Angular и TypeScript

Сайтостроение для профессионалов

Яков Файн
Антон Моисеев

MANNING

Angular 2 Development with TypeScript

YAKOV FAIN
ANTON MOISEEV

MANNING
Shelter Island

Яков Файн, Антон Моисеев

Angular и TypeScript

Сайтостроение для профессионалов

ПИТЕР®
Санкт-Петербург • Москва • Екатеринбург • Воронеж
Нижний Новгород • Ростов-на-Дону • Самара • Минск

2018

Яков Файн, Антон Моисеев

Angular и TypeScript. Сайтостроение для профессионалов

Серия «Библиотека программиста»

Перевели с английского *Н. Вильчинский, Е. Зазноба*

Заведующая редакцией	*Ю. Сергиенко*
Руководитель проекта	*О. Сивченко*
Ведущий редактор	*Н. Гринчик*
Литературный редактор	*Н. Хлебина*
Художественный редактор	*С. Заматевская*
Корректоры	*Е. Павлович, Е. Рафалюк-Бузовская*
Верстка	*Г. Блинов*

ББК 32.988.02
УДК 004.738.5

Файн Я., Моисеев А.

Ф17 Angular и TypeScript. Сайтостроение для профессионалов. — СПб.: Питер, 2018. — 464 с.: ил. — (Серия «Библиотека программиста»).

ISBN 978-5-4461-0496-3

Если вы занимаетесь веб-разработкой — от веб-клиентов до полнофункциональных одностраничных приложений, — то фреймворк Angular будет для вас просто спасением. Этот ультрасовременный инструмент полностью интегрирован со статически типизированным языком TypeScript, который отлично вписывается в экосистему JavaScript.

Вы научитесь:

- Проектировать и строить модульные приложения.
- Правильно транспилировать TypeScript в JavaScript.
- Пользоваться новейшими инструментами JavaScript — в частности npm, Karma и Webpack.

Если вам знаком язык JavaScript — берите и читайте! Знаний TypeScript или AngularJS для изучения книги не требуется.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ISBN 978-1617293122 англ.
ISBN 978-5-4461-0496-3

© 2017 by Manning Publications Co. All rights reserved
© Перевод на русский язык ООО Издательство «Питер», 2018
© Издание на русском языке, оформление ООО Издательство «Питер», 2018
© Серия «Библиотека программиста», 2018

Права на издание получены по соглашению с Manning. Все права защищены. Никакая часть данной книги не может быть воспроизведена в какой бы то ни было форме без письменного разрешения владельцев авторских прав.

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги.

Изготовлено в России. Изготовитель: ООО «Прогресс книга».
Место нахождения и фактический адрес: 191123, Россия, город Санкт-Петербург,
улица Радищева, дом 39, корпус Д, офис 415. Тел.: +78127037373.
Дата изготовления: 12.2017. Наименование: книжная продукция. Срок годности: не ограничен.
Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12 —
Книги печатные профессиональные, технические и научные.
Подписано в печать 08.12.17. Формат 70×100/16. Бумага офсетная. Усл. п. л. 37,410. Тираж 1000. Заказ 0000.
Отпечатано в ОАО «Первая Образцовая типография». Филиал «Чеховский Печатный Двор».
142300, Московская область, г. Чехов, ул. Полиграфистов, 1.
Сайт: www.chpk.ru. E-mail: marketing@chpk.ru
Факс: 8(496) 726-54-10, телефон: (495) 988-63-87

Краткое содержание

Оглавление

Предисловие

Мы принялись подбирать хороший фреймворк для JavaScript примерно четыре года назад. Тогда мы работали над платформой для страховой компании и большая часть интерфейса системы была написана с помощью фреймворка Apache Flex (ранее известного как Adobe Flex). Это отличный инструмент для разработки веб-интерфейсов, но он требует наличия Flash Player, с некоторых пор впавшего в немилость.

После попыток создать несколько вводных проектов, написанных на JavaScript, мы обнаружили заметное снижение производительности наших разработчиков. Задача, выполнить которую с использованием Flex можно было за один день, требовала три дня работы в других фреймворках JavaScript, включая AngularJS. Основная причина спада — нехватка типов в JavaScript, недостаточная поддержка IDE и отсутствие поддержки компилятора.

Когда мы узнали, что компания Google начала разработку фреймворка Angular с TypeScript в качестве рекомендуемого языка, мы взяли данный фреймворк на вооружение летом 2015 г. В то время о нем было известно немногое. В нашем распоряжении была лишь информация из некоторых блогов и видеороликов, записанных на конференции, где команда разработчиков Angular представляла новый фреймворк. Нашим основным источником информации был его исходный код. Но вскоре мы обнаружили у Angular отличный потенциал, поэтому решили начать создавать с его помощью обучающее ПО для отдела разработки нашей компании. В то же время Майк Стивенс (Mike Stephens), сотрудник издательства Manning, искал авторов, заинтересованных в написании книги об Angular. Именно так и появилось это издание.

После года работы с Angular и TypeScript мы можем подтвердить: этот дуэт предоставляет наиболее продуктивный способ создания средних и крупных веб-приложений, которые могут работать в любом современном браузере, а также на мобильных платформах. Перечислим основные причины, позволившие нам прийти к такому выводу.

- *Четкое разделение интерфейса и логики.* Код, который отвечает за пользовательский интерфейс, и код, реализующий логику приложения, четко разделены. Интерфейс не должен создаваться в HTML, существуют другие продукты с поддержкой его нативной отрисовки для iOS и Android.
- *Модульность.* Существует простой механизм разбиения приложения на модули с возможностью их ленивой загрузки.
- *Поддержка навигации.* Маршрутизатор поддерживает сложные сценарии навигации в одностраничных приложениях.
- *Слабое связывание.* Внедрение зависимостей (Dependency Injection, DI) позволяет установить связь между компонентами и сервисами. С помощью связывания

и событий есть возможность создавать слабо связанные компоненты, которые можно использовать повторно.

- *Жизненный цикл компонентов*. Каждый компонент проходит через тщательно определенный жизненный цикл, разработчикам доступны процедуры, позволяющие перехватывать важные события компонентов.
- *Определение изменений*. Автоматический (и быстрый) механизм определения изменений дает возможность отказаться от обновления интерфейса вручную, также предоставляя способ выполнить тонкую настройку этого процесса.
- *Отсутствие ада обратных вызовов*. Angular поставляется вместе с библиотекой RxJS, которая позволяет построить процесс обработки асинхронных данных, основанный на подписке, что помогает избежать появления ада обратных вызовов.
- *Формы и валидация*. Поддержка форм и пользовательская валидация хорошо спроектированы. Вы можете создавать формы, добавляя директивы к элементам формы как в шаблонах, так и программно.
- *Тестирование*. Фреймворк поддерживает модульное (блочное) и сквозное тестирование, вы можете интегрировать тесты в свой автоматизированный процесс сборки.
- *Упаковка и оптимизация с помощью Webpack*. Упаковка и оптимизация кода с помощью Webpack (и различных плагинов для него) позволяет поддерживать небольшой размер развертываемых приложений.
- *Инструментальные средства*. Инструментальные средства поддерживаются так же хорошо, как и для платформ Java и .NET. Анализатор кода TypeScript указывает на ошибки по мере набора текста, а инструмент для создания временных платформ и развертывания (Angular CLI) позволяет избегать написания стереотипного кода и сценариев конфигурирования.
- *Лаконичный код*. Использование классов и интерфейсов TypeScript делает код более лаконичным, а также упрощает его чтение и написание.
- *Компиляторы*. Компилятор TypeScript генерирует код JavaScript, который человек может прочитать. Код TypeScript может быть скомпилирован для версий JavaScript ES3, ES5 или ES6. Компиляция перед выполнением (Ahead-of-time, AoT) кода Angular (не путать с компилированием TypeScript) помогает избавиться от необходимости поставлять с вашим приложением компилятор для Angular, что еще больше снижает накладные расходы, требуемые для фреймворка.
- *Отрисовка на стороне сервера*. Angular Universal превращает ваше приложение в HTML на этапе сборки кода офлайн. Данный код может быть использован для отрисовки на стороне сервера, что, в свою очередь, значительно улучшает индексирование поисковыми системами и SEO.

