    };
};
```

[5] http://de.wikipedia.org/wiki/Ajax_(Programmierung)
[6] http://de.wikipedia.org/wiki/JSON

1.3.4 Bindungen

Ein wesentlicher Bestandteil sind Bindungen. Eine Bindung stellt eine Verbindung zwischen Quelle und Ziel her. Die Quelle ist der Teil, der den Wert ursprünglich zur Verfügung stellt, an dieser Stelle ein ViewModel. Das Ziel ist in Knockout.js eine View. Sobald sich der Wert in der Quelle ändert, wird das Ziel automatisch nachgezogen und spiegelt diesen wider. Je nach Einstellung ist ein Durchschreiben von geänderten Werten an die Quelle ebenfalls möglich. Hierzu bietet Knockout.js mehrere Möglichkeiten an, die etwas später genauer besprochen werden.

Durch Datenbindungen kann sehr viel infrastruktureller Code vermieden werden. Dieser würde benötigt werden, um Formular-Elemente (Eingabefelder etc.) mit Werten zu füllen bzw. die Werte wieder daraus auszulesen. Hinzuzufügen ist ebenfalls, dass auf sämtliche Eigenheiten der unterschiedlichen Browser (und deren Versionen) Rücksicht genommen werden muss. Ein klarer Vorteil, wenn uns dies durch eine Bibliothek abgenommen wird. Finden Sie eine grundlegende Übersicht der Bestandteile und Zusammenhänge von Bindungen in der folgenden Abbildung:



Bindungen