

Angular

для профессионалов

Адам Фримен

Адам Фримен

Angular для профессионалов

Серия «Для профессионалов»

Перевел с английского Е. Матвеев

Заведующая редакцией	Ю. Сергиенко
Ведущий редактор	Н. Римицан
Художественный редактор	С. Заматевская
Корректоры	С. Беляева, Н. Викторова
Верстка	Н. Лукьянова

ББК 32.988.02-018.1

УДК 004.43

Фримен А.

Ф88 Angular для профессионалов. — СПб.: Питер, 2018. — 800 с.: ил. — (Серия «Для профессионалов»).

ISBN 978-5-4461-0451-2

Выжмите из Angular — ведущего фреймворка для динамических приложений JavaScript — всё. Адам Фримен начинает с описания MVC и его преимуществ, затем показывает, как эффективно использовать Angular, охватывая все этапы, начиная с основ и до самых передовых возможностей, которые кроются в глубинах этого фреймворка.

Каждая тема изложена четко и лаконично, снабжена большим количеством подробностей, которые позволяют вам стать действительно эффективными. Наиболее важные фишки даны без излишних подробностей, но содержат всю необходимую информацию, чтобы вы смогли обойти все подводные камни.

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

ISBN 978-1484223062 англ.
ISBN 978-5-4461-0451-2

© 2017 by Adam Freeman
© Перевод на русский язык ООО Издательство «Питер», 2018
© Издание на русском языке, оформление ООО Издательство «Питер», 2018
© Серия «Для профессионалов», 2018

Права на издание получены по соглашению с Apress. 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: 09.2017. 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 07.09.17. Формат 70×100/16. Бумага офсетная. Усл. п. л. 64,500. Тираж 1200. 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

Оглавление

Об авторе	22
О научном редакторе	22
От издательства	22
Глава 1. Подготовка	23
Что вам необходимо знать?.....	23
Много ли в книге примеров?	24
Где взять примеры кода?	25
Как подготовить среду разработки?	26
Как связаться с автором.....	26
Итоги	26
Глава 2. Первое приложение	27
Подготовка среды разработки	27
Установка Node.js	27
Установка пакета angular-cli.....	28
Установка Git.....	29
Установка редактора	29
Установка браузера	30
Создание и подготовка проекта	30
Создание проекта	30
Создание файла пакета	31
Установка пакета NPM	32
Запуск сервера.....	33
Редактирование файла HTML.....	33
Добавление функциональности Angular в проект	36
Подготовка файла HTML.....	36
Создание модели данных.....	37
Создание шаблона	40
Создание компонента	40
Импортирование.....	41
Декораторы	42
Класс.....	42
А теперь все вместе.....	43
Расширение функциональности приложения.....	45
Добавление таблицы	45
Создание двусторонней привязки данных	48

Фильтрация задач	50
Добавление задач.....	51
Итоги	53
Глава 3. Angular в контексте.....	54
Сильные стороны Angular.....	55
Приложения с круговой передачей и одностраничные приложения	55
Паттерн MVC.....	58
Модели	60
Контроллеры/компоненты	62
Данные представления	63
Представления/шаблоны	63
REST-совместимые службы.....	63
Распространенные ошибки проектирования	66
Неверный выбор места для размещения логики	66
Использование формата хранилища данных	66
Начальные трудности.....	67
Итоги	67
Глава 4. Краткий курс HTML и CSS	68
Подготовка проекта	68
Понимание HTML.....	70
Пустые элементы.....	71
Атрибуты	71
Применение атрибутов без значений.....	72
Литералы в атрибутах	72
Контент элементов	72
Структура документа	73
Bootstrap	74
Применение базовых классов Bootstrap	75
Контекстные классы	76
Поля и отступы.....	77
Изменение размеров элементов.....	78
Использование Bootstrap для оформления таблиц	79
Использование Bootstrap для создания форм.....	80
Использование Bootstrap для создания сеток.....	82
Создание адаптивных сеток.....	83
Создание упрощенного сетчатого макета.....	86
Итоги	87
Глава 5. JavaScript и TypeScript: часть 1	88
Подготовка примера.....	90
Создание файлов HTML и JavaScript.....	90
Настройка компилятора TypeScript.....	91
Выполнение примера.....	92
Элемент script	92

Использование загрузчика модулей JavaScript	93
Основной процесс.....	94
Команды	95
Определение и использование функций.....	95
Определение функций с параметрами	97
Параметры по умолчанию и остаточные параметры	97
Определение функций, возвращающих результаты	99
Функции как аргументы других функций.....	99
Лямбда-выражения.....	100
Переменные и типы	100
Примитивные типы	102
Работа со строками.....	102
Работа с числами	104
Операторы JavaScript	104
Условные команды.....	105
Оператор равенства и оператор тождественности	105
Явное преобразование типов.....	106
Преобразование чисел в строки.....	107
Работа с массивами.....	108
Литералы массивов	109
Чтение и изменение содержимого массива	109
Перебор элементов массива	109
Встроенные методы массивов	110
Итоги	112
Глава 6. JavaScript и TypeScript: часть 2	113
Подготовка примера.....	113
Работа с объектами.....	114
Объектные литералы	115
Функции как методы	115
Определение классов	116
Определение свойств с get- и set-методами.....	118
Наследование	118
Работа с модулями JavaScript	119
Создание модулей	120
Импортирование из модулей JavaScript.....	121
Импортирование конкретных типов	121
Назначение псевдонимов при импортировании	122
Импортирование всех типов в модуле.....	123
Полезные возможности TypeScript.....	124
Аннотации типов	124
Применение аннотаций типов к свойствам и переменным	127
Кортежи	130
Индексируемые типы	130
Модификаторы доступа	131
Итоги	132