Таким образом, Angular 2 готов к работе сразу после установки.

С точки зрения управления этот фреймворк может быть привлекательным, поскольку в мире более миллиона разработчиков предпочитают AngularJS и большинство из них переключатся на Angular. Эта версия была выпущена в сентябре

2016 г., и каждые полгода будут выпускаться новые крупные релизы. Команда разработчиков Angular потратила два года на разработку Angular 2.0, и к моменту выхода фреймворка более полумиллиона разработчиков уже использовали его. Наличие большого количества работников с определенными навыками — важный фактор, который следует учитывать при выборе технологий для новых проектов. Кроме того, платформами Java или .NET пользуются более 15 миллионов разработчиков, и многие из них сочтут синтаксис TypeScript гораздо более привлекательным, чем синтаксис JavaScript, благодаря поддержке классов, интерфейсов, обобщенных типов, аннотаций, переменных членов класса, а также закрытых и открытых переменных, не говоря о полезном компиляторе и поддержке известных IDE.

Писать книгу об Angular было трудно, поскольку мы начали работу с ранних альфа-версий фреймворка, которые изменились по мере перехода к бета- и релиз-версиям. Но нам нравится результат, и мы уже начали разрабатывать с помощью Angular проекты для решения задач реальной сложности.

Предисловие к русскому изданию

В эпоху бурного развития Интернета требования к сложности и качеству сайтов постоянно растут. Пользователям и организациям уже недостаточно простых статических страниц. В связи с этим давно наметился спрос на полноценные веб-приложения, однако до недавнего времени он удовлетворялся в основном за счет таких закрытых технологий, как Adobe Flex и JavaFX.

Около десяти лет назад ситуация начала меняться. Последовательная стандартизация языка JavaScript привела к появлению множества инструментов для веб-разработки, которые не требуют дополнений и поддерживаются во всех современных браузерах. Пожалуй, самым известным из них стал фреймворк Angular.

Angular разрабатывается компанией Google и при этом является полностью бесплатным и открытым. Это устоявшийся программный продукт, который пользуется большой популярностью как на корпоративном рынке, так и в небольших студиях. Количество программистов, работающих с ним, давно перевалило за миллион.

Эта книга посвящена Angular 2 — современной версии данного фреймворка. Он вобрал в себя все последние веяния в мире веб-разработки, сохранив традиционные черты, характерные для первой версии. Главной особенностью Angular 2 является его изначальная ориентированность на TypeScript. И хотя поддержка JavaScript (ES5 и ES6) по-прежнему присутствует, TypeScript остается основным языком разработки. Это, в сущности, надстройка над ECMAScript 6 со строгой типизацией. Развитая система типов и программных интерфейсов позволяет писать более безопасный и выразительный код, который легче сопровождать. Кроме того, TypeScript имеет отличную поддержку в современных средах разработки.

Еще одной ключевой особенностью Angular 2 является четкое разделение интерфейса и бизнес-логики. Это позволяет создавать компоненты, которые могут работать в том числе на таких мобильных операционных системах, как iOS и Android. Angular 2 позволяет разбивать код на модули, которые загружаются по мере необходимости и содержат описание своих зависимостей. Это значительно упрощает процедуру тестирования (как модульного, так и сквозного) и облегчает интеграцию тестов в процесс сборки.

Разработчикам доступен богатый инструментарий для написания и развертывания приложений. Автодополнение кода, упаковка и оптимизация, отрисовка на стороне сервера, компиляция (статическая и на лету) — все это становится возможным благодаря современным технологиям, на которых основаны Angular 2 и TypeScript. В частности, для управления проектами предусмотрена утилита командной строки Angular CLI, а для создания гибких пользовательских интерфейсов компания Google предоставляет библиотеку компонентов Angular Material 2.

На протяжении двух лет разработкой Angular 2 занимались сотни программистов. Все старания были направлены на то, чтобы вы получили современный инструмент, полностью готовый к работе сразу после установки. Этот фреймворк покрывает собой все аспекты создания сложных веб-приложений, начиная с обработки событий и навигации и заканчивая развертыванием в промышленных масштабах. Все это и многое другое подробно рассматривается на страницах книги.

Благодарности

Оба автора хотели бы поблагодарить издательство Manning Publications и Алана М. Кауниота (Alain M. Couniot) за его предложения по улучшению технической стороны книги и Коди Сэнда (Cody Sand) — за техническую редактуру. Благодарим также следующих рецензентов, предоставивших ценную обратную связь: Криса Коппенбаргера (Chris Coppenbarger), Дэвида Баркола (David Barkol), Дэвида Ди Мария (David DiMaria), Фредрика Энгберга (Fredrik Engberg), Ирака Илиша Рамоса Эрнандеса (Irach Ilish Ramos Hernandez), Джереми Брайана (Jeremy Bryan), Камала Раджа (Kamal Raj), Лори Уилкинс (Lori Wilkins), Мауро Керциоли (Mauro Quercioli), Себастьяна Нишеля (Sébastien Nichèle), Полину Кесельман (Polina Keselman), Субира Растоги (Subir Rastogi), Свена Лесекана (Sven Lösekann) и Висама Цагала (Wisam Zaghal).

Яков хотел бы поблагодарить своего лучшего друга Сэмми (Sammy) за создание теплой и комфортной атмосферы во время работы над этой книгой. К сожалению, Сэмми не умеет разговаривать, но, безусловно, любит Якова. Порода Сэмми — золотистый ретривер.

Антон хотел бы поблагодарить Якова Файна (Yakov Fain) и издательство Manning Publications за полученную возможность стать соавтором книги и приобрести бесценный писательский опыт. Он также благодарен своей семье за терпение, которое она проявляли, пока он работал над книгой.

Об этой книге

Программы на Angular могут быть написаны с помощью двух версий JavaScript (ES5 и ES6), а также на языках Dart или TypeScript. Сам фреймворк разработан с использованием TypeScript, и именно его мы будем задействовать во всех примерах нашей книги. В приложении Б вы найдете раздел «Зачем создавать Angular-приложения[1] на TypeScript», где мы объясняем причины, по которым выбрали этот язык.

Поскольку мы по своей природе практики, то и книгу писали для практиков. Мы не только объясним особенности фреймворка, приводя простые примеры кода, но и на протяжении нескольких глав покажем, как создать одностраничное приложение для онлайн-аукционов.

Во время написания и редактирования данной книги мы провели несколько семинаров с использованием примеров кода. Это позволило получить раннюю (и крайне положительную) обратную связь об изложенном материале. Мы очень надеемся на то, что вам понравится изучать Angular.

Как читать книгу

Ранние версии книги начинались с глав, посвященных ECMAScript 6 и TypeScript. Некоторые редакторы предложили переместить этот материал в приложения, чтобы читатели смогли быстрее приступить к изучению Angular. Мы согласились с таким изменением. Но если вы еще не знакомы с синтаксисом ECMAScript 6 и TypeScript, то информация, представленная в приложениях, поможет разобраться в примерах кода каждой главы.

Структура издания

Книга состоит из десяти глав и двух приложений.

