Содержание

Пролог

Всем нам знакома избитая фраза: «Не изобретайте велосипед».

В общем смысле я признаю мудрость этого старого изречения, особенно применительно к разработке программного обеспечения. Предполагается, что программисты всегда должны работать в рамках известных моделей и как можно быстрее добиваться нужного результата. В области программной инженерии в ходу масса выражений, принижающих роль, казалось бы, ненужных экспериментов и проб: не дергайся при бритье, не изобретай велосипед, самоварное золото, конфигурирование, подстройка, баловство, переделка или изготовление снежинок под заказ. Кроме того, часто можно услышать: «Самая короткая дорога та, которую знаешь». И действительно, самые титулованные разработчики программного обеспечения гордо стоят на плечах гигантов и следуют установленным методикам и стандартам. Самооправданием такого подхода служит лозунг: «Это придумано не нами». Точно придерживаться плана, оставаться сосредоточенным, не тратить времени зря и делать только то, что нам уже знакомо.

Если и есть сообщество разработчиков программного обеспечения, которое отвергает такое мировоззрение, то это программисты на JavaScript. Постоянно меняющиеся возможности браузеров, нескончаемый приток разработчиков с различным опытом работы в других областях и часто пересматриваемые стандарты самого JavaScript – все это вместе определяет непрерывную смену методик.

Обновление подходов здесь является делом обычным, и так было всегда. При взаимодействии с DOM в браузере возникали проблемы, и эта область стала следующей целью для обновления. Причина появления на свет Sizzle, JQuery и в конечном итоге встроенной функции `querySelectorAll` кроется в фундаментальной неудовлетворенности существующими стандартами. Форматы XML и JSON из просто применяемых на практике превратились в доминирующие стандарты обмена информацией в Интернете. Загрузите любой современный JavaScript-фреймворк и вы убедитесь, что он построен на основе сразу нескольких моделей. Взгляните на список велосипедов различной формы и размеров: MVVM, MVC, MVW, MVP, Chain of Responsibility, PubSub, Event-Driven, Declarative, Functional, Object-Oriented, Modules, Prototypes. Единственно правильного способа

разработки программ просто не существует. Кроме того, даже беглый взгляд на мир препроцессоров, таких как CoffeeScript, LiveScript, Babel, Typescript и ArnoldC, показывает, что разработчики заняты лихорадочным обновлением и самого JavaScript. Здесь ничто не является неприкосновенным, и, возможно, именно поэтому JavaScript развивается так быстро.

Я хорошо помню свое первое знакомство с React. Я тогда присутствовал на довольно известной конференции в Сан-Франциско и во время обеда имел счастье сидеть рядом с разработчиками из Facebook и Khan Academy, которые вели оживленую беседу. В то время самыми популярными инструментами были Ember, Backbone и, конечно же, Angular (в рамках конференции им было посвящено что-то около тридцати докладов). Мы начали обсуждать плюсы и минусы существующих инструментов, и разговор зашел о проблемах, возникающих из-за сложившегося мнения об абстрактности веб-приложений. Именно тогда человек, сидящий рядом со мной, сказал: «Может быть, вам нужно присоединиться к сообществу React», – и пригласил меня послушать его доклад, который должен был состояться в тот же день. Конечно же, я принял приглашение. Закончилось тем, что самой полезной (и самой спорной) презентацией стала та, на которую я пошел.

Мой собеседник за обедом, который представился как Пит Хант (Pete Hunt), оказался ключевым действующим лицом, продвигающим новый взгляд на веб-приложения. Я присутствовал на его выступлении и сразу понял, что наблюдаю очередное великое изобретение велосипеда в JavaScript. Обычные двухсторонние методы связывания данных были отброшены и заменены на более четкий односторонний поток данных, а стандартный шаблон MVC организации приложений был переосмыслен и преобразован в действия, хранилища и диспетчеры. Однако самой интересной и радикальной особенностью React был его способ обращения с DOM, который состоял в его полной и бескомпромиссной перестройке с нуля из JavaScript.

Если вы выбрали эту книгу, значит, вас интересует будущее JavaScript. Эта часто упоминаемая тема переосмысления сейчас более актуальна, чем когда-либо в последние несколько лет. React, ES6, современные системы сборки, скаффолдинг и многое другое являются новыми инструментами, заполняющими ландшафт JavaScript. Эта книга важна, потому что она рассматривает React без отрыва от современной экосистемы. Прочитав ее, вы освоите принципы, необходимые для проектирования и разработки реальных приложений, и в конечном итоге научитесь применять их на практике.