Глава 7. SportsStore: реальное приложение	133
Подготовка проекта	134
Создание структуры папок.....	134
Установка дополнительных пакетов NPM	134
Подготовка REST-совместимой веб-службы.....	136
Подготовка файла HTML.....	138
Запуск примера	138
Запуск REST-совместимой веб-службы	139
Подготовка проекта Angular	139
Обновление корневого компонента	140
Обновление корневого модуля	140
Анализ файла начальной загрузки.....	141
Начало работы над моделью данных.....	142
Создание классов модели	142
Создание фиктивного источника данных	143
Создание репозитория модели.....	144
Создание функционального модуля	146
Создание хранилища	146
Создание компонента магазина и шаблона	147
Создание функционального модуля хранилища	148
Обновление корневого компонента и корневого модуля.....	149
Добавление функциональности: подробная информация о товарах	150
Вывод подробной информации о товарах	150
Добавление выбора категорий.....	152
Страничный вывод списка товаров	154
Создание нестандартной директивы	158
Итоги	161
Глава 8. SportsStore: выбор товаров и оформление заказа.....	162
Подготовка приложения.....	162
Создание корзины	162
Создание модели корзины	162
Создание компонентов для сводной информации корзины	164
Интеграция корзины в приложение	166
Маршрутизация URL.....	169
Создание компонентов для содержимого корзины и оформления заказа	169
Создание и применение конфигурации маршрутизации	170
Навигация в приложении.....	172
Защитники маршрутов	175
Завершение вывода содержимого корзины	177
Обработка заказов	179
Расширение модели.....	180
Обновление репозитория и источника данных.....	181
Обновление функционального модуля.....	182
Получение информации о заказе.....	183

Использование REST-совместимой веб-службы	186
Применение источника данных	188
Итоги	189
Глава 9. SportsStore: администрирование	190
Подготовка приложения.....	190
Создание модуля	190
Настройка системы маршрутизации URL	193
Переход по URL администрирования.....	194
Реализация аутентификации	196
Система аутентификации.....	196
Расширение источника данных	197
Создание службы аутентификации	198
Включение аутентификации	200
Расширение источника данных и репозитория.....	202
Создание структуры подсистемы администрирования	206
Создание временных компонентов.....	206
Подготовка общего контента и функционального модуля	207
Реализация работы с товарами	210
Реализация управления заказами	214
Итоги	216
Глава 10. SportsStore: развертывание	217
Подготовка приложения к развертыванию	217
Контейнеризация приложения SportsStore	217
Установка Docker	218
Подготовка приложения	218
Создание контейнера.....	220
Запуск приложения.....	220
Итоги	221
Глава 11. Создание проекта Angular	222
Подготовка проекта Angular с использованием TypeScript.....	223
Создание структуры папок проекта.....	223
Создание документа HTML.....	223
Подготовка конфигурации проекта	224
Добавление пакетов	225
Настройка компилятора TypeScript.....	229
Настройка сервера HTTP для разработки	231
Запуск процессов-наблюдателей	234
Начало разработки приложений Angular с TypeScript.....	234
Создание модели данных.....	238
Создание репозитория модели.....	239
Создание шаблона и корневого компонента	241
Создание модуля Angular	242
Начальная загрузка приложения	243
Настройка загрузчика модулей JavaScript	244

Разрешение модулей RxJS	245
Разрешение нестандартных модулей приложения	246
Разрешение модулей Angular	246
Обновление документа HTML.....	248
Применение загрузчика модулей JavaScript	249
Стилевое оформление контента	249
Запуск приложения.....	250
Итоги	251
Глава 12. Привязки данных	252
Подготовка проекта	253
Односторонние привязки данных	254
Цель привязки	256
Привязки свойств	257
Выражения в привязках данных.....	257
Квадратные скобки	258
Управляющий элемент.....	259
Стандартные привязки свойств и атрибутов	260
Стандартные привязки свойств.....	260
Привязки со строковой интерполяцией.....	262
Привязки атрибутов.....	263
Назначение классов и стилей	264
Привязки классов	265
Назначение всех классов элемента с использованием стандартной привязки	265
Назначение отдельных классов с использованием специальной привязки класса	267
Назначение классов директивой ngClass.....	268
Привязки стилей.....	270
Назначение одного стилового свойства	270
Назначение стилей директивой ngStyle.....	272
Обновление данных в приложении.....	274
Итоги	276
Глава 13. Встроенные директивы	277
Подготовка проекта	278
Использование встроенных директив	280
Директива ngIf	280
Директива ngSwitch	283
Предотвращение проблем с литералами.....	284
Директива ngFor	286
Минимизация операций с элементами	292
Использование директивы ngTemplateOutlet	296
Предоставление контекстных данных	297
Ограничения односторонних привязок данных.....	299
Идемпотентные выражения	299
Контекст выражения.....	302
Итоги	304

Глава 14. События и формы	305
Подготовка проекта	306
Добавление модуля	306
Подготовка компонента и шаблона	308
Использование привязки события	309
Динамически определяемые свойства.....	311
Использование данных события.....	314
Использование ссылочных переменных шаблона	316
Двусторонние привязки данных	318
Директива ngModel	319
Работа с формами	321
Добавление формы в приложение	321
Проверка данных форм	324
Стилевое оформление элементов с использованием классов проверки данных	325
Вывод сообщений проверки данных на уровне полей	328
Использование компонента для вывода сообщений проверки данных	332
Проверка данных для всей формы.....	334
Вывод сводки проверки данных	337
Блокировка кнопки отправки данных.....	339
Использование форм на базе моделей	341
Включение поддержки форм на базе моделей	341
Определение классов модели для формы	342
Использование модели для проверки	346
Генерирование элементов по модели	350
Нестандартные правила проверки данных.....	351
Нестандартные классы проверки данных	351
Применение нестандартной проверки данных	352
Итоги	354
Глава 15. Создание директив атрибутов	355
Подготовка проекта	356
Создание простой директивы атрибута	359
Применение нестандартной директивы.....	360
Обращение к данным приложения в директиве.....	361
Чтение атрибутов управляющего элемента.....	361
Использование одного атрибута управляющего элемента.....	363
Создание входных свойств с привязкой к данным.....	364
Реакция на изменения входных свойств.....	367
Создание нестандартных событий	369
Привязка нестандартного события.....	372
Создание привязок управляющих элементов.....	373
Создание двусторонней привязки для управляющего элемента.....	374
Экспортирование директивы для использования в переменной шаблона	378
Итоги	381