В главе 1 приводится обзор архитектуры Angular, кратко рассматриваются популярные фреймворки и библиотеки JavaScript. Вы также познакомитесь с приложением для онлайн-аукционов, которое начнете разрабатывать в главе 2.

Вы будете разрабатывать эту программу с помощью TypeScript. Информация, изложенная в приложении Б, позволит приступить к работе с этим замечательным языком, представляющим собой расширенный набор функций языка JavaScript. Вы узнаете не только о том, как писать классы, интерфейсы и обобщенные классы, но и как скомпилировать код TypeScript для современных версий JavaScript, которые могут быть развернуты во всех браузерах. В TypeScript

[1] Здесь и далее: имеется в виду приложение, созданное с помощью Angular. — *Примеч. ред.*

реализована большая часть синтаксиса спецификации ECMAScript 6 (она рассматривается в приложении А), а также синтаксис, который будет включен в будущие релизы ECMAScript.

В главе 2 вы начнете разрабатывать простые приложения с помощью Angular и создадите свои первые Angular-компоненты. Вы узнаете, как работать с загрузчиком модулей SystemJS, и мы предложим вам свою версию стартового проекта Angular, который может быть использован в качестве отправной точки для всех примеров приложений этой книги. В конце главы вы создадите первую версию главной страницы для онлайн-аукциона.

Глава 3 посвящена маршрутизатору Angular, предлагающему гибкий способ настройки навигации в одностраничных приложениях. Вы узнаете, как сконфигурировать маршруты для компонентов-предков и компонентов-потомков, передавать данные из одного маршрута в другой и выполнять «ленивую» загрузку модулей. В конце главы вы проведете рефакторинг для онлайн-аукциона, разбив его на несколько элементов и добавив для них маршрутизацию.

В главе 4 представлена информация о шаблоне (паттерне) «Внедрение зависимостей» и его реализации в Angular. Вы познакомитесь с концепцией поставщиков, которая позволит указать, как создавать объекты внедряемых зависимостей. В новой версии онлайн-аукциона вы примените внедрение зависимостей для наполнения данными представления Product Details (Информация о продукте).

В главе 5 мы рассмотрим разные виды привязки данных, познакомим вас с реактивным программированием с помощью наблюдаемых потоков данных и покажем, как работать с каналами. Глава заканчивается новой версией онлайн-аукциона: в нее добавлен наблюдаемый поток событий, используемый для фильтрации популярных продуктов на главной странице.

Глава 6 посвящена разработке компонентов, которые могут общаться друг с другом в слабо связанной манере. Мы рассмотрим их входные и выходные параметры, шаблон проектирования «Посредник», а также жизненный цикл компонента. Кроме того, в этой главе приводится обзор механизма обнаружения изменений, представленного в Angular. Онлайн-аукцион приобретет функциональность оценки продуктов.

Глава 7 посвящена обработке форм в Angular. После рассмотрения основ Forms API мы обсудим валидацию форм и реализуем ее в поисковом компоненте для еще одной версии онлайн-аукциона.

В главе 8 мы объясним, как клиентское Angular-приложение может взаимодействовать с серверами, используя протоколы HTTP и WebSocket, и рассмотрим примеры. Вы создадите серверное приложение, применяя фреймворки Node.js и Express. Далее вы развернете фрагмент онлайн-аукциона, написанный на Angular, на сервере Node. С помощью клиентской части (front end) аукциона можно будет начать общаться с сервером Node.js с применением протоколов HTTP и WebSocket.

Глава 9 посвящена модульному тестированию. Мы рассмотрим основные принципы работы с Jasmine и библиотекой для проверки Angular. Вы узнаете, как тестировать сервисы, компоненты и маршрутизатор, сконфигурировать и использовать

Karma для запуска тестов, а также реализуете несколько модульных тестов для онлайн-аукциона.

Глава 10 посвящена автоматизации процессов сборки и развертывания. Вы увидите, как можно применять упаковщик Webpack для минимизации и упаковки вашего кода для развертывания.

Кроме того, вы узнаете, как использовать Angular CLI для генерации и развертывания проектов. Размер развернутой версии онлайн-аукциона снизится с 5,5 Мбайт (в разработке) до 350 Кбайт (на производстве).

В приложении А вы познакомитесь с новым синтаксисом, добавленным в ECMAScript 2015 (также известным как ES6). Приложение Б содержит вводную информацию о языке TypeScript.

Соглашения, принятые в этой книге, и материалы для скачивания

В данной книге содержится множество примеров исходного кода, который отформатирован с использованием `моноширинного шрифта`, чтобы выделить его на фоне обычного текста. Нередко оригинальный исходный код переформатирован; добавлены разрывы строк, а также переработаны отступы с учетом свободного места на страницах. Иногда, когда этого оказалось недостаточно, листинги содержат метки продолжения строки (➥). Кроме того, комментарии к исходному коду были удалены там, где код описан в тексте. Во многих листингах приведены комментарии, указывающие на важные концепции.

Исходный код примеров книги доступен для загрузки с сайта издателя https://www.manning.com/books/angular-2-development-with-typescript.

Авторы также создали репозиторий на GitHub, в котором содержатся примеры исходного кода, на https://github.com/Farata/angular2typescript. Если в результате выхода новых релизов Angular какие-то примеры перестанут работать, то на указанной странице вы можете рассказать об этом и авторы постараются разобраться с проблемой.

Об авторах

Яков Файн (Yakov Fain) является сооснователем двух компаний: Farata Systems и SuranceBay. Farata Systems — фирма, занимающаяся IT-консалтингом. Яков руководит отделом обучения и проводит семинары по Angular и Java по всему миру. SuranceBay — производственная компания, которая автоматизирует различные рабочие потоки в отрасли страхования в Соединенных Штатах, предлагая приложения, использующие ПО как услугу (Software as a Service, SaaS). В этой компании Яков руководит различными проектами.

Яков — большой энтузиаст Java. Он написал множество книг, посвященных разработке ПО, а также более тысячи статей в блоге по адресу yakovfain.com. Кроме того, его книга *Java Programming for Kids, Parents and Grandparents* («Программирование на Java для детей, родителей, бабушек и дедушек») доступна для бесплатного скачивания на нескольких языках по адресу http://myflex.org/books/java4kids/java4kids.htm. Его никнейм в Twitter — @yfain.

Антон Моисеев (Anton Moiseev) — ведущий разработчик ПО в компании SuranceBay. Он более десяти лет занимается разработкой промышленных приложений с помощью технологий Java и .NET и имеет богатый опыт создания многофункциональных интернет-приложений для разных платформ. Сфера деятельности Антона — веб-технологии, причем он специализируется на приемах, позволяющих фронтенду и бэкенду без проблем взаимодействовать друг с другом. Он провел множество занятий, посвященных фреймворкам AngularJS и Angular 2.

Время от времени Антон пишет статьи в блоге по адресу antonmoiseev.com. Его никнейм в Twitter — @anton-moiseev.

1 Знакомство с Angular

В этой главе:

- краткий обзор фреймворков и библиотек JavaScript;
- общий обзор Angular 1 и 2;
- набор инструментов для Angular-разработчика;
- пример приложения.

Angular 2 — фреймворк с открытым исходным кодом, написанный на JavaScript и поддерживаемый компанией Google. Он представляет собой полностью переработанную версию своего популярного предшественника, AngularJS. С помощью Angular вы можете разрабатывать приложения на JavaScript (применяя синтаксис ECMAScript 5 или 6), Dart или TypeScript. В книге мы будем использовать TypeScript. Причины, по которым был выбран именно этот синтаксис, описаны в приложении Б.

