KNOCKOUTJS И ANGULARJS. СРАВНЕНИЕ ФРЕЙМВОРКОВ

Белорусский государственный университет информатики и радиоэлектроники
г. Минск, Республика Беларусь

Паршиков В. А.

Скудняков Ю. А. – канд. техн. наук, доцент

В настоящее время веб приложения становятся все более и более сложными. Что обязывает использовать какий либо библиотеки или фреймворки, одними из которых являются KnockoutJS и AngularJS

AngularJS и KnockoutJS очень близки по своей идеологии. Они являются фреймворками для динамических веб-приложений и используют HTML в качестве шаблона. Они позволяют расширить синтаксис HTML для того, чтобы описать компоненты вашего приложения более ясно и лаконично. Из коробки они устраняют необходимость писать код, который раньше создавался для реализации связи model-view-controller. AngularJS и KnockoutJS — это по сути то, чем HTML и JavaScript были бы, если бы они разрабатывались для создания современных веб-приложений. HTML — это прекрасный декларативный язык для статических документов. Но, к сожалению, в нем нет многого, что необходимо для создания современных веб-приложений.

Особенности:

- Data-binding: простой и хороший способ связи UI и модели данных;
- мощный набор инструментов для разработчика (в частности у AngularJS, KnockoutJS имеет место достаточно бедный набор);
- легко расширяемый инструментарий.

Архитектура приложения:

Согласно документации, Angular предлагает структурировать приложение, разделяя его на модули. Каждый модуль состоит из: функции, конфигурирующей модуль (она запускается сразу после загрузки модуля); контроллера; сервисов; директив.

Контроллер в понимании Angular — это функция, которая конструирует модель данных. Для создания модели используется сервис $scope, но о нем немного дальше. Директивы — это расширения для HTML.

В свою очередь, Knockout предлагает строить приложение, разделяя его на ModelView, которые являются миксом из модели и контроллера. В пределах объекта ko.bindingHandlers размещены data-bindings, которые являются аналогами директив Angular. Для построения связи между моделью и ее представлением используются observable и observableArray.

Говоря о модульности, нельзя не вспомнить про шаблон AMD — Asynchronous Module Definition. Angular и Knockout не имеют собственной реализации AMD шаблона. Советуем использовать библиотеку RequireJS. Она себя очень хорошо зарекомендовала в плане совместимости и с Angular, и с Knockout.

На данный момент уже существует огромное количество шаблонизаторов (к примеру, jQuery Templates, к сожалению, уже не поддерживается). Большинство из них работают по принципу: возьми статический template как string, смешай его с данными, создав новую строку, и полученную строку вставь в необходимый DOM-элемент посредством innerHTML - свойства. Такой подход означает рендеринг темплейта каждый раз после какого-либо изменения данных. В данном подходе существует ряд известных проблем, к примеру: чтение вводимых пользователем данных и соединение их с моделью, потеря пользовательских данных из-за их перезаписи, управление всем процессом обновления данных и/или представления. Кроме того, данный подход может негативно сказываться на производительности.

Angular и Knockout используют иной подход. А именно two-way binding. Отличительная особенность данного подхода — это создание двунаправленной связи элемента страницы с элементами модели. Такой подход позволяет получить достаточно стабильный DOM. В Knockout двунаправленная связь реализована посредством функций observable и observableArray. Для анализа шаблона используется HTML парсер jQuery (если подключен, в противном случае аналогичный родной парсер). Результатом работы упомянутых функций является функция, которая инкапсулирует текущее состояние элемента модели и отвечает за two-way binding. Данная реализация не очень удобна, поскольку возникает проблема, связанная с копированием состояния модели: скоуп функции не копируются, поэтому необходимо сперва получить данные из элемента модели, обратившись к нему, как к функции и только после этого клонировать результат.

В Angular двунаправленная связь строится непосредственно компилятором (сервис $compile). Разработчику нет необходимости использовать функции, подобные observable. Это намного удобнее, поскольку нет необходимости использовать дополнительные конструкции и не возникает проблемы при копировании состояния элемента модели.

Ключевой же разницей в реализации шаблонизаторов в Angular и Knockout является способ рендеринга элементов: Angular генерирует DOM-элементы, которые потом использует; Knockout — генерирует строки и innerHTML-ит их. Поэтому генерация большого числа элементов занимает у Knockout больше времени (наглядный пример немного ниже).

Говоря о модели данных в Angular, обязательно стоит остановиться на сервисе $scope. Этот сервис содержит ссылки на модель данных. Поскольку Angular предполагает наличие достаточно сложной архитектуры приложения, $scope также имеет более сложную структуру.

