

*Посвящается моему отцу — он всегда верил в меня,
даже если то, что я делал, не имело для него
(а порой и меня самого) никакого смысла!*

Содержание

Об авторе	13
Благодарности	14
Глава 1. Знакомство с React.....	15
Многостраничный дизайн старой школы.....	16
Одностраничные приложения новой школы	17
Знакомьтесь с React.....	21
Автоматическое управление состоянием пользовательского интерфейса	22
Быстрое манипулирование DOM.....	22
API для создания компонуемых пользовательских интерфейсов ...	23
Визуальные элементы, целиком определенные в коде JavaScript.....	25
Только V в архитектуре MVC.....	28
Заключение	28
Глава 2. Создание первого React-приложения	29
Работа с JSX	30
Использование React	32
Отображение имени	34
Пока ничего нового	37
Смена места назначения.....	37
Добавление стилей	39
Заключение	41
Глава 3. Компоненты React.....	43
Краткий обзор функций	44
Изменения во взаимодействии с пользовательским интерфейсом.....	46
Встречайте компонент React.....	50
Создание компонента HelloWorld.....	51
Указание свойств	56

Часть первая: Обновление определения компонента	57
Вторая часть: Изменение кода вызова компонента	57
Работа с дочерними элементами	58
Заключение	61
Глава 4. Стили в библиотеке React	62
Отображение нескольких гласных	63
Стилизация контента React с помощью CSS	66
Разберитесь в сгенерированном HTML-коде	66
Примените стиль	67
Стилизация контента методом библиотеки React	69
Создание объекта стиля	70
Стилизация контента	71
Настройка фонового цвета	73
Заключение	75
Глава 5. Создание сложных компонентов	76
От визуализации до компонентов	76
Определение основных визуальных элементов	78
Определение компонентов	82
Создание компонентов	85
Компонент <code>Card</code>	87
Компонент <code>Square</code>	89
Компонент <code>Label</code>	91
Передача свойств	93
Почему так важна компоновка компонентов	96
Заключение	98
Глава 6. Передача свойств	99
Обзор проблемы	99
Глубокий анализ проблемы	104
Оператор расширения	110
Лучший способ передачи свойств	111
Заключение	114
Глава 7. Встречайте JSX.. Вновь!	116
Что происходит с JSX?	116
Причуды JSX, которые надо запомнить	119
Обработка выражений	119

Возвращение нескольких элементов	120
Нельзя использовать встроенные стили CSS	123
Комментарии	124
Регистр, HTML-элементы и компоненты	125
JSX-код может быть где угодно.....	126
Заключение	126
Глава 8. Работа с состояниями React	128
Использование состояний	128
Начало работы	129
Настройка счетчика	132
Установка начального значения состояния.....	133
Запуск таймера и установка состояния.....	135
Рендеринг смены состояния	138
Дополнительно: полный код	139
Заключение	142
Глава 9. Переход от данных к пользовательскому интерфейсу	143
Пример	143
JSX-код может быть где угодно, часть II.....	147
Работа с массивами	148
Заключение	152
Глава 10. События в React	154
Определение и реагирование на события.....	154
Начало работы	156
Создание кнопки действия	159
Свойства события	161
Синтетические события	162
Работа со свойствами события	164
Действия с событиями.....	166
Вы не можете напрямую прослушивать события компонентов	166
Прослушивание стандартных событий DOM	169
Значение <code>this</code> внутри обработчика событий.....	171
React... Почему? Почему?	173
Совместимость с браузерами.....	173
Повышение производительности.....	173
Заключение	175

Глава 11. Жизненный цикл компонента	176
Знакомство с методами жизненного цикла	177
Методы жизненного цикла в действии.....	177
Этап начального рендеринга	182
Получение свойств по умолчанию.....	183
Получение состояния по умолчанию.....	184
componentWillMount	184
render	184
componentDidMount.....	185
Этап обновления.....	185
Изменение состояний	185
shouldComponentUpdate	186
componentWillUpdate	187
render	187
componentDidUpdate.....	188
Изменения свойств	188
Этап размонтирования	189
Заключение	190
Глава 12. Доступ к элементам DOM в React	191
Код приложения «Палитра»	194
Знакомство со ссылками	197
Использование порталов.....	202
Заключение	207
Глава 13. Настройка среды разработки React	208
Проект Create React	211
Анализ произошедшего	214
Создание демонстрационного приложения	219
Сборка проекта	224
Заключение	225
Глава 14. Работа с внешними данными в React	226
Основы веб-запросов.....	229
Время работать с React.....	231
Начало работы	232
Получение IP-адреса.....	234
Визуальные эффекты уровнем выше.....	238
Заключение	243