ПРИМЕЧАНИЕ

Мы не ждем от вас опыта работы с AngularJS, но рассчитываем, что вы знакомы с синтаксисом JavaScript и HTML, а также понимаете, из чего состоит веб-приложение. Кроме того, предполагается, что вы имеете представление о CSS и знакомы с ролью объекта DOM в браузере.

Мы начнем эту главу с очень краткого обзора нескольких популярных фреймворков JavaScript. Далее рассмотрим архитектуру AngularJS и Angular 2, выделяя улучшения, привнесенные в новую версию. Кроме того, кратко рассмотрим инструменты, используемые Angular-разработчиками. Наконец, мы взглянем на приложение-пример, которое будем создавать на протяжении книги.

ПРИМЕЧАНИЕ

Наша книга посвящена фреймворку Angular 2, и для краткости мы будем называть его Angular. Если мы упомянем AngularJS, то это значит, что речь идет о версиях 1.x данного фреймворка.

1.1. Примеры фреймворков и библиотек JavaScript

Обязательно ли использовать фреймворки? Нет, можно написать клиентскую часть веб-приложений на чистом JavaScript. В этом случае не нужно изучать что-то новое, достаточно знания языка JavaScript. Отказ от фреймворков приведет к возникновению трудностей при поддержке совместимости между браузерами, а также к увеличению циклов разработки. Фреймворки же позволяют полностью управлять архитектурой, шаблонами проектирования и стилями кода вашего приложения. Большая часть современных веб-приложений написаны при сочетании нескольких фреймворков и библиотек.

В этом разделе мы кратко рассмотрим популярные фреймворки и библиотеки для работы с JavaScript. В чем заключается разница между ними? *Фреймворки* позволяют структурировать ваш код и заставляют писать его определенным способом. *Библиотеки* обычно предлагают несколько компонентов и API, которые могут быть использованы по желанию в любом коде. Другими словами, фреймворки предоставляют большую гибкость при разработке приложения.

Angular — один из многих фреймворков, применяемых для разработки веб-приложений.

1.1.1. Полноценные фреймворки

Содержат все, что понадобится для разработки веб-приложения. Они предоставляют возможность легко структурировать ваш код и поставляются вместе с библиотекой, содержащей компоненты и инструменты для сборки и развертывания приложения.

Например, *Ext JS* — полноценный фреймворк, созданный и поддерживаемый компанией Sencha. Он поставляется вместе с полным набором UI-компонентов, включающим продвинутые сетки данных и таблицы (их наличие критически важно для разработки промышленных приложений для офисов). Ext JS значительно увеличивает объем кода программы — вам не удастся найти созданное с его помощью приложение, которое весит меньше 1 Мбайт. Кроме того, данный фреймворк довольно глубоко внедряется — будет трудно при необходимости переключиться на другой инструмент.

Sencha также предлагает фреймворк Sencha Touch, используемый при создании веб-приложений для мобильных устройств.

1.1.2. Легковесные фреймворки

Позволяют структурировать веб-приложение, предлагают способ настройки навигации между представлениями, а также разбивают приложения на слои, реализуя шаблон проектирования «Модель — Представление — Контроллер» (Model — View — Controller, MVC). Кроме того, существует группа легковесных фреймворков, которые специализируются на тестировании приложений, написанных на JavaScript.

Angular — фреймворк с открытым исходным кодом, предназначенный для разработки веб-приложений. Упрощает создание пользовательских компонентов, которые могут быть добавлены в документы HTML, а также реализацию логики приложения. Активно использует привязку данных, содержит модуль внедрения зависимостей, поддерживает модульность и предоставляет механизм для настройки маршрутизации. AngularJS был основан на шаблоне MVC, в отличие от Angular. Последний не содержит элементов для создания пользовательского интерфейса.

Ember.js — это фреймворк с открытым исходным кодом, основанный на MVC; служит для разработки веб-приложений. Содержит механизм маршрутизации и поддерживает двухстороннюю привязку данных. В коде этого фреймворка используется множество соглашений, что повышает продуктивность разработчиков ПО.

Jasmine — фреймворк с открытым исходным кодом, предназначенный для тестирования кода JavaScript. Не требует наличия объекта DOM. Содержит набор функций, проверяющих, ведут ли себя части приложения запланированным образом. Нередко используется вместе с Karma — программой для запуска тестов, которая позволяет проводить проверки в разных браузерах.

1.1.3. Библиотеки

Библиотеки, рассмотренные в этом подразделе, служат для разных целей и могут быть задействованы в веб-приложениях вместе с другими фреймворками или самостоятельно.

jQuery — популярная библиотека для JavaScript. Довольно проста в использовании и не требует значительного изменения стиля написания кода для веб-программ. Позволяет находить объекты DOM и манипулировать ими, а также обрабатывать события браузера и справляться с несовместимостью браузеров. Это расширяемая библиотека, разработчики со всего мира создали для нее тысячи плагинов. Если вы не можете найти плагин, который отвечает вашим нуждам, то всегда можете создать его самостоятельно.

Bootstrap — библиотека компонентов для создания пользовательского интерфейса с открытым исходным кодом, разработанная компанией Twitter. Они строятся согласно принципам адаптивного веб-дизайна, что значительно повышает ценность библиотеки, если ваше веб-приложение должно автоматически подстраивать свой макет в зависимости от размера экрана устройства пользователя. В этой книге мы будем использовать Bootstrap при разработке приложения-примера.

ПРИМЕЧАНИЕ

> В компании Google была разработана библиотека компонентов под названием Material Design, которая может стать альтернативой Bootstrap. Она оптимизирована для использования на разных устройствах и поставляется вместе с набором интересных элементов пользовательского интерфейса.

React — созданная компанией Facebook библиотека с открытым исходным кодом, предназначенная для сборки пользовательских интерфейсов. Представляет

собой слой V в аббревиатуре MVC. Не внедряется глубоко, и ее можно применять вместе с любой другой библиотекой или фреймворком. Создает собственный виртуальный объект DOM, минимизируя доступ к объекту DOM браузера, в результате чего повышается производительность. Что касается отрисовки содержимого, React вводит формат JSX — расширение синтаксиса JavaScript, которое выглядит как XML. Использование JSX рекомендуется, но не обязательно.

Polymer — библиотека, созданная компанией Google для сборки пользовательских компонентов на основе стандарта Web Components. Поставляется вместе с набором интересных настраиваемых элементов пользовательского интерфейса, которые можно включить в разметку HTML в виде тегов. Кроме того, содержит компоненты приложений, предназначенных для работы в режиме офлайн, а также элементы, использующие разнообразные API от Google (например, календарь, карты и др.).

RxJS — набор библиотек, необходимых для создания асинхронных программ и программ, основанных на событиях, с использованием наблюдаемых коллекций. Позволяет приложениям работать с асинхронными потоками данных наподобие серверного потока котировок акций или событий, связанных с движением мыши. С помощью RxJS потоки данных представляются в виде наблюдаемых последовательностей. Эту библиотеку можно применять как с другими фреймворками JavaScript, так и без них. В главах 5 и 8 вы увидите примеры использования наблюдаемых последовательностей в Angular.

Чтобы увидеть статистику, которая показывает, на каких сайтах задействованы те или иные фреймворки, вы можете посетить страницу BuiltWith: http://trends.builtwith.com/javascript.

Переход от Flex к Angular

Мы работаем на компанию Farata Systems, которая за много лет разработала много довольно сложного ПО, используя фреймворк Flex от Adobe. Это очень продуктивный фреймворк, созданный на основе строго типизированного скомпилированного языка ActionScript. Приложения, написанные с его помощью, развертываются в плагине для браузера Flash Player (применяется как виртуальная машина). Когда веб-сообщество начало отходить от плагинов, мы потратили два года, пытаясь найти замену Flex. Мы экспериментировали с разными фреймворками, основанными на JavaScript, но эффективность труда наших разработчиков серьезно падала. Наконец, мы увидели свет в конце туннеля, когда объединили язык TypeScript, фреймворк Angular и библиотеку для работы с пользовательским интерфейсом, такую как Angular Material.