Глава 16. Создание структурных директив	382
Подготовка проекта	383
Создание простой структурной директивы	384
Реализация класса структурной директивы.....	385
Регистрация структурной директивы	388
Определение исходного значения выражения.....	389
Компактный синтаксис структурных директив.....	390
Создание итеративных структурных директив	391
Предоставление дополнительных контекстных данных.....	394
Компактный структурный синтаксис	396
Изменения данных на уровне свойств.....	397
Изменения данных на уровне коллекции	399
Отслеживание представлений	406
Запрос контента управляющего элемента.....	410
Получение информации о нескольких контентных потомках.....	414
Получение уведомлений об изменениях в запросах.....	415
Итоги	417
Глава 17. Компоненты	418
Подготовка проекта	419
Применение компонентов для формирования структуры приложения	420
Создание новых компонентов	421
Новая структура приложения.....	424
Определение шаблонов	425
Определение внешних шаблонов.....	427
Использование привязок данных в шаблонах компонентов.....	428
Использование входных свойств для координации между компонентами.....	429
Использование директив в шаблоне дочернего компонента	431
Использование выходных свойств для координации между компонентами	432
Проецирование контента управляющего элемента	434
Завершение реструктуризации компонента	437
Использование стилей компонентов	437
Определение внешних стилей компонентов.....	439
Расширенные возможности стилей	440
Использование селекторов CSS теневой модели DOM.....	443
Выбор управляющего элемента	443
Выбор предков управляющего элемента.....	444
Продвижение стиля в шаблон дочернего компонента	446
Запрос информации о контенте шаблона.....	448
Итоги	450
Глава 18. Использование и создание каналов	451
Подготовка проекта	452
Установка полизаполнения интернационализации.....	454
Каналы.....	457

Создание нестандартного канала	458
Регистрация нестандартного канала	459
Применение нестандартного канала	460
Объединение каналов.....	461
Создание нечистых каналов.....	462
Использование встроенных каналов.....	466
Форматирование чисел.....	467
Форматирование денежных величин.....	470
Форматирование процентов.....	473
Форматирование дат	474
Изменение регистра символов в строке	478
Сериализация данных в формате JSON.....	479
Срезы массивов данных.....	480
Итоги	482
Глава 19. Службы	483
Подготовка проекта	484
Проблема распределения объектов.....	485
Суть проблемы	485
Распределение объектов как служб, использующих внедрение зависимостей.....	490
Регистрация службы	493
Анализ изменений при использовании внедрения зависимостей.....	494
Объявление зависимостей в других структурных блоках.....	496
Объявление зависимостей в директивах	499
Проблема изоляции тестов.....	503
Изоляция компонентов с использованием служб и внедрение зависимостей.....	503
Регистрация служб	505
Подготовка зависимого компонента.....	505
Переход на работу со службами	506
Обновление корневого компонента и шаблона	507
Обновление дочерних компонентов.....	508
Итоги	510
Глава 20. Провайдеры служб	511
Подготовка проекта	513
Использование провайдеров служб.....	514
Использование провайдера класса	517
Для чего нужен маркер?.....	518
Класс OAuthToken	519
Свойство useClass	521
Разрешение зависимостей с множественными объектами	523
Использование провайдера значения	525
Использование провайдера фабрики	527
Использование провайдера существующей службы	530
Использование локальных провайдеров.....	531
Ограничения модели с одним объектом службы	531

Создание локальных провайдеров в директиве.....	533
Создание локальных провайдеров в компонентах.....	535
Создание локального провайдера для всех потомков	538
Создание провайдера для потомков представлений	538
Управление разрешением зависимостей	540
Ограничения при поиске провайдера.....	540
Игнорирование самоопределяемых провайдеров	541
Итоги	542
Глава 21. Использование и создание модулей.....	543
Подготовка проекта	544
Корневой модуль.....	546
Свойство imports.....	548
Свойство declarations	548
Свойство providers	549
Свойство bootstrap.....	549
Создание функциональных модулей	551
Создание модуля модели	553
Создание определения модуля	554
Обновление других классов в приложении	555
Обновление корневого модуля	556
Создание вспомогательного функционального модуля.....	557
Обновление классов в новом модуле	559
Создание определения модуля	559
Свойство imports.....	560
Свойство providers	560
Свойство declarations	561
Свойство exports.....	561
Обновление других классов в приложении	561
Обновление корневого модуля	563
Создание функционального модуля с компонентами.....	563
Создание папки модуля и перемещение файлов	563
Обновление URL шаблонов.....	564
Создание определения модуля	566
Обновление корневого модуля	567
Итоги	567
Глава 22. Создание проекта.....	568
Начало работы над проектом	568
Добавление и настройка пакетов.....	569
Настройка TypeScript.....	570
Настройка сервера HTTP для разработки	571
Настройка загрузчика модулей JavaScript	571
Создание модуля модели	571
Создание типа данных Product.....	572

Создание источника данных и репозитория.....	572
Завершение модуля модели.....	574
Создание базового модуля.....	574
Создание службы общего состояния.....	574
Создание компонента таблицы.....	575
Создание шаблона компонента таблицы.....	576
Создание компонента формы.....	576
Создание шаблона для компонента формы.....	577
Создание базового модуля.....	578
Создание модуля сообщений.....	579
Создание модели сообщения и службы.....	579
Создание компонента и шаблона.....	580
Завершение модуля сообщений.....	581
Завершение проекта.....	581
Создание файла начальной загрузки Angular.....	582
Создание модуля Reactive Extensions.....	582
Создание документа HTML.....	583
Запуск приложения.....	584
Итоги.....	585
Глава 23. Reactive Extensions.....	586
Подготовка проекта.....	587
Суть проблемы.....	588
Решение проблемы при помощи Reactive Extensions.....	591
Объекты Observable.....	592
Объекты Observer.....	594
Объекты Subject.....	595
Использование канала async.....	596
Использование канала async с нестандартными каналами.....	598
Масштабирование функциональных модулей приложения.....	600
Расширенные возможности.....	602
Фильтрация событий.....	603
Преобразование событий.....	605
Использование разных объектов событий.....	606
Получение уникальных событий.....	607
Нестандартная проверка равенства.....	608
Передача и игнорирование событий.....	609
Итоги.....	611
Глава 24. Асинхронные запросы HTTP.....	612
Подготовка проекта.....	613
Настройка загрузчика модулей JavaScript.....	615
Настройка функционального модуля модели.....	615
Обновление компонента формы.....	616
Запуск проекта.....	617
REST-совместимые веб-службы.....	618