Я не могу представить себе лучшего проводника в этом захватывающем путешествии по передовым областям проектирования приложений, чем Адам. Я познакомился с ним, когда был студентом, и с тех пор не раз имел удовольствие слышать его выступления на конференции Thunder Plains, посвященной самым современным и грандиозным веб-разработкам. Он представлял прихотливую коллекцию своих личных проектов, таких как посадка в игре со смещением средней точки и полностью переделанный механизм бросания лучей в трехмерной графике на чистом JavaScript.

Адам – уникальный программист. Он работает как ученый, мастер и художник. Он не боится перестраивать существующие системы, чтобы лучше в них разобраться, и экспериментирует с новыми подходами, чтобы найти более эффективные способы достижения своих целей. Чтобы разобраться в этих новых захватывающих событиях в мире JavaScript, вам потребуется гид, который поможет критически взглянуть на них, заняться их исследованием и сделать для себя открытия.

Другим вашим гидом станет Райан Вайс, на протяжении многих лет трижды удостаиваемый звания Microsoft MVP, автор книг по промышленной разработке, которые часто обсуждаются на различных отраслевых мероприятиях, а также закаленный в сражениях разработчик программного обеспечения. И что еще более важно, Райан создал собственную компанию Vice Software LLC, которая основывает свои решения на React. Его реальный опыт в разработке проектов, основанных на React, характеризует его как отличного наставника, способного помочь вам перейти к созданию собственных передовых веб-приложений.

Велосипеды тоже нуждаются в обновлении. Если вы с этим не согласны, попробуйте поставить на свой автомобиль колеса, некогда изобретенные впервые. Будьте последовательны в своих убеждениях и прокатитесь с ветерком по шоссе, двигаясь на громоздких каменных дисках. А я буду мечтать о летающих автомобилях и делать ставку на JavaScript.

Джесси Харлин (*Jesse Harlin*),
http://jesseharlin.net/,
разработчик на JavaScript и лидер сообщества.

Об авторе

Адам Хортон (Adam Horton) – ярый поклонник старых игр, а также создатель, разрушитель и перестройщик всего и вся в Сети, вычислительной технике и компьютерных играх. Начинал карьеру как разработчик встроенного программного обеспечения в подразделении по разработке высокопроизводительных серверов Superdome в компании Hewlett Packard. Там он с помощью JavaScript и C управлял питанием, охлаждением, контролем исправности и настройкой этих чудовищных устройств. Затем он занимался веб-разработкой для PayPal, используя кросс-доменные технологии JavaScript, основанные на приемах идентификации пользователей. В последнее время работает в ESO Solutions ведущим разработчиком на JavaScript и занимается созданием приложений следующего поколения для сбора добольничных электронных медицинских карт (Electronic Health Record, EHR).

Адам верит в общедоступность, повсеместность и открытость Интернета. Ценит прагматизм и преобладание практики над догмой при проектировании и разработке вычислительных приложений и в образовании.

Я хотел бы поблагодарить мою жену за ее бесконечное терпение и поддержку. Она – как попутный ветер, подталкивающий меня во всех моих начинаниях, в том числе и при работе над этой книгой. Я также хотел бы поблагодарить своих родителей за то, что заботливо заправляют мою ракету, пока я играю с системой наведения.

Райан Вайс (Ryan Vice) – основатель и главный разработчик компании Vice Software, специализирующейся на решениях по индивидуальным заказам клиентов, желающих вывести свои MVP на рынок или модернизировать существующие приложения. Vice Software предлагает не только весьма конкурентоспособные цены по всем направлениям, но и основывает свою ценовую политику на сложности задач, что позволяет клиентам оплачивать решения по ставке высококлассного специалиста только при возникновении такой необходимости, а на более простые работы расценки гораздо ниже. Райан трижды был удостоен звания Microsoft MVP, является автором еще одной книги

по архитектуре программного обеспечения, а также часто выступает на конференциях и принимает участие в мероприятиях, проходящих в Техасе. Кроме того, Райану выпала удача жениться на Хизер, и он большую часть своего свободного времени тратит на попытки не отставать от трех своих детей: Грейс, Дилана и Ноя.

О техническом рецензенте

Тунг Дао (Tung Dao) – специалист по комплексной разработке веб-приложений с многолетним опытом создания веб-сайтов и служб.

Ныне работает инженером-программистом в FPT Software, во Вьетнаме, где разрабатывает веб-службы RESTful, основанные на NoSQL и Elasticsearch. В свободное время занимается созданием веб-приложений на Clojure/Go или возится со своим Raspberry Pi.