Глава 15. Создание планировщика в React.....	244
Поехали!	246
Создание интерфейса.....	248
Построение остальной части приложения.....	250
Добавление элементов.....	251
Отображение элементов.....	256
Форматирование контента приложения	259
Удаление элементов	262
Анимация! Анимация! Анимация!	266
Заключение	268
Глава 16. Создание плавающего меню в React.....	269
Как работает плавающее меню	269
Настройка плавающего меню	273
Начало работы.....	275
Отображение и сокрытие меню	279
Создание кнопки	281
Создание меню	283
Заключение	287
Глава 17. Избегайте ненужных операций рендеринга	288
Вкратце о методе <code>render</code>	288
Оптимизация количества вызовов метода <code>render</code>	291
Разбираем пример.....	291
Анализ вызовов метода <code>render</code>	293
Переопределение обновления компонента	297
Использование компонента <code>PureComponent</code>	300
Заключение	302
Глава 18. Создание одностраничного приложения React	
с помощью роутера.....	304
Пример	306
Начало работы.....	307
Создание одностраничного приложения	309
Отображение начального фрейма	309
Создание страниц с контентом.....	311
Использование библиотеки <code>React Router</code>	313
Последние штрихи.....	318

Исправление путей маршрутизации	318
Добавление правил CSS.....	318
Выделение активной ссылки	321
Заключение	322
Глава 19. Введение в Redux	324
Что такое Redux.....	325
Создание простого приложения с помощью Redux.....	330
Настало время Redux.....	331
Свет! Камера! Мотор!.....	332
Управление редуктором	333
Работа с хранилищем.....	336
Заключение	338
Глава 20. Совместное использование Redux с React	340
Управление состояниями React с помощью Redux	348
Как пересекаются React и Redux	349
Начало работы	352
Создание приложения	353
Заключение	362
Предметный указатель	363

Об авторе

Кирупа Чиннатамби потратил большую часть своей жизни на то, чтобы помочь другим людям полюбить веб-разработку так, как ее любит он.

В 1999 году, еще до появления слова «блог», он начал публиковать обучающие материалы на сайте **kirupa.com**. С тех пор он написал сотни статей, стал автором несколько книг (не таких хороших, как эта, разумеется!), а также создал множество видеороликов, которые вы можете найти на YouTube. Когда он не пишет и не говорит о веб-разработке, он тратит свои часы бодрствования на развитие Всемирной паутины, работая в качестве менеджера проектов в компании Microsoft. В часы небодрствования он, вероятно, спит или пишет о самом себе в третьем лице.

Вы можете найти его в Twitter (twitter.com/kirupa), Facebook (facebook.com/kirupa) или связаться с ним по электронной почте (kirupa@kirupa.com). Не стесняйтесь обращаться к нему в любое время.

Благодарности

Во-первых, эта книга не увидела бы свет без одобрения и поддержки моей замечательной жены, **Мины**. Если бы она не отсрочила достижение собственных целей, позволив мне потратить шесть месяцев на обдумывание, написание и доработку того, что вы здесь видите, то создание этой книги так и осталось бы мечтой.

Также я хотел бы поблагодарить **своих родителей**. Они всегда поощряли мои бесцельные поиски и позволяли делать то, что мне нравится, например, учить незнакомцев через Интернет делать классные вещи с помощью программирования, чем я занимался в конце 1990-х годов. Если бы не они, я не стал бы и вполнину таким суровым домоседом/ученым/воином, каким являюсь сегодня.

Написать слова, которые вы читаете, довольно легко, однако чрезвычайно сложно сделать так, чтобы эта книга попала к вам в руки. Чем больше я узнаю о деталях этого сложного механизма, тем больше восхищаюсь людьми, которые неустанно работают, следя за тем, чтобы он функционировал, как надо. Я благодарю **каждого сотрудника издательства Pearson**, работавшего над этим проектом! Тем не менее есть несколько человек, которых я хотел бы поблагодарить особо. Во-первых, я благодарю **Марка Тэйбера** за предоставленные возможности для совместной работы, **Криса Зана** за терпеливое решение моих многочисленных проблем, **Кристу Хэнсинг** за преобразование моей версии английского языка в нечто понятное, а также **Лоретту Йетс** за то, что когда-то она помогла наладить связи, позволившие воплотить этот проект. Техническая составляющая книги была тщательно проверена моими давними друзьями и сотрудниками **Кайлом Мюрреем (a. k. a. Krilnon)** и **Тревором МакКоли (a. k. a. senocular)**. Я бесконечно благодарен им за их подробные (и часто очень забавные!) комментарии.