Замена статического источника данных	619
Создание новой службы источника данных.....	619
Настройка источника данных.....	622
Использование REST-совместимого источника данных	623
Сохранение и удаление данных	625
Консолидация запросов HTTP.....	627
Создание междоменных запросов	629
Использование запросов JSONP	631
Настройка заголовков запроса	633
Обработка ошибок	636
Генерирование сообщений для пользователя	637
Обработка ошибок.....	638
Итоги	640
Глава 25. Маршрутизация и навигация: часть 1	641
Подготовка проекта	642
Блокировка вывода событий изменения состояния	644
Знакомство с маршрутизацией	646
Создание конфигурации маршрутизации	646
Создание компонента маршрутизации	648
Обновление корневого модуля	649
Завершение конфигурации	650
Добавление навигационных ссылок	650
Эффект маршрутизации.....	653
Завершение реализации маршрутизации.....	656
Обработка изменений маршрутов в компонентах.....	656
Использование параметров маршрутов.....	659
Множественные параметры маршрутов	661
Необязательные параметры маршрутов.....	664
Навигация в программном коде	665
Получение событий навигации	667
Удаление привязок событий и вспомогательного кода	670
Итоги	672
Глава 26. Маршрутизация и навигация: часть 2	673
Подготовка проекта	673
Добавление компонентов в проект	676
Универсальные маршруты и перенаправления	679
Универсальные маршруты	679
Перенаправления в маршрутах.....	683
Навигация внутри компонента	684
Реакция на текущие изменения маршрутизации	685
Стилевое оформление ссылок для активных маршрутов	688
Исправление всех кнопок	691

Создание дочерних маршрутов.....	692
Создание элемента router-outlet для дочернего маршрута	693
Обращение к параметрам из дочерних маршрутов	695
Итоги	699
Глава 27. Маршрутизация и навигация: часть 3	700
Подготовка проекта	700
Защитники маршрутов	702
Отложенная навигация с использованием резольвера	703
Создание службы резольвера	704
Регистрация службы резольвера.....	706
Применение резольвера	706
Отображение временного контента	707
Использование резольвера для предотвращения проблем со вводом URL.....	708
Блокировка навигации с использованием защитников	710
Предотвращение активизации маршрута	711
Консолидация защитников дочерних маршрутов	715
Предотвращение деактивизации маршрутов.....	718
Динамическая загрузка функциональных модулей	723
Создание простого функционального модуля.....	724
Динамическая загрузка модулей.....	726
Создание маршрута для динамической загрузки модуля.....	726
Использование динамически загружаемого модуля	728
Защита динамических модулей	729
Применение защитника динамической загрузки.....	731
Именованные элементы router-outlet	732
Создание дополнительных элементов router-outlet	733
Навигация при использовании нескольких элементов router-outlet	735
Итоги	737
Глава 28. Анимация	738
Подготовка проекта	739
Добавление полизаполнения анимации	740
Отключение задержки HTTP	741
Упрощение шаблона таблицы и конфигурации маршрутизации	742
Основы работы с анимацией в Angular	744
Создание анимации	745
Определение групп стилей	745
Определение состояний элементов.....	746
Определение переходов состояний.....	747
Применение анимации.....	748
Тестирование эффекта анимации	751
Встроенные состояния анимации	753
Переходы элементов	754
Создание переходов для встроенных состояний	755

Анимация добавления и удаления элементов	755
Управление анимацией переходов	756
Определение начальной задержки	758
Использование дополнительных стилей во время перехода.....	759
Параллельные анимации	760
Группы стилей анимации.....	762
Определение общих стилей в группах для многократного использования.....	762
Использование преобразований элементов	763
Применение стилей фреймворка CSS.....	765
События триггеров анимации	767
Итоги	770

Глава 29. Модульное тестирование в Angular **771**

Подготовка проекта	773
Добавление пакетов тестирования	774
Настройка Karma	775
Создание простого модульного теста.....	777
Запуск инструментария.....	778
Работа с Jasmine	779
Тестирование компонентов Angular	781
Работа с классом TestBed	781
Настройка TestBed для работы с зависимостями	783
Тестирование привязок данных	785
Тестирование компонента с внешним шаблоном.....	787
Тестирование событий компонентов.....	789
Тестирование выходных свойств.....	791
Тестирование входных свойств.....	793
Тестирование с асинхронными операциями	796
Тестирование директив Angular.....	798
Итоги	800

1

Подготовка

Angular заимствует некоторые лучшие аспекты разработки на стороне сервера и использует их для расширения возможностей разметки HTML в браузере. Таким образом закладывается основа, которая упрощает и облегчает создание приложений с расширенной функциональностью. Приложения Angular строятся на базе паттерна проектирования MVC («модель — представление — контроллер»), который ориентирован на создание приложений, обладающих следующими характеристиками:

- *Простота расширения*: если вы понимаете основы, вам будет легко разобраться даже в самом сложном приложении Angular, а это означает, что вы сможете легко расширять приложения для поддержки новой полезной функциональности.
- *Удобство сопровождения*: приложения Angular просты в отладке, в них легко исправляются ошибки, а это означает простоту сопровождения кода в долгосрочной перспективе.
- *Удобство тестирования*: в Angular реализована хорошая поддержка модульного и сквозного тестирования. Следовательно, вы сможете находить и устранять дефекты до того, как ваши пользователи столкнутся с ними.
- *Стандартизация*: Angular работает на основе внутренней функциональности браузера, не создавая никаких препятствий для вашей работы. Это позволяет вам создавать веб-приложения, соответствующие стандартам, в которых задействована новейшая функциональность (например, различные API HTML5), популярные инструменты и фреймворки.

Angular — библиотека JavaScript с открытым кодом, финансированием разработки и сопровождения которой занимается компания Google. Она использовалась в ряде крупнейших и сложнейших веб-приложений. В этой книге вы узнаете все, что необходимо знать для использования Angular в ваших собственных проектах.

Что вам необходимо знать?

Чтобы книга принесла пользу, читатель должен быть знаком с основами веб-разработки, понимать, как работает HTML и CSS, а в идеале обладать практическими знаниями JavaScript. Для тех, кто забыл какие-то подробности, я кратко

напомню основные возможности HTML, CSS и JavaScript, используемые в книге, в главах 4, 5 и 6. Впрочем, вы не найдете здесь подробного справочника по элементам HTML и свойствам CSS. В книге, посвященной Angular, попросту не хватит места для полного описания HTML. Если вам требуется полный справочник по HTML и CSS, я рекомендую вам другую свою книгу, «The Definitive Guide to HTML5».

Много ли в книге примеров?

В книге *очень* много примеров. Angular лучше всего изучать на реальных примерах, поэтому я постарался включить в книгу как можно больше кода. Чтобы довести количество примеров до максимума, я воспользовался простой схемой, чтобы не приводить содержимое файлов снова и снова. Когда файл впервые встречается в главе, я привожу его полное содержимое, как в листинге 1.1. В заголовке листинга указывается имя файла и папка, в которой этот файл создается. При внесении изменений в код измененные команды выделяются жирным шрифтом.