Клиентские части своих веб-приложений в основном пишет на ClojureScript/Reagent (библиотека для связи React с Clojure). При работе над библиотекой применил несколько идей из React. Эта книга послужила ему напоминанием, так как он работает со следующим поколением JavaScript (ES6) и повторно осмысливает основные идеи React.

Большое спасибо авторам и сотрудникам Packt Publishing за их напряженную работу и поддержку.

Предисловие

Книга, которую вы читаете, является плодом сотрудничества между мною, ежедневно занимающимся разработкой веб-приложений, и Райаном Вайсом, ветераном .NET, экспертом, перешедшим на разработку веб-приложений, и предпринимателем. Я познакомился с Райаном, когда компания, в которой я сейчас работаю, разрабатывала весьма сложные веб-приложения для подразделений быстрого реагирования при чрезвычайных ситуациях, таких как аварийно-спасательная медицинская служба. В то время компания вносила революционные изменения в свой флагманский продукт. И тогда React находился на ранней стадии развития, но он предлагал нечто, чего не мог предоставить ни один другой MV*-фреймворк: новый подход к молниеносному отображению. Поскольку скорость имеет первостепенное значение для служб быстрого реагирования, мы решили применить React. Когда появилась возможность написать о React, Райан пригласил меня в помощники, и в результате появилась эта книга.

Сейчас пришло время технологий, подобных React. Открытая веб-платформа достигла успеха, следуя простому правилу безаварийной работы при любых условиях. В результате модель DOM, отвечающая за представление данных при отображении веб-страниц, стала огромной и громоздкой. Малейшее изменение в DOM вызывает шквал расчетов и согласований лишь для того, чтобы обновить несколько пикселей на экране. React же рассматривает DOM как затратный ресурс и минимизирует объем операций, производимых с ним. Сэкономленное при этом время можно потратить на выполнение сложной логики самого приложения.

Эта книга посвящена фундаментальным основам React и прагматическому подходу к созданию веб-приложений. Она научит вас выбирать инструменты, подходящие для конкретных нужд. В первых трех главах рассматриваются основы React, такие как состояние, свойства и JSX, а также более сложные темы, связанные с формами и валидацией. Глава 4 «Анатомия React-приложений» содержит сведения, обычно опускаемые в книгах, подобных этой, об анатомии веб-приложений и некоторых методах разработки, которые помогут вам сформулировать четкий план при создании собственных приложений. В главах с 5 по 9 будет описан процесс разработки многопользо-

вательского блога с помощью подходов и библиотек, рассмотренных в двух предыдущих главах. И наконец, при чтении *главы 10 «Анимация в React»* можно будет развлечься, познакомившись с массой способов создания с помощью React отличных анимаций.

О чем рассказывается в этой книге

Глава 1 «Введение в React» посвящена основам React. Она начинается с простого примера Hello World и затем переходит к описанию типов React-сущностей и их определению.

Глава 2 «Соединение компонентов и цикл их существования» содержит описание ключевых компонентов и приемов управления их состоянием, от включения в DOM до исключения из DOM.

Глава 3 «Динамические компоненты, примеси, формы и прочие элементы JSX» посвящена основам создания форм в React и шаблонам валидации в React.

Глава 4 «Анатомия React-приложений» описывает подходы к разработке веб-приложений и разъясняет порядок выбора из обширного списка веб-технологий, доступных в рамках React. Здесь мы попрактикуемся в применении технологий разработки и научимся создавать артефакты, направляющие развитие.

Глава 5 «Начало работы над React-приложением» начинает разработку полнофункционального многопользовательского блога на основе React. Здесь мы подготовим среду разработки, установим все инструменты и создадим представления для приложения.

Глава 6 «Реализация React-приложения блога, часть 1: действия и общие компоненты» описывает стратегию взаимодействий компонентов приложения с помощью Reflux Actions. Здесь будет создано несколько общих компонентов.

Глава 7 «Реализация React-приложения блога, часть 2: пользователи» описывает реализацию управления учетными записями пользователей в приложении.

Глава 8 «Реализация React-приложения блога, часть 3: статьи» рассматривает создание и просмотр статей в блоге.

Глава 9 «Реализация React-приложения блога, часть 4: бесконечная прокрутка и поиск» описывает добавление двух возможностей: бесконечной прокрутки с загрузкой и поиска.

Глава 10 «Анимация в React» рассматривает технологии веб-анимации в React.