Глава 1

Знакомство с React

Если на мгновение забыть об общем улучшении *внешнего вида* и *опыта использования* веб-приложения, то в этой сфере можно заметить фундаментальное изменение. Теперь веб-приложения проектируются и создаются иначе. Чтобы убедиться в этом, рассмотрим приложение, изображенное на рис. 1.1.



Рис. 1.1. Приложение

Данное приложение представляет собой каталог. Как и в случае с любой подобной программой, у нас есть обычный набор страниц, связанных с домашней страницей, страница результатов поиска, страница сведений и т. д. В следующих разделах мы рассмотрим два

подхода к созданию этого приложения. Да, по таинственному совпадению, мы в то же время познакомимся и с библиотекой React.

Вперед!

Многостраничный дизайн старой школы

Если бы вы создавали это приложение несколько лет назад, вероятно, применили бы подход, предполагающий использование нескольких отдельных страниц. В этом случае поток выглядел бы как на рис. 1.2.

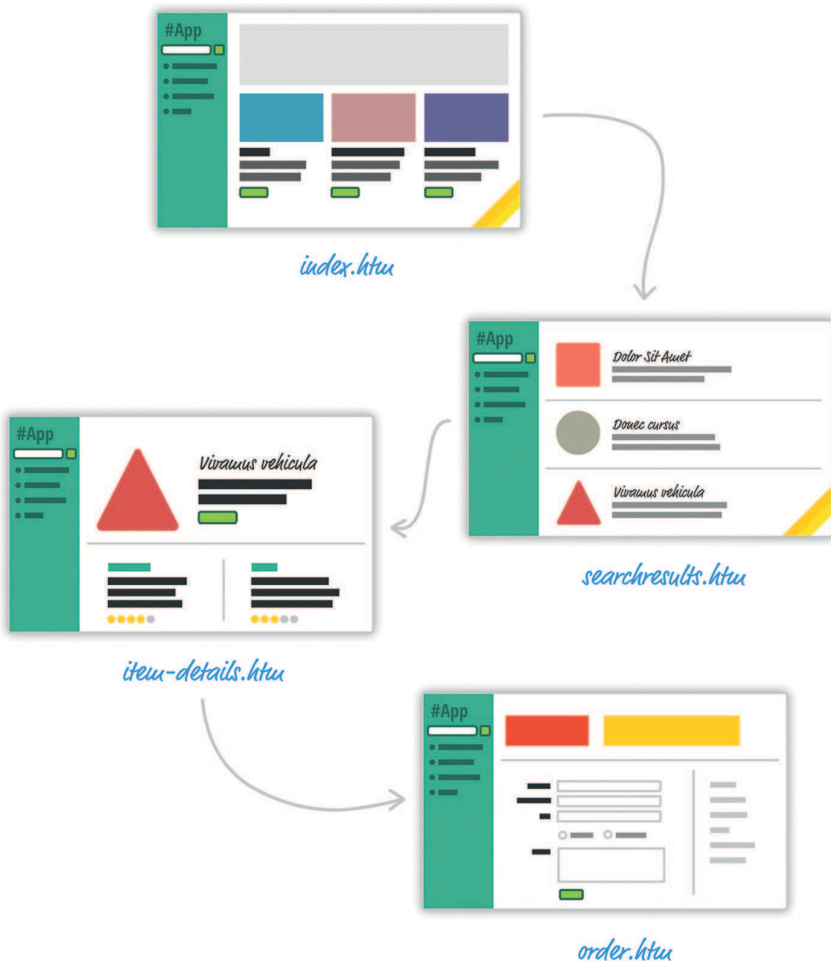


Рис. 1.2. Многостраничный дизайн

В ответ почти на каждое действие, изменяющее содержимое окна браузера, веб-приложение переводит вас на *совершенно другую страницу*. Помимо того, что уничтожение и воссоздание страниц ухудшает пользовательский опыт, это оказывает большое влияние на способ поддержания состояния приложения. За исключением сохранения пользовательских данных с помощью cookie-файлов и некоторых серверных механизмов вам не о чем беспокоиться. Жизнь хороша.