Листинг 1.1. Пример документа

```
import { NgModule } from "@angular/core";
import { BrowserModule } from "@angular/platform-browser";
import { ProductComponent } from "./component";
import { FormsModule, ReactiveFormsModule } from "@angular/forms";
import { PaAttrDirective } from "./attr.directive";

@NgModule({
  imports: [BrowserModule, FormsModule, ReactiveFormsModule],
  declarations: [ProductComponent, PaAttrDirective],
  bootstrap: [ProductComponent]
})
export class AppModule { }
```

Этот листинг позаимствован из главы 15. Неважно, что он делает; просто запомните, что при первом использовании каждого файла в главе будет приводиться полный листинг, как в листинге 1.1. Во втором и всех последующих примерах я привожу только измененные элементы, то есть *неполный листинг*. Частичные листинги легко узнать, потому что они начинаются и заканчиваются многоточием (...), как показано в листинге 1.2.

Листинг 1.2. Неполный листинг

```
...
<table class="table table-sm table-bordered table-striped">
  <tr><th></th><th>Name</th><th>Category</th><th>Price</th></tr>
  <tr *ngFor="let item of getProducts(); let i = index"
    [pa-attr]="getProducts().length < 6 ? 'bg-success' : 'bg-warning'">
    <td>{{i + 1}}</td>
    <td>{{item.name}}</td>
    <td [pa-attr]="item.category == 'Soccer' ? 'bg-info' : null">
```

```
        {{item.category}}
    </td>
    <td [pa-attr]='bg-info'>{{item.price}}</td>
</tr>
</table>
...
```

Листинг 1.2 также взят из главы 15. В нем приводится только контент элемента `body`, и часть команд выделена жирным шрифтом. Так я привлекаю ваше внимание к части примера, которая демонстрирует описываемый прием или возможность. В неполных листингах приводятся только части, изменившиеся по сравнению с полным листингом, приведенным где-то ранее. В некоторых случаях изменения приходится вносить в разных частях одного файла; тогда я просто опускаю некоторые элементы или команды для краткости, как показано в листинге 1.3.

Листинг 1.3. Часть команд опущена для краткости

```
import { ApplicationRef, Component } from "@angular/core";
import { Model } from "../repository.model";
import { Product } from "../product.model";
import { ProductFormGroup } from "../form.model";

@Component({
  selector: "app",
  templateUrl: "app/template.html"
})
export class ProductComponent {
  model: Model = new Model();
  form: ProductFormGroup = new ProductFormGroup();

  // ...Другие свойства и методы опущены для краткости...

  showTable: boolean = true;
}
```

Эта схема позволила мне включить в книгу больше примеров, однако она усложняет поиск описания конкретных возможностей. По этой причине все главы, в которых описываются возможности Angular, начинаются со сводной таблицы с описанием различных возможностей, представленных в этой главе, и листингов, демонстрирующих их использование.

Где взять примеры кода?

Проекты примеров всех глав этой книги можно загрузить на сайте www.apress.com. Архив загружается бесплатно и включает все поддерживающие ресурсы, необходимые для воссоздания примеров без необходимости вводить весь код заново. Загружать код не обязательно, но возможность простого копирования кода в проекты позволит вам легко экспериментировать с примерами.

Как подготовить среду разработки?

В главе 2 вы познакомитесь с Angular на примере создания простого приложения. В ходе рассмотрения примера я объясню, как подготовить среду разработки для работы с Angular.

Как связаться с автором

Если у вас возникнут проблемы с запуском примеров или вы обнаружите ошибки в книге, свяжитесь со мной по адресу adam@adam-freeman.com; я постараюсь вам помочь.

Итоги

В этой главе я кратко обрисовал содержимое и структуру книги. Разработку приложений Angular лучше всего изучать на примерах, поэтому в следующей главе мы с ходу возьмемся за дело. Вы узнаете, как подготовить среду разработки и как использовать ее для создания вашего первого приложения Angular.

2

Первое приложение

Лучший способ начать работу с Angular — просто взяться за дело и создать веб-приложение. В этой главе я покажу, как настроить среду разработки, и проведу вас по основным этапам процесса создания простейшего приложения, начиная с построения статического макета функциональности и применения возможностей Angular для построения динамического веб-приложения (пусть и несложного). В главах 7–10 будет рассмотрено создание более сложных и реалистичных приложений Angular, но пока будет достаточно простого примера, который представит основные компоненты приложений Angular и заложит основу для других глав в этой части книги.

Не беспокойтесь, если что-то из материала этой главы покажется непонятным. Поначалу изучение Angular создает немало трудностей, поэтому эта глава всего лишь дает общее представление об основной последовательности разработки приложений Angular и о том, как стыкуются различные компоненты. На первых порах что-то может остаться непонятным, но когда вы перевернете последнюю страницу, вы будете понимать все, что происходит в этой главе, а также многое другое.

Подготовка среды разработки

Разработка Angular-приложений требует определенной подготовки. Далее я объясню, как выполнить необходимые действия для создания вашего первого проекта. Angular широко поддерживается популярными инструментами разработки, так что вы можете выбрать тот вариант, который лучше подходит лично вам.

Установка Node.js

Многие инструменты, используемые для разработки приложений Angular, зависят от Node.js (также известной как Node) — простой и эффективной исполнительной среды для серверных приложений, написанных на JavaScript, которая была создана в 2009 году. Node.js использует ядро JavaScript, задействованное в браузере Chrome, и предоставляет API для выполнения кода JavaScript за пределами окружения браузера.

Среда Node.js пользовалась большим успехом как сервер приложений, но для этой книги она интересна прежде всего тем, что заложила основу для нового поколения кроссплатформенной разработки и средств построения. Некоторые умные решения, принятые командой разработки Node.js, а также кроссплатформенная поддержка со стороны исполнительной среды JavaScript для Chrome открыли замечательные возможности, которые были моментально замечены увлеченными разработчиками инструментальных средств. Короче говоря, среда Node.js стала важнейшим инструментом разработки веб-приложений.

Очень важно, чтобы вы загрузили ту же версию Node.js, которая используется в этой книге. Хотя среда Node.js относительно стабильна, время от времени в API происходят критические изменения, которые могут нарушить работоспособность приведенных в книге примеров.

Я использую версию 6.10.1, самую актуальную версию с долгосрочной поддержкой на момент написания книги. Возможно, вы выберете для своих проектов более свежий выпуск, но для примеров книги следует придерживаться версии 6.10.1. Полный набор всех выпусков 6.10.1 с программами установки для Windows и Mac OS и двоичными пакетами для других платформ доступен по адресу <https://nodejs.org/dist/v6.10.1>.