Что потребуется для работы с книгой

Все программное обеспечение, используемое в этой книге, распространяется с открытым исходным кодом и является бесплатным. Вам понадобятся веб-браузер (естественно) и умение устанавливать Node и npm. Некоторые действия придется выполнять в командной строке. Для таких задач рекомендуется использовать Bash. Он доступен для OSX, Linux, а также Windows, через Git (git-bash). В книге используются Node 4.x и React 0.14.

Кому адресована эта книга

Эта книга ориентирована на профессиональных разработчиков веб-приложений, разбирающихся в JavaScript, CSS и имеющих представление, как работают веб-браузеры. Предыдущий опыт работы с другими фреймворками необязателен, но полезен. Умение работать в командной строке упростит восприятие.

Соглашения

В этой книге используется несколько различных стилей оформления текста для выделения разных видов информации. Ниже приводятся примеры этих стилей с объяснением их назначения.

Программный код в тексте, имена таблиц баз данных, имена папок, имена файлов, расширения файлов, адреса страниц в Интернете, ввод пользователя и ссылки в Twitter будут выглядеть так: «Далее был определен литерал второго объекта с методом `componentWillMount`, вызывающий метод console.log и передающий ему `componentWillMount` из `ReactMixin2`».

Блоки программного кода оформляются так:

```
var path    = require('path')
,   webpack = require('webpack')
;
```

Чтобы привлечь внимание к определенному фрагменту в коде, соответствующие строки или элементы будут выделены жирным:

```
    PostStore.getPostsByPage( this.state.page,
      Object.assign({}, this.state.search ? {q: this.state.search} :
{}, this.props)
    ).then(function (results) { var data = results.results;
```

Новые термины и **важные слова** будут выделены жирным. Текст, отображаемый на экране, например в меню или в диалогах, будет оформляться так: «Нажмите кнопку **OK** для просмотра вывода».

Предупреждения или важные сообщения будут выделены так.

Подсказки и советы будут выглядеть так.

О примерах кода

Первые три главы этой книги содержат начальные сведения о React. К этим главам прилагаются краткие примеры, иллюстрирующие основные идеи React, которые запускаются в интернет-браузере. Такой же формат имеют примеры анимации к *главе 10* «*Анимация в React*». Примеры для остальных глав, с 4-й по 9-ю, поставляются в виде ZIP-файлов, которые можно загрузить на сайте www.dmkpress.com или www.дмк.рф в разделе **Читателям – Файлы к книгам**.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф в разделе **Читателям – Файлы к книгам**.

Как опробовать примеры

После знакоства с несколькими первыми примерами мы перейдем к использованию преимуществ интеграции JsFiddle (http://JsFiddle.net/), с онлайн-окружением JavaScript и хранилищами Gist GitHub. Интеграция позволяет открывать любые ссылки на хранилище Gist в отладчике JsFiddle и выполнять код в браузере. На рис. 0.1 показано, как выглядит окно Fiddle.

Fiddle: http://j.mp/MasteringReact-1-2-1-Fiddle.

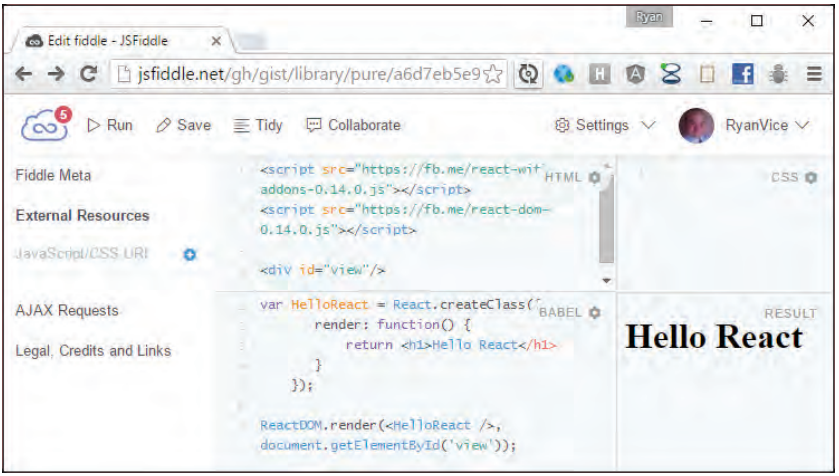

Рис. 0.1 ❖ Окно **JsFiddle**

Как видите, окно **JsFiddle** содержит четыре раздела:

1. Раздел **HTML**: разметка HTML, определяющая внешний вид страницы.
2. Раздел **CSS**: разметка CSS, определяющая стиль страницы.
3. Раздел **Babel**: код на JavaScript, загружаемый и выполняемый страницей. Обратите внимание, что Babel – это компилятор JavaScript, преобразующий код JSX в JavaScript.
4. Окно **Result**: результат объединения HTML, CSS и JavaScript и совместного их выполнения.