Одностраничные приложения новой школы

Сегодня модель веб-приложения, которая требует перехода между отдельными страницами, устаревает (см. рис. 1.3).

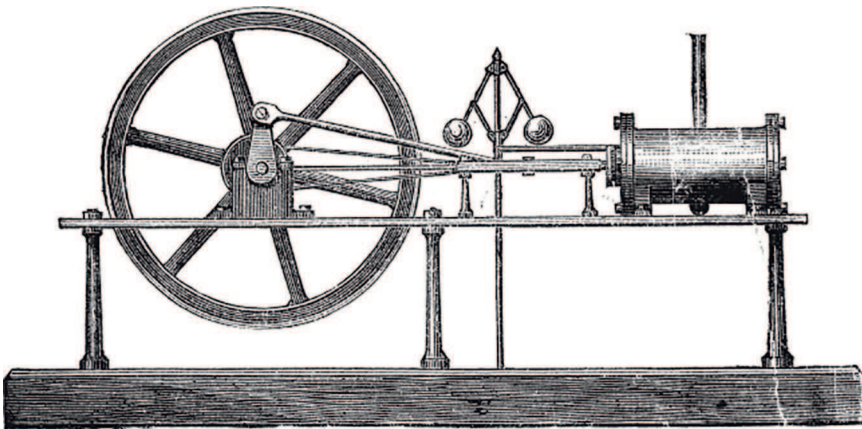


Рис. 1.3. Модель, использующая отдельные страницы, устарела, как этот паровой двигатель

Вместо этого современные приложения обычно основываются на так называемой **модели SPA** (Single-Page Applications, одностраничные приложения). Она позволяет создать приложение, в котором вы никогда не переходите на другие страницы и даже не перезагружаете их. При ее использовании различные представления приложения загружаются в одной и той же странице.

В случае с нашим приложением это выглядит как на рис. 1.4.



Рис. 1.4. Одностраничное приложение

В процессе взаимодействия пользователей с приложением мы заменяем содержимое области, обведенной красной пунктирной линией, данными и HTML-кодом, который соответствует тому, что пытается сделать пользователь. Это дает гораздо более плавный пользовательский опыт. Вы можете применять множество визуальных приемов для плавного перехода между элементами контента, как в тех классных приложениях, с которыми вы работаете на своем мобильном устройстве или настольном компьютере. Такие вещи невозможны, когда вы переходите со страницы на страницу.

Все это кажется непонятным, если вы никогда не слышали об одностраничных приложениях, но, вероятно, вы уже с ними сталкивались. Если вы использовали такие популярные веб-приложения, как Gmail, Facebook, Instagram или Twitter, то знайте: вы использовали

одностраничное приложение. Во всех этих приложениях контент отображается динамически, не требуя обновления или перехода на другую страницу.

Одностраничные приложения могут показаться сложными. Это совсем не так. Благодаря большому количеству улучшений как в языке JavaScript, так и в различных сторонних фреймворках и библиотеках, создание одностраничных приложений никогда не было таким простым. Но возможности для улучшения не закончились.

При создании одностраничных приложений вы однажды столкнетесь с тремя основными проблемами:

- 1. В одностраничном приложении большая часть времени уходит на синхронизацию данных с пользовательским интерфейсом.** Например, что вы сделаете, когда пользователь загрузит новое содержимое? Очистите поле поиска? Оставьте активную вкладку по-прежнему видимой на навигационном элементе? Какие элементы вы сохраните на странице, а какие уничтожите?

Эти вопросы актуальны именно для одностраничных приложений. При навигации между страницами в старой модели мы предполагали, что все элементы пользовательского интерфейса будут уничтожены и созданы заново. Это никогда не было проблемой.

- 2. Манипулирование DOM* происходит очень медленно.** На запрашивание элементов вручную, добавление дочерних элементов (см. рис. 1.5), удаление поддеревьев и выполнение других операций с DOM в браузере тратится больше всего времени. К сожалению, в одностраничном приложении вам придется часто этим заниматься. Манипулирование DOM — это основной способ, позволяющий вам реагировать на действия пользователя и отображать новое содержимое.

* Document object model — объектная модель документа (*прим. перев.*).