1.1.4. Что такое Node.js

Node.js (или просто *Node*) — не просто фреймворк или библиотека. Это еще и среда времени выполнения. На протяжении большей части книги мы будем использовать время выполнения Node для запуска различных утилит наподобие Node Package Manager (npm). Например, для установки TypeScript можно запустить npm из командной строки:

```
npm install typescript
```

Фреймворк Node.js используется для разработки программ на языке JavaScript, которые функционируют вне браузера. Вы можете создать серверный слой веб-приложения на JavaScript или Typescript; в главе 8 вы напишете веб-сервер, применяя Node. Компания Google разработала для браузера Chrome высокопроизводительный движок JavaScript V8. Его можно задействовать для запуска кода, написанного с помощью API Node.js. Фреймворк Node.js включает в себя API для работы с файловой системой, доступа к базе данных, прослушивания запросов HTTP и др.

Члены сообщества JavaScript создали множество утилит, которые будут полезны при разработке веб-приложений. Используя движок JavaScript для Node, вы можете запускать их из командной строки.

1.2. Общий обзор AngularJS

Теперь вернемся к основной теме нашей книги: фреймворку Angular. Этот раздел — единственный, посвященный AngularJS.

Перечислим причины, по которым AngularJS стал таким популярным.

- Фреймворк содержит механизм создания пользовательских тегов и атрибутов HTML с помощью концепции директив, которая позволяет расширять набор тегов HTML в соответствии с потребностями приложения.
- AngularJS не слишком глубоко внедряется. Вы можете добавить атрибут `ng-app` к любому тегу `<div>`, и AngularJS будет управлять содержимым лишь этого тега. Остальная часть веб-страницы может быть написана на чистом HTML и JavaScript.
- Фреймворк позволяет легко связывать данные с представлениями. Изменение данных приводит к автоматическому обновлению соответствующего элемента представления, и наоборот.
- AngularJS поставляется с настраиваемым маршрутизатором; он разрешает соотносить шаблоны URL с соответствующими компонентами приложения, которые изменяют представление на веб-странице в зависимости от установленных соотношений.
- Поток данных приложения определяется в *контроллерах*, они являются объектами JavaScript, содержащими свойства и функции.
- Приложения, созданные с помощью AngularJS, используют иерархию *областей видимости* — объектов, необходимых для сохранения данных, общих для контроллеров и представлений.
- Фреймворк содержит модуль внедрения зависимостей, который позволяет разрабатывать слабо связанные приложения.

В то время как в jQuery были упрощены манипуляции с объектом DOM, AngularJS позволяет разработчикам отвязывать логику приложения от интерфейса путем структурирования приложения по шаблону проектирования MVC. На рис. 1.1 показан пример рабочего потока приложения, созданного с помощью этого фреймворка.

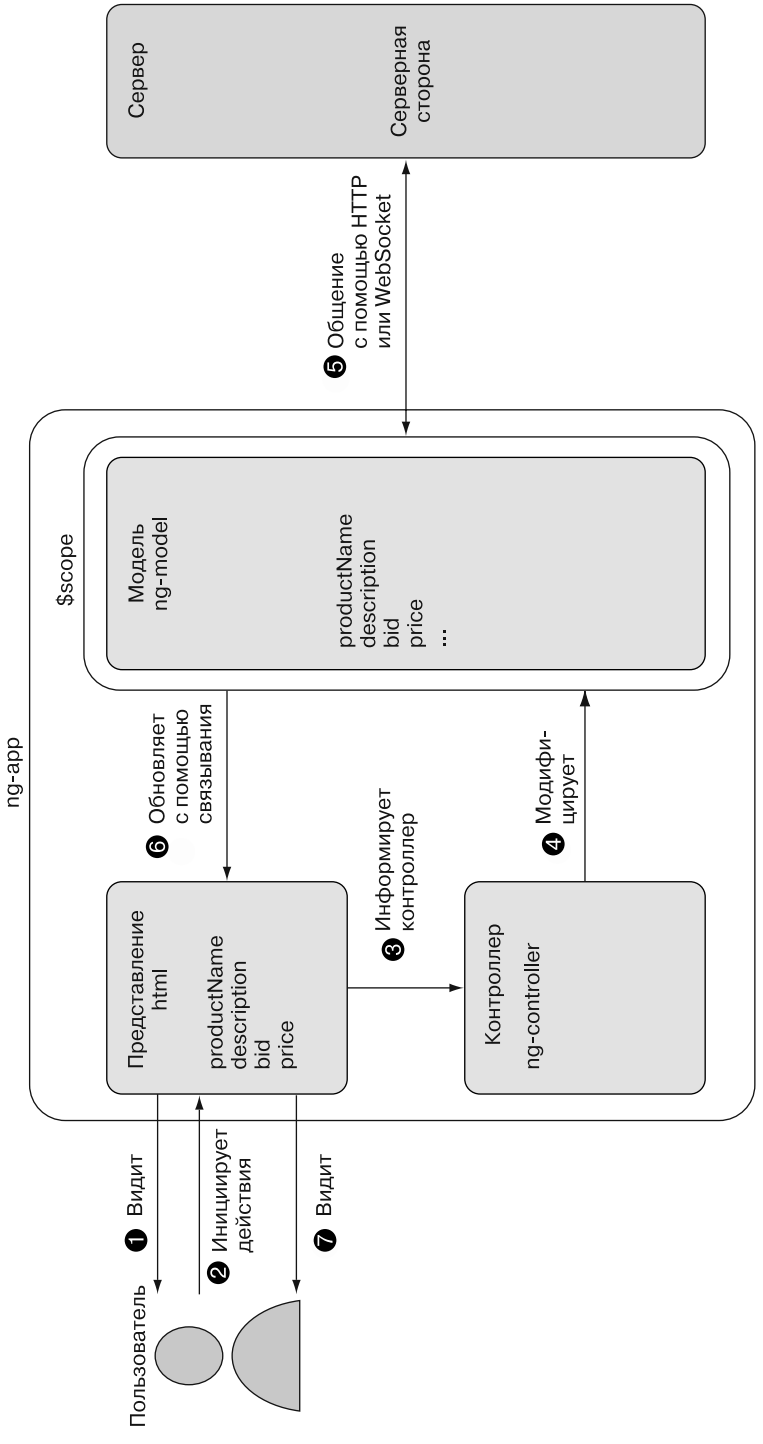

Рис. 1.1. Пример архитектуры приложения, написанного с использованием AngularJS

AngularJS может управлять всем веб-приложением. Для этого включите директиву `ng-app` в HTML-тег `<body>`:

```
<body ng-app="ProductApp">
```

На рис. 1.1, чтобы получить данные о продукте, пользователь загружает приложение ❶ и вводит идентификатор продукта ❷. Представление оповещает контроллер ❸, который обновляет модель ❹ и выполняет HTTP-запрос ❺ на удаленный сервер, используя сервис `$http`. AngularJS заполняет свойства модели на основе полученных данных ❺, и изменения в модели автоматически отразятся в пользовательском интерфейсе с помощью *связывающего выражения* ❻. После этого пользователь увидит данные о запрошенном продукте ❼.

AngularJS автоматически обновляет представление при модификации данных модели. Изменения в пользовательском интерфейсе вносятся в модель, если пользователь меняет данные в элементах управления представления, связанных с вводом. Такой двунаправленный механизм обновлений называется *двухсторонней привязкой*, он показан на рис. 1.2.