Я рекомендую при работе с примерами открывать окна **JsFiddle** и получать готовую страницу щелчком на кнопке **Run**.

Открыв пример кода, поэкспериментируйте с ним, чтобы убедиться, что понимаете, как он работает.

Мы коснемся некоторых основ использования JsFiddle, но, если вы еще незнакомы с этим инструментом, желательно ознакомиться с его документацией на сайте http://doc.jsfiddle.net/.

JsFiddle дает возможность поэкспериментировать с кодом, и вы поймете, насколько хорош этот инструмент, позволяющий в интерактивном режиме быстро освоить идеи, описываемые здесь.

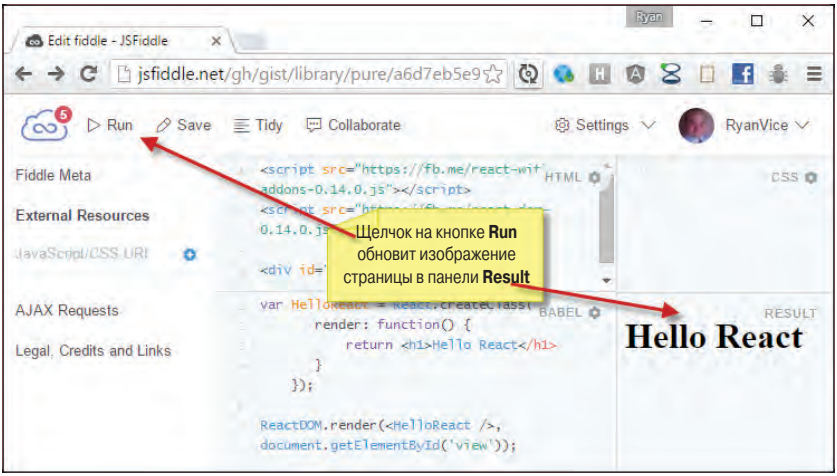

Рис. 0.2 ❖ Кнопка **Run** и отображение в окне **JsFiddle**

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв прямо на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com, при этом напишите название книги в теме письма.

Если есть тема, в которой вы квалифицированы, и вы заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы удостовериться в качестве наших текстов, ошибки всё равно случаются. Если вы найдёте ошибку в одной из наших книг – возможно, ошибку в тексте или в коде, – мы будем очень благодарны, если вы сообщите нам

о ней. Сделав это, вы избавите других читателей от расстройств и поможете нам улучшить последующие версии данной книги.

Если вы найдёте какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в Интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и «Packt» очень серьезно относятся к вопросам защиты авторских прав и лицензирования. Если вы столкнетесь в Интернете с незаконно выполненной копией любой нашей книги, пожалуйста, сообщите нам адрес копии или веб-сайта, чтобы мы могли принять меры.

Пожалуйста, свяжитесь с нами по адресу электронной почты dmkpress@gmail.com со ссылкой на подозрительные материалы.

Мы высоко ценим любую помощь по защите наших авторов, помогающую нам предоставлять вам качественные материалы.

Глава 1

Введение в React

В этой книге обсуждается сложная тема использования React, но нам хотелось написать учебник для начинающих и сделать эту книгу одновременно исчерпывающей и доступной. Мы не будем тратить много времени на описание всех тонкостей каждой из рассматриваемых технологий. Вместо этого мы будем изучать короткие примеры, иллюстрирующие разные инструменты и технологии. Также в соответствующих местах будут приводиться ссылки, упрощающие доступ к коду и его опробование.

В этой главе будут рассмотрены следующие вопросы:

- о примерах кода;
- пример «Hello World» для React;
- JSX;
- свойства;
- состояния.

Пример Hello React

Первый пример, который мы рассмотрим, – это работоспособный пример в стиле **Hello World** для React. Чтобы создать его, выполните следующие действия:

1. Создайте новый HTML-файл с именем `hello-react.html`.
2. Вставьте в него следующий код:

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title>Hello React</title>
    <script src="https://fb.me/react-with-addons-0.14.0.js">
 </script>
```

```
    <script src="https://fb.me/react-dom-0.14.0.js">
 </script>
</head>
<body>

<script>
    var HelloReact = React.createClass({
              render: function() {
                  return React.DOM.h1(null, 'Hello React');
              }
          });