Внутри каждого модуля создается новый экземпляр $scope, который является наследником $rootScope. Существует возможность программно создать новый экземпляр $scope из существующего. В таком случае созданный экземпляр будет наследником того $scope, из которого он был создан. Разобраться с иерархией

$scope в Angular не составит труда для тех, кто хорошо знает JavaScript. Такая возможность очень удобна, когда есть необходимость создания различных widgets, к примеру pop-ups. Перейдем к вопросу производительности. Были произведены 2 теста: холодный старт приложения “Hello World!” и рендеринг массива из 1000 элементов.

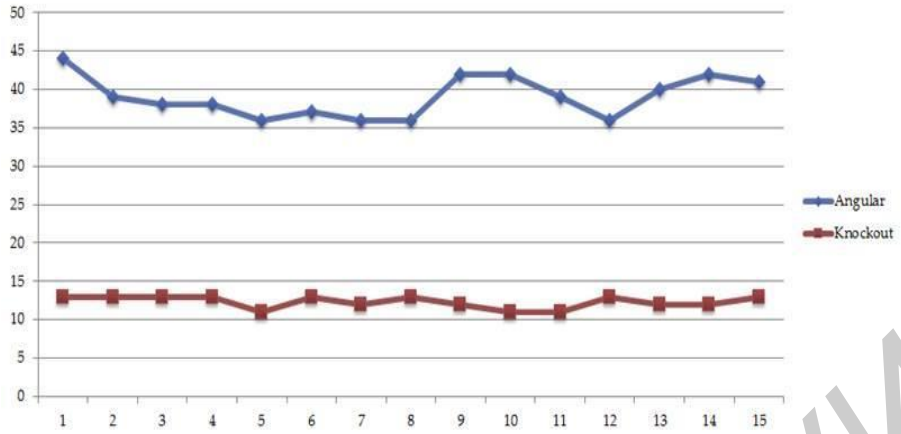

Рис. 1 — Холодный старт приложения “Hello World!”

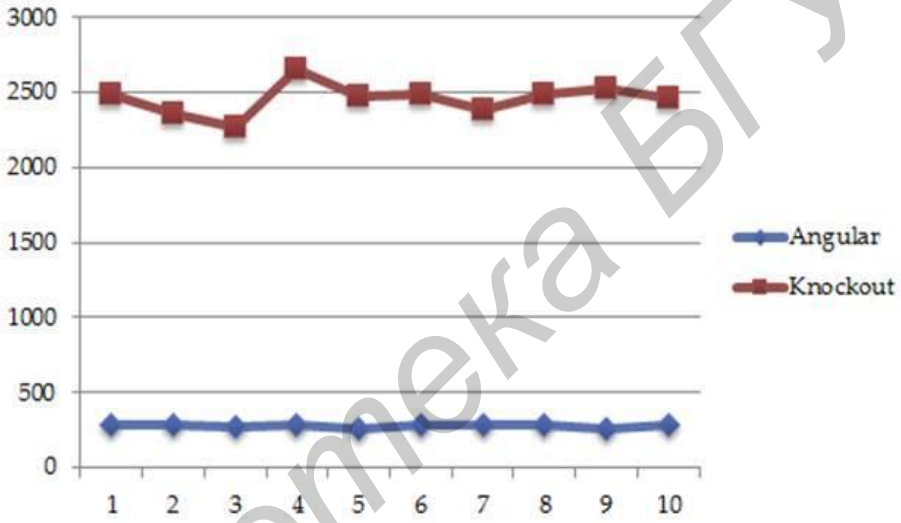

Рис. 2 — Рендеринг массива из 1000 элементов

На всех схемах по вертикали — миллисекунды, по горизонтали номер эксперимента. Здесь хорошо видно, что холодный старт у Knockout происходит намного быстрее, чем у Angular.

А вот когда речь заходит о рендеринге, здесь очевидно лидирует Angular. Как мы видим, для рендеринга 1000 строк Knockout тратит до 2,5 секунд, в то же время Angular хватает меньше 500 миллисекунд для выполнения этой задачи. Кроме того, отображение отрендеринговых элементов на экране пользователя также занимает разное время: для Angular это 1-3 секунды, а для Knockout — 14-20 секунд. Это происходит из-за того, что Knockout генерирует строки, а Angular — DOM-элементы.

Самый главный вопрос заключается в определении области применения Angular и Knockout. Проведя несколько простых экспериментов, были сформулированы следующие выводы:

Knockout применим в случаях, когда нет необходимости в создании сложной архитектуры, сложных workflow-ов. Его основная функция — связь модели и представления, поэтому его лучше всего использовать для простых одностраничных приложений (к примеру, создание различного уровня сложности форм).

Относительно Angular можно сделать вывод, что он будет полезен в тех случаях, когда требуется создание RichUI.

Список использованных источников

1. AngularJS [Электронный ресурс].– Режим доступа: http://angularjs.org/. – Дата доступа 18.02.2014.
2. KnockoutJS [Электронный ресурс]. – Режим доступа: http://knockoutjs.com/. – Дата доступа 18.02.2014.