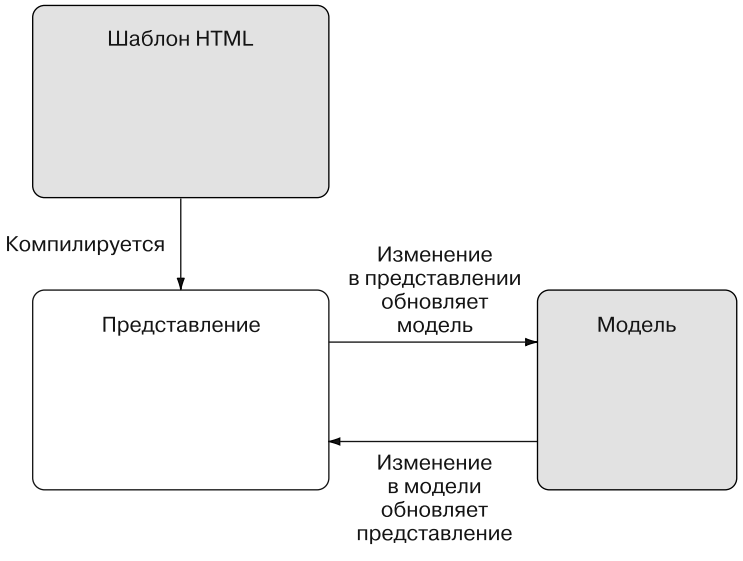

Рис. 1.2. Двухсторонняя привязка

В AngularJS модель и представление тесно связаны, поскольку двухсторонняя привязка означает, что один из элементов пары автоматически обновляет другой. Подобная возможность автоматического обновления очень удобна, но имеет свою цену.

При каждом обновлении модели фреймворк запускает специальный цикл `$digest`, который проходит по всему приложению, применяет привязку данных

и обновляет DOM, когда это необходимо. Каскадные обновления приводят к нескольким запускам цикла `$digest`, что может повлиять на производительность крупных приложений, использующих двухстороннюю привязку.

Манипуляции с объектом DOM браузера — самая медленная операция. Чем меньше приложение обновляет DOM, тем лучше оно работает.

Данные модели существуют в контексте определенного объекта `$scope`, области видимости AngularJS формируют иерархию объектов. Элемент `$rootScope` создается для целого приложения. Контроллеры и директивы (пользовательские компоненты) имеют собственные объекты `$scope`, и принципы работы областей видимости в фреймворке могут быть сложны для понимания.

Вы можете реализовать модульность, создавая и загружая объекты `module`. Когда конкретный модуль зависит от других объектов (таких как контроллеры, модули или сервисы), экземпляры этих объектов создаются и *внедряются* путем использования механизма внедрения зависимостей, представленного в AngularJS. В следующем фрагменте кода показывается один из способов, которым фреймворк внедряет один объект в другой:

```
var SearchController = function ($scope) {
        //..
};

SearchController['$inject'] = ['$scope'];

angular.module('auction').controller('SearchController', SearchController);
```

Определяет SearchController как функцию-конструктор, имеющую параметр $scope

Добавляет свойство $inject в контроллер, давая команду внедрить объект $scope в функцию-конструктор

Указывает, что объект SearchController должен стать контроллером модуля аукциона

В этом фрагменте кода квадратными скобками обозначен массив, и AngularJS может внедрять сразу несколько объектов следующим образом: `['$scope', 'myCustomService']`.

Данный фреймворк зачастую используется для создания одностраничных приложений, в которых лишь отдельные фрагменты страницы (подпредставления) обновляются в результате действий пользователя или из-за отправки данных на сервер. Хорошим примером таких подпредставлений является веб-приложение, показывающее котировки: при продаже акции обновляется лишь элемент, содержащий значение цены.

Навигация между представлениями в AngularJS выполняется с помощью конфигурирования компонента маршрутизатора `ng-route`. Можно указать количество параметров `.when`, чтобы направить приложение к соответствующему представлению в зависимости от шаблона URL. В следующем фрагменте кода

маршрутизатору предписывается использовать разметку из файла `home.html` и контроллер `HomeController`, если только URL не содержит элемент `/search`. В этом случае представление отрисует страницу `search.html`, и в качестве контроллера будет применен объект `SearchController`:

```
angular.module('auction', ['ngRoute'])
    .config(['$routeProvider', function ($routeProvider) {
        $routeProvider
          .when('/', {
             templateUrl: 'views/home.html',
             controller: 'HomeController' })
          .when('/search', {
              templateUrl: 'views/search.html',
              controller: 'SearchController' })
          .otherwise({
              redirectTo: '/'
    });
}]);
```

Маршрутизатор фреймворка поддерживает глубокое связывание, которое представляет собой способность поместить в закладки не только целую веб-страницу, но и определенное состояние внутри нее.

Теперь, после общего обзора AngularJS, взглянем, что нам может предложить Angular 2.

1.3. Общий обзор Angular

Фреймворк Angular гораздо производительнее, чем AngularJS. Он более понятен для освоения; архитектура приложений была упрощена, и код с его помощью легче читать и писать.

В этом разделе приводится краткий обзор Angular, в котором выделяются его более сильные стороны относительно AngularJS. Подробный архитектурный обзор Angular доступен в документации продукта на https://angular.io/guide/architecture.

1.3.1. Упрощение кода

Во-первых, Angular-приложение содержит стандартные модули в форматах ECMAScript 6 (ES6), Asynchronous Module Definition (AMD) и CommonJS. Обычно один модуль находится в одном файле. Нет необходимости прибегать к синтаксису, характерному для фреймворков, для загрузки и использования модулей. Запустите универсальный загрузчик модулей SystemJS (он рассматривается в главе 2) и добавьте операторы импорта, чтобы применить функциональные возможности, реализованные в загруженных модулях. Вам не нужно волноваться

насчет правильного использования тегов `<script>` в ваших файлах HTML. Если для модуля A требуются функции, содержащиеся в модуле B, просто импортируйте модуль B в модуль A.

Файл HTML для посадочной страницы вашего приложения содержит модули Angular и их зависимости. Код приложения загружается путем загрузки корневого модуля приложения. Все необходимые компоненты и сервисы будут загружены на основании объявлений в операторах `module` и `import`.

В следующем фрагменте кода показано типичное содержимое файла `index.html` приложения, созданного с помощью Angular, куда вы можете включить требуемые модули фреймворка. В сценарий `systemjs.config.js` входит конфигурация загрузчика SystemJS. Вызов `System.import('app')` загружает высокоуровневый элемент приложения, сконфигурированный в файле `systemjs.config.js` (показан в главе 2). Пользовательский тег `<app>` представляет собой значение, определенное в свойстве `selector` корневого компонента:

```
<!DOCTYPE html>
<html>
<head>
  <title>Angular seed project</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <script src="node_modules/core-js/client/shim.min.js"></script>
  <script src="node_modules/zone.js/dist/zone.js"></script>
  <script src="node_modules/typescript/lib/typescript.js"></script>
  <script src="node_modules/systemjs/dist/system.src.js"></script>
  <script src="node_modules/rxjs/bundles/Rx.js"></script>
  <script src="systemjs.config.js"></script>
  <script>
    System.import('app').catch(function(err){ console.error(err); });
  </script>
</head>

<body>
<app>Loading...</app>
</body>
</html>
```

HTML-фрагмент в каждом приложении содержится либо внутри компонента (свойство `template`), либо в файле, на который ссылается этот компонент с помощью свойства `templateURL`. Второй вариант позволяет дизайнерам работать над интерфейсом вашего приложения, не изучая Angular.