В ходе установки Node.js не забудьте указать ключ для добавления пути к исполняемым файлам Node.js в переменную окружения. После завершения установки выполните следующую команду:

```
node -v
```

Если установка прошла так, как положено, команда выводит следующий номер версии:

```
v6.10.1
```

Установка Node.js включает менеджер пакетов проекта NPM (Node Package Manager). Выполните следующую команду, чтобы убедиться в том, что NPM работает:

```
npm -v
```

Если все работает так, как должно, выводится следующий номер версии:

```
3.10.10
```

Установка пакета angular-cli

Пакет `angular-cli` стал стандартным инструментом создания и управления пакетами Angular в ходе разработки. В исходной версии книги я показывал, как создавать пакеты Angular «с нуля»; это довольно долгий и ненадежный процесс, который упрощается благодаря `angular-cli`. Чтобы установить `angular-cli`, откройте новую командную строку и введите следующую команду:

```
npm install --global @angular/cli@1.0.0
```

В системе Linux или macOS вам, возможно, придется использовать команду `sudo`.

Установка Git

Система управления версиями Git нужна для управления некоторыми пакетами, необходимыми для разработки Angular. Если вы работаете в Windows или macOS, загрузите и запустите программу установки по адресу <https://git-scm.com/downloads>. (Возможно, в macOS вам придется изменить настройки безопасности для открытия программы установки, которую разработчики не снабдили цифровой подписью.)

Система Git заранее устанавливается в большинстве дистрибутивов Linux. Если вы захотите установить самую свежую версию, обратитесь к инструкциям для своего дистрибутива по адресу <https://git-scm.com/download/linux>. Например, для Ubuntu — моего дистрибутива Linux — используется следующая команда:

```
sudo apt-get install git
```

Завершив установку, откройте новую командную строку и выполните следующую команду, чтобы проверить правильность установки Git:

```
git --version
```

Команда выводит версию установленного пакета Git. На момент написания книги новейшей версией Git для Windows была версия 2.12.0, для macOS — 2.10.1, а для Linux — 2.7.4.

Установка редактора

В разработке приложений Angular может использоваться любой редактор для программистов. Выбор возможных вариантов огромен. В некоторых редакторах предусмотрена расширенная поддержка работы с Angular, включая выделение ключевых терминов и хорошую интеграцию с инструментарием. Если у вас еще нет любимого редактора для разработки веб-приложений, в табл. 2.1 представлены некоторые популярны варианты. Материал книги не зависит от какого-либо конкретного редактора; используйте тот редактор, в котором вам удобнее работать.

Один из важнейших критериев при выборе редактора — возможность фильтрации содержимого проекта, чтобы вы могли сосредоточиться на работе с некоторым подмножеством файлов. Проект Angular может содержать кучу файлов, многие из которых имеют похожие имена, поэтому возможность быстро найти и отредактировать нужный файл чрезвычайно важна. В редакторах эта функция может быть реализована разными способами: с выводом списка файлов, открытых для редактирования, или возможностью исключения файлов с заданным расширением.

ПРИМЕЧАНИЕ

Если вы работаете в Visual Studio (полноценной среде Visual Studio, не в Visual Studio Code), процесс работы с проектами Angular становится еще сложнее, особенно если вы захотите добавить Angular в проект ASP.NET Core MVC. Я планирую выпустить отдельное дополнение по использованию Angular в Visual Studio, вы сможете бесплатно загрузить его из репозитория GitHub этой книги.

Таблица 2.1. Популярные редакторы с поддержкой Angular

Название	Описание
Sublime Text	Sublime Text — коммерческий кроссплатформенный редактор с пакетами для поддержки большинства языков программирования, фреймворков и платформ. За подробностями обращайтесь по адресу www.sublimetext.com
Atom	Atom — бесплатный кроссплатформенный редактор с открытым кодом, уделяющий особое внимание возможностям настройки и расширения. За подробностями обращайтесь по адресу atom.io
Brackets	Brackets — бесплатный редактор с открытым кодом, разработанный компанией Adobe. За подробностями обращайтесь по адресу brackets.io
WebStorm	WebStorm — платный кроссплатформенный редактор с множеством интегрированных инструментов, чтобы вам не пришлось пользоваться командной строкой во время разработки. За подробностями обращайтесь по адресу www.jetbrains.com/webstorm
Visual Studio Code	Visual Studio Code — бесплатный кроссплатформенный редактор с открытым кодом, разработанный компанией Microsoft, с хорошими возможностями расширения. За подробностями обращайтесь по адресу code.visualstudio.com

Установка браузера

Остается принять последнее решение: выбрать браузер, который будет использоваться для проверки результатов работы в ходе разработки. Все браузеры последнего поколения обладают хорошей поддержкой для разработчика и хорошо сочетаются с Angular. Я использовал в книге Google Chrome; этот же браузер я могу порекомендовать и вам.

Создание и подготовка проекта

После того как вы установите Node.js, `angular-cli`, редактор и браузер, в вашем распоряжении окажется все необходимое для запуска процесса разработки.

Создание проекта

Выберите подходящую папку и выполните следующую команду в режиме командной строки, чтобы создать новый проект с именем `todo`:

```
ng new todo
```

Команда `ng` предоставляется пакетом `angular-cli`, а подкоманда `ng new` создает новый проект. Процесс установки создает папку с именем `todo`, которая содержит все файлы конфигурации, необходимые для разработки Angular, некоторые временные файлы, упрощающие начальную стадию разработки, и пакеты NPM, необходимые для разработки, запуска и развертывания приложений Angular. (Пакетов NPM довольно много; это означает, что создание проекта может занять много времени.)

Создание файла пакета

NPM использует файл с именем `package.json` для чтения списка программных пакетов, необходимых для проекта. Файл `package.json` создается `angular-cli` как часть инфраструктуры проекта, но он содержит только базовые пакеты, необходимые для разработки Angular. Для приложения этой главы понадобится пакет `Bootstrap`, не входящий в базовый набор пакетов. Отредактируйте файл `project.json` в папке `todo` и добавьте пакет `Bootstrap`, как показано в листинге 2.1.

ВНИМАНИЕ

К тому моменту, когда вы будете читать книгу, будут выпущены новые версии по крайней мере части пакетов из листинга 2.1. Чтобы ваши результаты соответствовали результатам примеров в этой и других главах, очень важно использовать конкретные версии, указанные в листинге. Если у вас возникнут проблемы с примерами этой или какой-либо из последующих глав, попробуйте использовать исходный код из архива, прилагаемого к книге; его можно загрузить на сайте издательства `apress.com`. Если же положение окажется совсем безвыходным, отправьте мне сообщение по адресу `adam@adam-freeman.com`, я постараюсь помочь вам.