    ReactDOM.render(React.createElement(HelloReact), document.body);
</script>
</body>
</html>
```

3. Открыв файл `hello-react.html` в браузере, вы должны увидеть страницу, как на рис. 1.1.

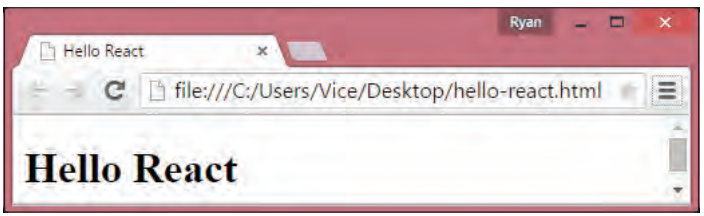

Рис. 1.1 ❖ Страница `hello-react.html` в браузере

React – это фреймворк с компонентной организацией, позволяющий собирать представления из компонентов. Чтобы создать компонент React, прежде всего нужно подключить исходные коды React, как показано ниже:

```
<script src="https://fb.me/react-with-addons-0.14.0.js"></script>
<script src="https://fb.me/react-dom-0.14.0.js"></script>
```

Начиная с версии 0.13.0 программный интерфейс React API был разбит на два файла. Один файл содержит код управления деревом DOM, а другой – остальную часть программного интерфейса React API. Причиной тому стало широкое распространение React, и в настоящее время React Native используется для создания мобильных приложений. Его также можно использовать для разработки настольных приложений Windows и Mac с применением таких платформ, как Electron.

Этот код подключает версию v0.14.0 React. В следующем блоке `script` имеется код, создающий параметр `ReactElement` типа `h1` и записывающий в дочерний элемент строку `Hello React`:

```
React.createElement('h1', null, 'Hello React');
```

Далее параметр `ReactElement` передается в первом аргументе методу ReactDOM.render(), как показано ниже:

```
ReactDOM.render(
    React.createElement('h1', null, 'Hello React'), document.body);
```

Метод `render` принимает в первом аргументе параметр `ReactElement`, DOM-элемент document.body – во втором, вставляет разметку HTML, генерируемую первым аргументом `ReactElement`, как дочерний элемент во второй аргумент document.body. На рис. 1.2 изображен результат на вкладке **Elements** в браузере Chrome:

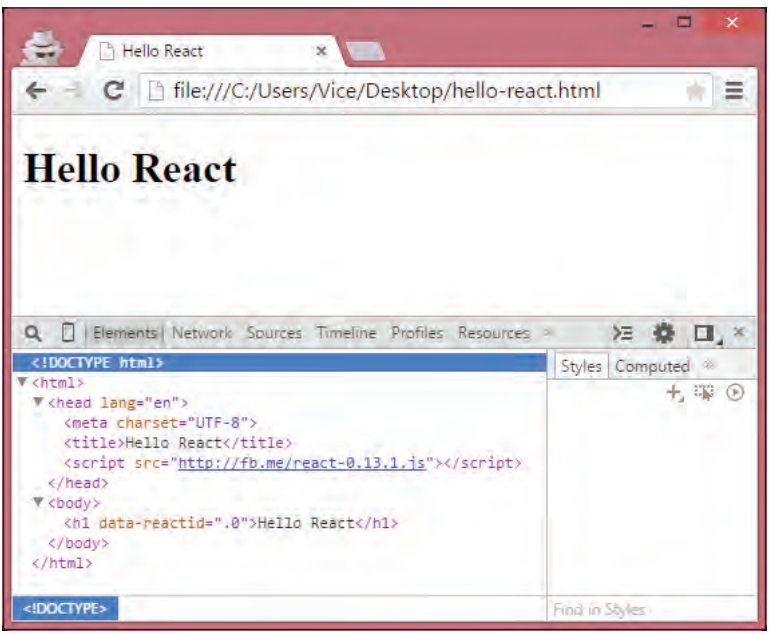

Рис. 1.2 ❖ Результат на вкладке **Elements** в браузере Chrome

Как видите, теперь у нас в дереве DOM есть элемент `h1` со строкой `Hello React`.

Однако вас может удивить, что, несмотря на заявление о компонентной организации фреймворка, до сих пор не было создано ни одного компонента. Компоненты – это именно то, что превращает React в мощный и гибкий фреймворк, поэтому давайте посмотрим, как будет выглядеть наш пример, если в нем создать компонент. Для этого измените файл `hello-react.html`, как показано ниже, и обновите страницу в браузере, чтобы убедиться, что программа по-прежнему производит тот же результат:

```
var HelloReact = React.createClass({ render: function() {
      return React.DOM.h1(null, 'Hello React');
      }
   });