Компонент Angular является центральным элементом новой архитектуры. На рис. 1.3 показана общая схема примера Angular-приложения, состоящего из четырех компонентов и двух сервисов; все они находятся внутри модуля внедрения зависимостей (Dependency Injection, DI). Этот модуль внедряет сервис `Http` в сервис `Service1`, а тот, в свою очередь, внедряется в компонент `GrandChild2`. Эта

схема отличается от приведенной рис. 1.1, где был показан принцип работы AngularJS.

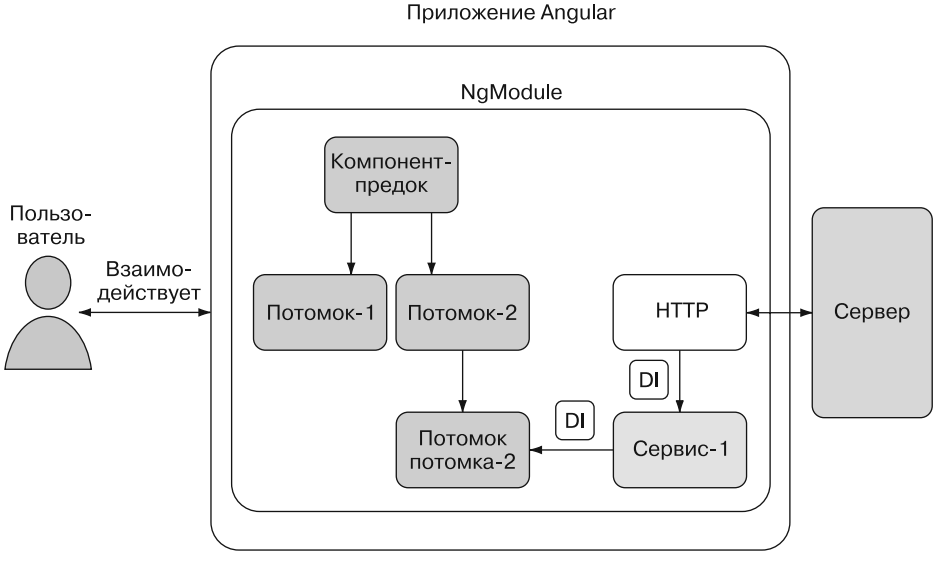

Рис. 1.3. Пример архитектуры Angular-приложения

Самый простой способ объявления компонента заключается в написании класса с помощью TypeScript (можно использовать ES5, ES6 или Dart). В приложении Б мы кратко расскажем о том, как писать компоненты Angular на TypeScript, а также приведем пример кода. Попробуйте понять этот код, прочитав минимальное количество пояснений.

Класс TypeScript содержит аннотацию метаданных `@NgModule` и представляет собой модуль. Класс TypeScript включает аннотацию метаданных `@Component` и представляет собой компонент. Аннотация `@Component` (также известная как декоратор) имеет свойство `template`, указывающее на фрагмент HTML, который должен быть отрисован в браузере.

Аннотации метаданных дают возможность изменять свойства компонентов во время разработки. Шаблон HTML может включать в себя выражения для привязки данных, окруженных двойными фигурными скобками. Ссылки на обработчики событий помещены в свойство `template` аннотации `@Component` и реализованы как методы класса. Еще одним примером аннотации метаданных выступает аннотация `@Injectable`, позволяющая отметить элемент, с которым должен работать модуль DI.

Аннотация `@Component` также содержит селектор, объявляющий имя пользовательского тега, который будет использован в документе HTML. Когда Angular видит элемент HTML, чье имя соответствует селектору, он знает, какой элемент

его реализует. В следующем фрагменте HTML показан родительский компонент `<auction-application>` с одним потомком, `<search-product>`:

```
<body>
  <auction-application>
    <search-product [productID]= "123"></search-product>
  </auction-application>
</body>
```

Предок отправляет данные потомкам путем привязки к входным свойствам потомка (обратите внимание на квадратные скобки в предыдущем фрагменте кода), а потомок общается с предками, отправляя события с помощью своих выходных свойств.

В конце главы вы найдете рис. 1.7, на котором показана главная страница (компонент-предок), чьи элементы-потомки окружены толстыми границами.

В следующем фрагменте кода показывается класс `SearchComponent`. Вы можете включить его в документ HTML как `<search-product>`, поскольку его объявление содержит свойство `selector` с таким же именем:

```
@Component({
  selector: 'search-product',
  template:
    `<form>
      <div>
        <input id="prodToFind" #prod>
        <button (click)="findProduct(prod.value)">Find Product</button>
        Product name: {{product.name}}
      </div>
    `</form>

})
  class SearchComponent {
    @Input() productID: number;

    product: Product; // Опустим код класса Product

    findProduct(prodName: string){
    // Здесь будет расположен реализация обработчика щелчков кнопкой мыши
  }
  // Здесь будет расположен другой код
}
```

Если вы знакомы с любым объектно-ориентированным языком, имеющим классы, то должны понять большую часть предыдущего фрагмента кода. Аннотированный класс `SearchComponent` объявляет переменную `product`, которая может представлять объект с несколькими свойствами, и одно из них (`name`) привязано к представлению (`{{product.name}}`). Локальная переменная шаблона `#prod` будет иметь ссылку на элемент `<input>`, поэтому вам не нужно запрашивать DOM для того, чтобы получить введенное значение.

Выражение `(click)` представляет собой событие щелчка кнопкой мыши. Функция обработчика событий получает значение аргумента из входного параметра `productID`, который будет заполнен родительским элементом путем привязки.

Мы кратко рассмотрели пример компонента. Более подробное описание содержимого этих элементов будет представлено в начале следующей главы. Не стоит беспокоиться, если вы раньше никогда не работали с классами, — данная тема раскрывается в приложениях А и Б. На рис. 1.4 показана внутренняя работа примера компонента, выполняющего поиск некоторых продуктов.

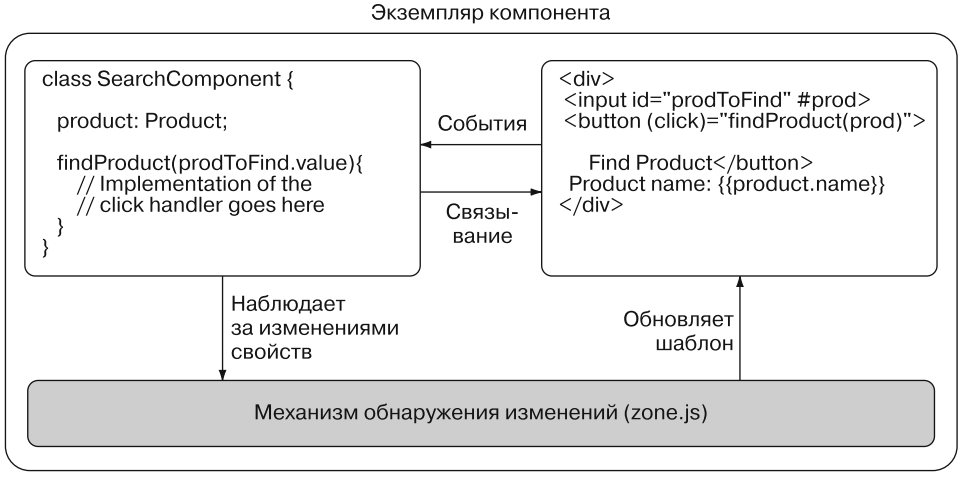

Рис. 1.4. Содержимое компонента

Компонент отрисовывает данные продукта, полученные из сервиса, представленного классом. В TypeScript класс `Product` будет выглядеть следующим образом:

```
class Product{
    id: number,
    name: string;
    description: string;
    bid: number;
    price: number;

    // Здесь будут расположены конструктор и другие методы
}
```

Обратите внимание: TypeScript позволяет объявить переменные класса с помощью типов. Чтобы указать компоненту пользовательского интерфейса `SearchComponent` на данные, можно объявить переменную класса, такую как `product`:

```
@Component({ /* код опущен для краткости */ })
class SearchComponent {
```

```
    product: Product;

    findProduct(productID){
        // Реализация обработчика щелчков кнопкой мыши
        // для раздела «Поиск компонентов» будет находиться здесь
    }
}
```

Если компонент поиска может вернуть несколько продуктов, то для их сохранения можно объявить массив:

```
products: Array<Product>;
```

Выражение *обобщений* объясняется в приложении Б. В предыдущем фрагменте кода конструкция `<Product>` указывает компилятору TypeScript, что в этом массиве можно хранить только объекты типа `Product`.