Листинг 2.1. Содержимое файла `package.json` в папке `todo`

```
{
  "name": "todo",
  "version": "0.0.0",
  "license": "MIT",
  "scripts": {
    "ng": "ng",
    "start": "ng serve",
    "build": "ng build",
    "test": "ng test",
    "lint": "ng lint",
    "e2e": "ng e2e"
  },
  "private": true,
  "dependencies": {
    "@angular/common": "^4.0.0",
    "@angular/compiler": "^4.0.0",
    "@angular/core": "^4.0.0",
    "@angular/forms": "^4.0.0",
    "@angular/http": "^4.0.0",
    "@angular/platform-browser": "^4.0.0",
    "@angular/platform-browser-dynamic": "^4.0.0",
    "@angular/router": "^4.0.0",
    "core-js": "^2.4.1",
    "rxjs": "^5.1.0",
    "zone.js": "^0.8.4",
    "bootstrap": "4.0.0-alpha.4"
  },
  "devDependencies": {
    "@angular/cli": "1.0.0",
  }
}
```

```

"@angular/compiler-cli": "^4.0.0",
"@types/jasmine": "2.5.38",
"@types/node": "~6.0.60",
"codelyzer": "~2.0.0",
"jasmine-core": "~2.5.2",
"jasmine-spec-reporter": "~3.2.0",
"karma": "~1.4.1",
"karma-chrome-launcher": "~2.0.0",
"karma-cli": "~1.0.1",
"karma-jasmine": "~1.1.0",
"karma-jasmine-html-reporter": "^0.2.2",
"karma-coverage-istanbul-reporter": "^0.2.0",
"protractor": "~5.1.0",
"ts-node": "~2.0.0",
"tslint": "~4.5.0",
"typescript": "~2.2.0"
}
}

```

В файле `package.json` перечислены пакеты, необходимые для начала разработки приложений Angular, и некоторые команды для их использования. Все параметры конфигурации будут описаны в главе 11, а пока достаточно понимать, для чего нужна каждая секция файла `package.json` (табл. 2.2).

Таблица 2.2. Секции файла `package.json`

Имя	Описание
<code>scripts</code>	Список сценариев, запускаемых в режиме командной строки. Секция <code>scripts</code> в листинге запускает команды, используемые для компиляции исходного кода, и сервер HTTP для разработки
<code>dependencies</code>	Список пакетов NPM, от которых зависит работа веб-приложения. Для каждого пакета указан номер версии. В секции <code>dependencies</code> в листинге перечислены базовые пакеты Angular; библиотеки, от которых зависит Angular; и CSS-библиотека Bootstrap, которая используется для стилизового оформления контента HTML в книге
<code>devDependencies</code>	Список пакетов NPM, которые используются в разработке, но не нужны для работы приложения после его развертывания. В этой секции перечислены пакеты, компилирующие файлы TypeScript, предоставляющие функциональность сервера HTTP в процессе разработки и обеспечивающие тестирование

Установка пакета NPM

Чтобы файл `package.json` был обработан NPM для загрузки и установки указанного в нем пакета Bootstrap, выполните следующую команду из папки `todo`:

```
npm install
```

NPM выдает несколько предупреждений относительно обрабатываемых пакетов, но сообщений об ошибках быть не должно.

Запуск сервера

Инструментарий и базовая структура находятся на своих местах; пришло время убедиться в том, что все работает нормально. Выполните следующие команды из папки `todo`:

```
npm serve --port 3000 --open
```

Команда запускает сервер HTTP для разработки, который был установлен `angular-cli` и настроен для работы с инструментарием разработчика Angular. Запуск занимает немного времени для подготовки проекта, а вывод выглядит примерно так:

```
** NG Live Development Server is running on http://localhost:3000 **
Hash: b8843310528d229c2540
Time: 11251ms
chunk {0} polyfills.bundle.js, polyfills.bundle.js.map (polyfills) 158 kB {4}
[initial] [rendered]
chunk {1} main.bundle.js, main.bundle.js.map (main) 3.69 kB {3} [initial]
[rendered]
chunk {2} styles.bundle.js, styles.bundle.js.map (styles) 9.77 kB {4} [initial]
[rendered]
chunk {3} vendor.bundle.js, vendor.bundle.js.map (vendor) 2.37 MB [initial]
[rendered]
chunk {4} inline.bundle.js, inline.bundle.js.map (inline) 0 bytes [entry]
[rendered]
webpack: Compiled successfully.
```

Не беспокойтесь, если в вашем случае вывод будет выглядеть немного иначе — главное, чтобы после завершения подготовки появилось сообщение `Compiled successfully`. Через несколько секунд откроется окно браузера (рис. 2.1); это означает, что запуск проекта прошел успешно и в нем используется временный контент, сгенерированный `angular-cli`.

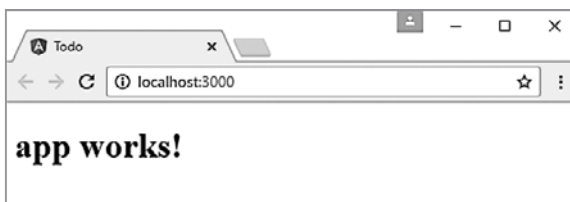


Рис. 2.1. Временный контент HTML

Редактирование файла HTML

Хотя пакет `angular-cli` добавил временный контент, мы сейчас удалим все лишнее и начнем с заготовки HTML, содержащей статический контент. Позже эта заготовка будет расширена для Angular. Отредактируйте файл `index.html` в папке `todo/src` и включите в него контент из листинга 2.2.