ReactDOM.render(React.createElement(HelloReact), document.body);
```

Мы создали JavaScript-переменную `HelloReact` и присвоили ей компонент, созданный вызовом метода React.createClass():

```
var HelloReact = React.createClass({
              render: function() {
                  return React.DOM.h1(null, 'Hello React');
              }
          });
```

Метод `createClass()` принимает объект, реализующий метод `render`, который возвращает единственный отображаемый объект класса `ReactClass`. Здесь был создан объект `ReactClass`, который представляет DOM-элемент `h1` со строкой `Hello React`.

Обратите внимание, что для создания экземпляра `h1` элемента `ReactElement` был использован программный интерфейс `React.DOM`. Этот программный интерфейс имеет удобные методы для создания элементов HTML и внутренне вызывает метод `React.createElement()` с необходимыми параметрами. Может показаться странным, что программный интерфейс `React.DOM` не был включен в `ReactDOM`, подобно `ReactDOM.Render`. Однако команда разработчиков React решила, что элементы HTML и виджеты являются частью их универсального подхода к созданию пользовательских интерфейсов, а фактическое отображение зависит от платформы. Ниже приведена выдержка из документации React, посвященная реструктуризации.

«При просмотре таких пакетов, как react-native, react-art, react-canvas и react-three, становится ясно, что элегантность и сущность решений React не имеют ничего общего с браузерами или DOM.

> Чтобы подчеркнуть это и облегчить сборку для множества окружений, где React может выполнять отображение, мы разделили основной пакет React на две части: `react` и `react-dom`. Это позволяет писать компоненты, которые могут совместно использоваться в веб-версии React и React Native. Мы не планируем совместно применять в приложениях весь код, но хотим дать возможность одновременного использования компонентов, которые будут работать одинаково на разных платформах.
> Пакет React содержит `React.createElement`, `.createClass`, `.Component`, `.PropTypes`, `.Children` и другие вспомогательные методы, имеющие отношение к классам элементов и компонентов. Мы считаем их изоморфными, или универсальными, помощниками, которые вам понадобятся для создания компонентов».

Как показано в следующем фрагменте, в вызов метода `React.render()` передается результат вызова `React.createElement(HelloReact)`:

```
ReactDOM.render(React.createElement(HelloReact), document.body);
```

Теперь мы добавили в код создание React-компонента, и, как убедимся ниже, такие компоненты и придают мощь и гибкость веб-приложениям.

Исходный код: http://bit.ly/MasteringReact-1-1-Gist.

JSX

Чтобы упростить работу с программным интерфейсом компонентов, в React предусмотрен собственный синтаксис JSX, сочетающий в себе JavaScript и HTML. Измените пример в `hello-react.html`, добавив в него код JSX, как показано ниже, и обновите страницу в браузере:

```
<!DOCTYPE html>
<html>
<head lang="en">
    <meta charset="UTF-8">
    <title>Hello React</title>
    <script src="https://fb.me/react-with-addons-0.14.0.js"></script>
    <script src="https://fb.me/react-dom-0.14.0.js"></script>
  <script src="http://fb.me/JSXTransformer-0.13.1.js"></script>
</head>
<body>

<script type="text/jsx">
    var HelloReact = React.createClass({
```

```
            render: function() {
                return React.DOM.h1(null, 'Hello React');
            }
        });

    ReactDOM.render(React.createElement(HelloReact), document.body);
</script>
</body>
</html>
```

Обратите внимание на присутствие атрибута `type="text/jsx"` в теге `script`.

Как это работает

Прежде всего мы добавили ссылку на библиотеку JSX Transformer, как показано ниже:

```
<script src="http://fb.me/JSXTransformer-0.13.1.js"></script>
```

Обратите внимание, что библиотеку Transformer рекомендуется использовать в браузере только для тестирования. В следующих главах будут описаны более удобные способы преобразования кода JSX в JavaScript. Также отметьте, что начиная с версии React 0.14 запрещается применять JSX Transformer для преобразования JSX и рекомендуется использовать Babel.

Затем изменили код создания компонента:

```
var HelloReact = React.createClass({
    render: function() {
        return <h1>Hello React</h1>;
    }
});
```

Теперь вместо вызова методов React API для определения компонентов структуры DOM нужная структура DOM просто прописывается в операторе `return`.

Как мы увидим позже, имеется даже возможность вставки ссылок на другие React-компоненты, подобных ссылкам на HTML-элементы!

И для включения компонента в DOM можно просто передать его в виде HTML-тега в метод `React.render()`:

