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

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 вполовину 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.

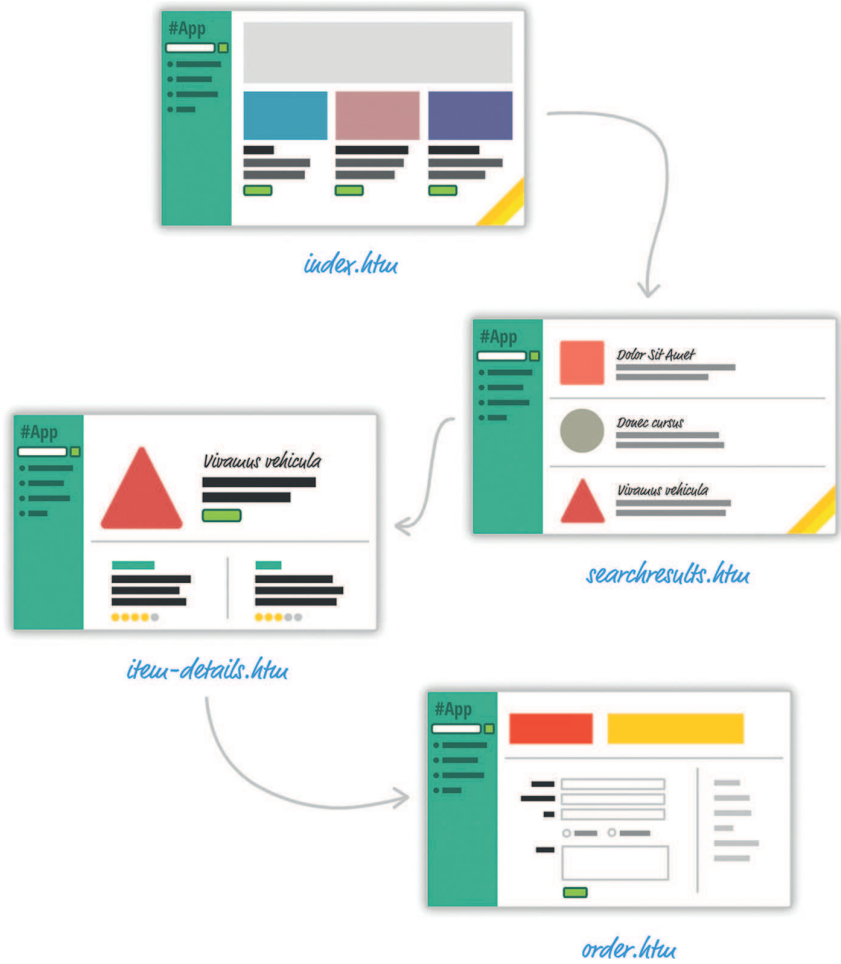

Рис. 1.2. Многостраничный дизайн

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).

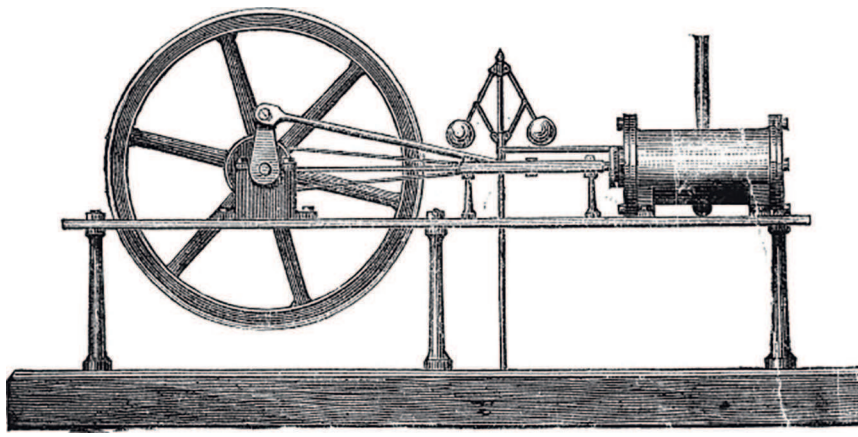

Рис. 1.3. Модель, использующая отдельные страницы, устарела, как этот паровой двигатель

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 бóльшая часть времени уходит на синхронизацию данных с пользовательским интерфейсом.** Например, что вы сделаете, когда пользователь загрузит новое содержимое? Очистите поле поиска? Оставите 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.*).