Листинг 2.2. Содержимое файла index.html в папке todo/src

```
<!DOCTYPE html>
<html>
<head>
  <title>ToDo</title>
  <meta charset="utf-8" />
  <link href="node_modules/bootstrap/dist/css/bootstrap.min.css"
        rel="stylesheet" />
</head>
<body class="m-a-1">

  <h3 class="bg-primary p-a-1">Adam's To Do List</h3>

  <div class="m-t-1 m-b-1">
    <input class="form-control" />
    <button class="btn btn-primary m-t-1">Add</button>
  </div>

  <table class="table table-striped table-bordered">
    <thead>
      <tr>
        <th>Description</th>
        <th>Done</th>
      </tr>
    </thead>
    <tbody>
      <tr><td>Buy Flowers</td><td>No</td></tr>
      <tr><td>Get Shoes</td><td>No</td></tr>
      <tr><td>Collect Tickets</td><td>Yes</td></tr>
      <tr><td>Call Joe</td><td>No</td></tr>
    </tbody>
  </table>
</body>
</html>
```

Сервер HTTP для разработки `angular-cli` добавляет фрагмент JavaScript в контент HTML, передаваемый браузеру. JavaScript открывает обратное подключение к серверу и ждет сигнала на перезагрузку страницы; сигнал отправляется при обнаружении сервером изменений в любых файлах из каталога `todo`. Как только вы сохраните файл `index.html`, сервер обнаружит изменения и отправит сигнал. Браузер перезагружается и выводит новый контент (рис. 2.2).

ПРИМЕЧАНИЕ

При внесении изменений в группу файлов может случиться так, что браузер не сможет загрузить и выполнить приложение, особенно в последующих главах, когда приложения станут более сложными. В большинстве случаев сервер HTTP для разработки инициирует перезагрузку в браузере, и все будет нормально, но если у него возникнут затруднения, просто нажмите кнопку обновления в браузере или перейдите по адресу `http://localhost:3000`, чтобы восстановить нормальную работу.

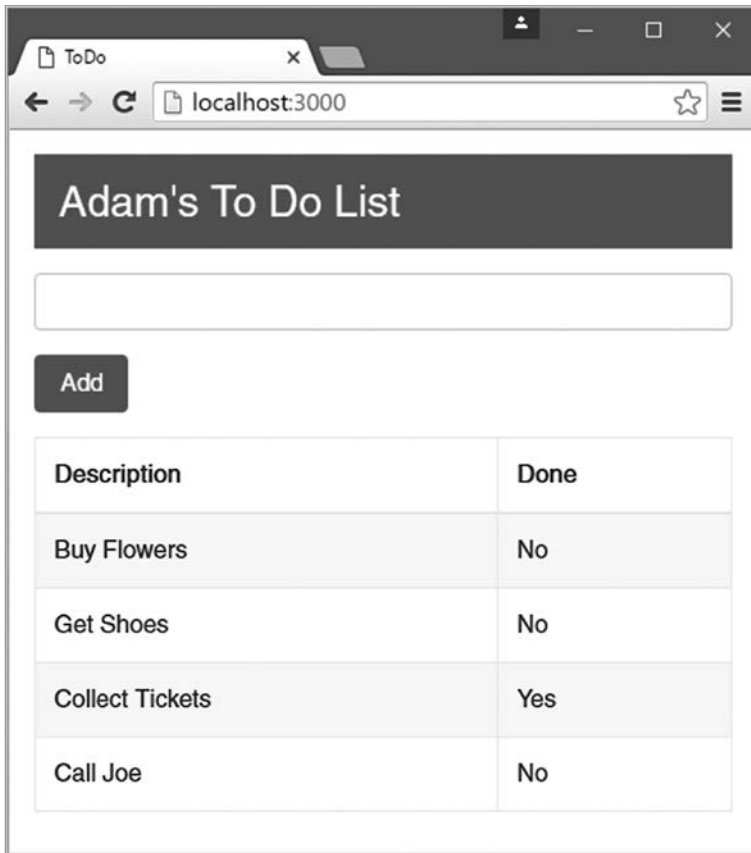


Рис. 2.2. Изменение содержимого файла HTML

Элементы HTML в файле `index.html` показывают, как будет выглядеть простое приложение Angular, которое мы создадим в этой главе. Его ключевые элементы — заголовок с именем пользователя, поле ввода, кнопка для добавления новой задачи в список и таблица со всеми задачами и признаками их завершения.

В этой книге я использую для оформления контента HTML превосходный CSS-фреймворк Bootstrap. Работа Bootstrap базируется на назначении элементам классов:

```
...  
<h3 class="bg-primary p-a-1">Adam's To Do List</h3>  
...
```

Элементу `h3` назначены два класса. Класс `bg-primary` назначает цветом фона элемента первичный цвет текущей темы Bootstrap. Я использую тему по умолчанию, в которой первичным цветом является темно-синий; также доступны другие цвета из темы, включая `bg-secondary`, `bg-info` и `bg-danger`. Класс `p-a-1` добавляет ко

всем сторонам элемента фиксированные отступы, чтобы текст был окружен небольшим свободным пространством.

В следующем разделе мы удалим из файла разметку HTML, разобьем ее на несколько меньших фрагментов и используем для создания простого приложения Angular.

ИСПОЛЬЗОВАНИЕ ПРЕДВАРИТЕЛЬНОЙ ВЕРСИИ BOOTSTRAP

В этой книге используется предварительная версия фреймворка Bootstrap. В то время, когда я пишу эти строки, команда Bootstrap занимается разработкой Bootstrap версии 4 и уже опубликовала несколько ранних выпусков. Эти выпуски считаются «альфа-версиями», но они содержат качественный код и работают достаточно стабильно для использования в примерах книги.

Когда мне пришлось выбирать между версией Bootstrap 3, которая скоро станет устаревшей, и предварительной версией Bootstrap 4, я решил использовать новую версию, несмотря на то что некоторые имена классов, используемых для оформления элементов HTML, могут измениться в окончательной версии. Это означает, что для получения предполагаемых результатов в примерах вы должны использовать ту же версию Bootstrap — как и в остальных пакетах, перечисленных в файле package.json из листинга 2.1.

Добавление функциональности Angular в проект

Статическая разметка HTML в файле index.html заменяет простейшее приложение. Пользователь может просматривать список задач, пометить выполненные и создавать новые задачи. В дальнейших разделах мы добавим в проект Angular и воспользуемся некоторыми базовыми возможностями для того, чтобы вдохнуть жизнь в приложение. Для простоты предполагается, что у приложения всего один пользователь и нам не нужно беспокоиться о сохранении данных в приложении, а это означает, что изменения в списке будут потеряны при закрытии или перезагрузке окна браузера. (В последующих примерах, включая приложение SportsStore, создаваемое в главах 7–10, будет продемонстрирован механизм долгосрочного хранения данных.)

Подготовка файла HTML

Первым шагом на пути включения Angular в приложение станет подготовка файла index.html (листинг 2.3).

Листинг 2.3. Подготовка к использованию Angular в файле index.html

```
<!DOCTYPE html>
<html>
<head>
  <title>ToDo</title>
```