Фреймворк Angular не основан на шаблоне проектирования MVC, и ваше приложение не будет иметь отдельные контроллеры (C в аббревиатуре MVC). Компонент и внедренные сервисы (если таковые нужны) содержат весь необходимый код. В нашем примере класс `SearchProduct` будет включать код, который выполняет обязанности контроллера в дополнение к коду, необходимому для элемента пользовательского интерфейса представления HTML. В целях более четкого разделения TypeScript и HTML содержимое раздела `template` аннотации `@Component` можно хранить в отдельном файле, используя `templateUrl` вместо `template`, но это уже дело вкуса.

Теперь рассмотрим, почему дизайн Angular проще, чем в AngularJS. В последнем все директивы загружались в глобальную область видимости, а в Angular вы сами указываете необходимые директивы на уровне модулей, что предоставляет более качественную инкапсуляцию.

Вам не нужно работать с иерархией объектов областей видимости, как было в AngularJS. Фреймворк Angular основан на компонентах, и свойства создаются для этого объекта, который становится областью видимости компонента.

Одним из способов создания экземпляров объектов является применение оператора `new`. Если объект A зависит от объекта B, то в коде объекта A можно написать `let myB = new B();`. «Внедрение зависимостей» — шаблон проектирования, изменяющий способ создания объектов, который будет использован для кода. Вместо того чтобы явно создавать экземпляры объектов (например, с помощью оператора `new`), фреймворк будет создавать и внедрять их в код. Angular поставляется с модулем DI (эту тему мы рассмотрим в главе 4).

В AngularJS существует несколько способов регистрации зависимостей, что иногда может показаться непонятным. В Angular внедрять зависимости в элемент можно только с помощью конструктора. В следующем фрагменте кода TypeScript показывается, как бы вы внедряли компонент `ProductService` в компонент `SearchComponent`. Нужно лишь указать поставщик и объявить аргумент конструктора, используя тип, который соответствует типу поставщика:

```
@Component({
  selector: 'search-product',
  providers: [ProductService],
```

```
  template:`<div>...<div>`
})
class SearchComponent {
  products: Array<Product> = [];

  constructor(productService: ProductService) {
    this.products = productService.getProducts();
  }
}
```

Этот код не использует оператор `new` — Angular создаст объект типа `ProductService` и передаст ссылку на него компоненту `SearchComponent`.

Таким образом, Angular проще AngularJS по нескольким причинам:

- каждый блок вашего приложения является элементом, имеющим хорошо инкапсулированную функциональность представления, контроллера и автоматически сгенерированного детектора изменений;
- компоненты могут быть запрограммированы как аннотированные классы;
- вам не нужно работать с иерархией областей видимости;
- зависимые элементы внедряются в конструктор компонента;
- двусторонняя привязка по умолчанию выключена;
- механизм определения изменений был переписан и работает быстрее.

Концепции Angular легко поймут программисты, работающие с Java, C# и C++, — большая часть разработчиков промышленных программ. Нравится вам это или нет, но фреймворк становится более популярным, когда его начинают использовать для промышленных целей. AngularJS широко применяется в промышленности, и навыки работы с ним имеют спрос. Поскольку писать приложения на Angular проще, чем на AngularJS, данная тенденция будет сохраняться.

1.3.2. Улучшение производительности

Сайт Repaint Rate Challenge (http://mathieuancelin.github.io/js-repaint-perfs) сравнивает производительность отрисовки для разных фреймворков. Вы можете сравнить эффективность AngularJS и Angular 2 — последний демонстрирует значительное улучшение. Оно появилось в основном благодаря заново спроектированному внутреннему устройству фреймворка Angular. Отрисовка пользовательского интерфейса и API приложений была разделена на два слоя, что позволяет запускать не связанный с пользовательским интерфейсом код в отдельном рабочем потоке. В дополнение к возможности запускать код данных слоев параллельно браузеры выделяют разные ядра для этих потоков везде, где возможно. Вы можете найти детальное описание новой архитектуры отрисовки в документе, хранящемся в Google Docs, который называется *Angular 2 Rendering Architecture* — он доступен по ссылке https://docs.google.com/document/d/1M9FmT05Q6qpsjgvH1XvCm840yn2eWEg0PMskSQz7k4E/edit.

Создание отдельного слоя для отрисовки имеет еще одно важное преимущество: использование разных отрисовщиков для различных устройств. Каждый элемент содержит аннотацию `@Component`, которая включает шаблон HTML, определяющий его внешний вид. Если вы хотите создать компонент `<stock-price>` для отображения котировок акций в браузере, то его часть, что отвечает за пользовательский интерфейс, может выглядеть так:

```
@Component({
  selector: 'stock-price',
  template: '<div>The price of an IBM share is $165.50</div>'
})
class StockPriceComponent {
...
}
```

Графический движок Angular — отдельный модуль, позволяющий сторонним поставщикам заменить стандартный отрисовщик DOM на тот, что работает для платформ, не являющихся браузерами. Например, это дает возможность повторно использовать код TypeScript для устройств, имеющих сторонние отрисовщики пользовательского интерфейса, которые отрисовывают нативные компоненты. Часть их кода, написанная с помощью TypeScript, остается такой же, но содержимое свойства `template` декоратора `@Component` может содержать XML или другой язык для отрисовки нативных компонентов.

Один подобный отрисовщик для Angular уже реализован во фреймворке NativeScript, который служит мостом между JavaScript и нативными компонентами пользовательского интерфейса для iOS и Android. С помощью NativeScript можно повторно использовать код компонента, просто заменив HTML в шаблоне на XML. Еще один отрисовщик пользовательского интерфейса позволяет задействовать Angular вместе с React Native, что является альтернативным способом создания нативных (не гибридных) пользовательских интерфейсов для iOS и Android.

Новый, улучшенный механизм определения изменений также вносит свою лепту в улучшение производительности Angular. Этот фреймворк не использует двухстороннюю привязку, если только вы не укажете ее вручную. Односторонняя привязка упрощает определение изменений в приложении, которое может иметь множество взаимозависимых привязок. Вы можете пометить компонент, и он будет исключен из рабочего потока определения изменений, поэтому не будет проверяться при обнаружении изменения в другом элементе.

ПРИМЕЧАНИЕ

Несмотря на то что Angular является полной переработкой AngularJS, если вы применяете последний, то можете начать писать код в стиле Angular путем использования ng-forward (см. https://github.com/ngUpgraders/ng-forward). Другой подход (ng-upgrade) заключается в том, чтобы постепенно переходить на новую версию фреймворка, запуская Angular и AngularJS в одном приложении (см. https://angular.io/docs/ts/latest/guide/upgrade.html), но это увеличит его размер.