```
ReactDOM.render(<HelloReact />, document.body);
```

Обратите внимание, что нет необходимости вызывать `React.createElement()`, – его создаст компилятор JSX.

Исходный код: http://j.mp/MasteringReact-1-2-Gist.

Теперь, узнав, как создать страницу для React-приложения, приступим к использованию JsFiddle для просмотра примеров. Только что рассмотренный пример можно увидеть в JsFiddle по адресу: http://j.mp/MasteringReact-1-2-1-Fiddle.

Перейдите по этой ссылке и щелкните на кнопке **Run**. В результате вы должны увидеть картину, изображенную на рис. 1.3.

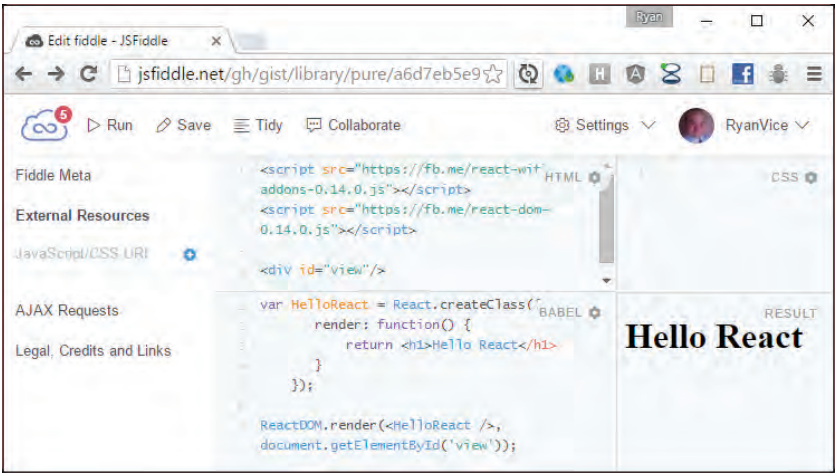

Рис. 1.3 ❖ После щелчка на кнопке **Run**

Декомпиляция JSX

На сайте Babel можно найти инструмент, позволяющий увидеть декомпилированный код JavaScript, который будет создан при преобразовании JSX. На рис. 1.4 показан инструмент **Babel**, который можно найти по адресу: https://babeljs.io/repl/.

Компилятор JSX имеет два окна для отображения кода. В окно слева можно ввести код на JSX, а в окне справа – увидеть результат его компиляции в React API.

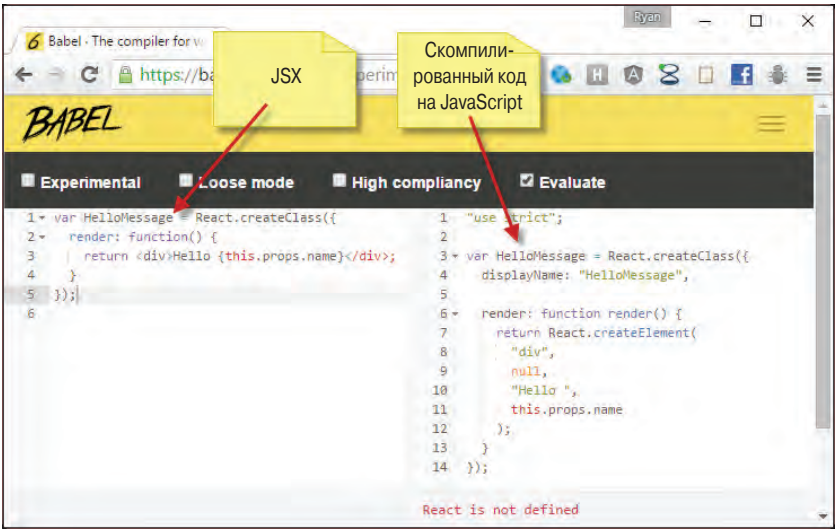

Рис.1.4 ❖ Инструмент **Babel**

Структура результата отображения

Имеется несколько тонких моментов, которые следует учитывать при использовании JSX для создания параметра `ReactElement`, возвращаемого методом `render()` компонента. Допустим, нужно вывести приветствие `Hello React How are you?` в двух строках. Можно попытаться структурировать код, как показано ниже:

```
var HelloMessage = React.createClass({
  render: function() {
    return <div>Hello React</div> // ошибка
      <div>How are you?</div>;
  }
});
```

Но если передать этот код компилятору JSX, появится следующее сообщение об ошибке:

```
Error: Parse Error: Line 1: Adjacent JSX elements must be wrapped in an
enclosing tag
```

Сообщение указывает, что код JSX необходимо заключить в один родительский элемент, как показано ниже: