REACTJS*, JAVASCRIPT*

Вышел React v16.0

ИЗ ПЕСОЧНИЦЫ

dagen 28 сентября 2017 в 16:42

Это перевод поста Эндрю Кларка о выходе столь ожидаемой версии React. Оригинальный пост в блоге React.

Мы с удовольствием сообщаем о выходе React v16.0! Среди изменений некоторые давно ожидаемые нововведения, например **фрагменты**, **обработка ошибок (error boundaries)**, **порталы**, поддержка **произвольных DOM-атрибутов**, улучшения в **серверном рендере**, и **уменьшенный размер файла**.

Новые типы для рендера: фрагменты и строки

Теперь вы можете вернуть массив элементов из `render`-метода компонента. Как и с другими массивами, вам надо добавлять ключ к каждому элементу, чтобы реакт не ругнулся варнингом:

```
render() {
  // Нет необходимости оборачивать в дополнительный элемент!
  return [
    // Не забудьте добавить ключи :)
    <li key="A">Первый элемент</li>,
    <li key="B">Второй элемент</li>,
    <li key="C">Третий элемент</li>,
  ];
}
```

В будущем мы, вероятно, добавим специальный синтакс для вывода фрагментов, который не будет требовать явного указания ключей.

Мы добавили поддержку и для возврата строк:

```
render() {
  return 'Мама, смотри, нет лишних спанов!';
}
```

Полный список поддерживаемых типов.

Улучшенная обработка ошибок

Ранее ошибки рендера во время исполнения могли полностью сломать ваше приложение, и описание ошибок часто было малоинформативным, а выход из такой ситуации был только в перезагрузке страницы. Для решения этой проблемы React 16 использует более надёжный подход. По-умолчанию, если ошибка появилась внутри рендера компонента или в lifecycle-методе, всё дерево компонентов отмонтируется от корневого узла. Это позволяет избежать отображения неправильных данных. Тем не менее, это не очень дружелюбный для пользователей вариант.

Вместо отмонтирования всего приложения при каждой ошибке, вы можете использовать *error boundaries*. Это специальные компоненты, которые перехватывают ошибки в своём поддереве и позволяют вывести резервный UI. Воспринимайте их как try-catch операторы, но для React-компонентов.

Для дальнейшей информации проверьте наш недавний пост про обработку ошибок в React 16.

Порталы

Порталы дают удобный способ рендера дочерних компонентов в DOM-узел, который находится за пределами дерева родительского компонента.

```
render() {
  // React не создаёт новый див. Он рендерит дочерние элементы в
domNode.
  // domNode - это любой валидный DOM-узел,
  // вне зависимости от его расположения в DOM-дереве.
  return ReactDOM.createPortal(
    this.props.children,
    domNode,
  );
}
```

Полный пример находится в документации по порталам.

Улучшенный серверный рендеринг

React 16 содержит полностью переписанный серверный рендерер, и он действительно быстрый. Он поддерживает **стриминг**, так что вы можете быстрее начинать отправлять байты клиенту. И благодаря новой стратегии сборки, которая убирает из кода обращения к `process.env` (хотите верьте, хотите — нет, но чтение `process.env` в Node очень медленное!), вам больше не надо

бандлить React для получения хорошей производительности серверного рендеринга.

Ключевой разработчик Саша Айкин написал замечательную статью, рассказывающую об улучшениях SSR в React 16. Согласно Сашиным бенчмаркам, серверный рендеринг в React 16 примерно **в 3 раза быстрее**, чем в React 15. При сравнении с рендером в React 15 с убранным из кода `process.env` получается ускорение в 2.4 раза в Node 4, около 3-х раз в Node 6 и аж в 3.8 раза в Node 8.4. А если вы сравните с React 15 без компиляции (без убранного `process.env`), React 16 оказывается на порядок быстрее в последней версии Node! (Как Саша указал, к этим синтетическим бенчмаркам надо относиться осторожно, так как они могут не отображать производительность реальных приложений).

Более того, React 16 лучше в восстановлении отрендеренного на сервере HTML, когда последний приходит в браузер. Больше не требуется начальный рендер, используемый для проверки результатов с сервера. Вместо этого он будет пытаться переиспользовать как можно больше существующего DOM. Больше не будут использоваться контрольные суммы! В целом мы не рекомендуем рендерить на клиенте отличающийся от сервера контент, но это может быть полезно в некоторых сценариях (например вывод меток времени).

Больше в документации по ReactDOMServer.

Поддержка произвольных DOM-атрибутов

Вместо игнорирования неизвестных HTML и SVG атрибутов, теперь React будет просто передавать их в DOM. Вдобавок это позволяет нам отказаться от длиннющих списков разрешённых атрибутов, что уменьшает размер бандла.

Уменьшенный размер файла

Несмотря на все эти нововведения, React 16 **меньше**, чем React 15.6.1!

- `react` весит 5.3 kb (2.2 kb gzipped), по сравнению с 20.7 kb (6.9 kb gzipped) ранее.
- `react-dom` весит 103.7 kb (32.6 kb gzipped), по сравнению с 141 kb (42.9 kb gzipped) ранее.
- `react` + `react-dom` вместе 109 kb (34.8 kb gzipped), по сравнению 161.7 kb (49.8 kb gzipped) ранее.

В общей сложности размер уменьшился на 32% по сравнению с прошлой версией (30% после gzip-сжатия).

На уменьшение размера частично влияют изменения в сборке. React теперь использует Rollup для создания "плоских" бандлов (*по-видимому Эндрю Кларк тут имел ввиду "scope hoisting", что давно было в Rollup, но в Webpack появилось только в третьей версии*) всех поддерживаемых форматов, что привело к выигрышу и в размере и в скорости работы. Также плоский формат бандла приводит к тому, что воздействие React на бандл приложения остаётся одинаковым, независимо от того, как вы доставляете

свой код конечным пользователям, напр. используя Webpack, Browserify, уже собранные UMD-модули или любой другой способ.

MIT лицензия

Если вы вдруг пропустили, React 16 теперь доступен под MIT лицензией. А для тех, кто не может обновиться немедленно, мы выложили версию React 15.6.2 под MIT.

Новая архитектура ядра

React 16 — это первая версия React, построенная на основе новой архитектуры, называемой Fiber. Вы можете почитать всё об этом проекте в инженерном блоге Facebook. (Спойлер: мы полностью переписали React!)

Fiber затрагивает большинство новых фич в React 16, такие как error boundaries или фрагменты. Через несколько релизов вы увидите несколько новых фич, так как мы будем постепенно раскрывать потенциал React.

Наверно наиболее впечатляющее нововведение, над которым мы работаем — это **асинхронный рендеринг**, позволяющий компонентам использовать кооперативную многозадачность в рамках рендера через периодическую передачу управления браузеру. Итог — с асинхронным рендерингом приложения более отзывчивы, так как React не блокирует главный поток.

Следующее демо даёт нам взглянуть на суть проблемы, решаемую асинхронным рендерингом (подсказка: обратите внимание на крутящийся черный квадрат).

> Ever wonder what "async rendering" means? Here's a demo of how to coordinate an async React tree with non-React work https://t.co/3snoahB3uV pic.twitter.com/egQ988gBjR
>
> — Andrew Clark (@acdlite) September 18, 2017

Мы думаем, что асинхронный рендеринг — очень важная вещь, двигающая React в будущее. Чтобы сделать переход на v16.0 как можно более безболезненным, мы пока не включили какие-либо асинхронные фичи, но мы рады выкатить их в ближайшие месяцы. Следите за обновлениями!

Установка

React v16.0.0 доступен в npm репозитории.

Для установки React 16 используя Yarn:

```
yarn add react@^16.0.0 react-dom@^16.0.0
```

Для установки React 16 используя npm:

```
npm install --save react@^16.0.0 react-dom@^16.0.0
```

Мы также предоставляем UMD-вариант, выложенный на CDN:

```
<script crossorigin
src="https://unpkg.com/react@16/umd/react.production.min.js">
</script>
<script crossorigin src="https://unpkg.com/react-dom@16/umd/react-
dom.production.min.js"></script>
```

Ссылка на документацию по детальным инструкциям по установке.

Переход со старой версии

Хотя React 16 включает значительные внутренние изменения, в случае обновления вы можете относится к этому релизу как к любому обычному мажорному релизу React. Мы используем React 16 в Facebook и Messenger.com с начала этого года, мы выкатили несколько бета-версий и релиз кандидатов, чтобы максимально исключить возможные проблемы. Если не учитывать некоторые нюансы, то **ваше приложение должно работать с 16-й версией, если с 15.6 оно работало без каких-либо варнингов**.

Устаревшие методы

Восстановление отрендеренного на сервере кода теперь имеет явное API. Для восстановления HTML вам надо использовать `ReactDOM.hydrate` вместо `ReactDOM.render`. Продолжайте использовать `ReactDOM.render`, если вы рендерите только на клиентской стороне.

React Addons

Как ранее было объявлено, мы прекращаем поддержку React Addons. Мы ожидаем, что последняя версия каждого дополнения (кроме`react-addons-perf`; см. ниже) будет работоспособна в ближайшем будущем, но мы не будем публиковать новых обновлений.

По ссылке есть ранее опубликованные предложения по миграции.

А `react-addons-perf` вообще не будет работать в React 16. Скорее всего мы выпустим новую версию этого инструмента в будущем. А пока вы можете использовать браузерные инструменты для измерения производительности.

Несовместимые изменения

React 16 включает несколько небольших изменений без обратной совместимости. Они влияют на редко используемые сценарии и затронут малую часть приложений.

- React 15 имел ограниченную и недокументированную поддержку error boundaries через использование `unstable_handleError`. Этот метод теперь переименован в `componentDidCatch`. Вы можете использовать codemod для автоматической миграции на новое API.
- `ReactDOM.render` и `ReactDOM.unstable_renderIntoContainer` теперь возвращают null, если вызваны из lifecycle-метода. Вместо этого теперь используйте порталы или ссылки.

- `setState`:
  - Вызов `setState` с null больше не будет вызывать реконсиляцию. Это позволит определять в коллбеке надо ли вызывать перерендер.
  - Вызов `setState` напрямую в рендере всегда вызывает реконсиляцию, чего раньше не было. Независимо от того, вам однозначно не надо вызывать setState из метода рендера.
  - Коллбек `setState`'а (второй аргумент) теперь вызывается немедленно после `componentDidMount` / `componentDidUpdate`, вместо того чтобы ждать полного рендера дерева компонентов.
- При замене `<A />` на `<B />`, `B.componentWillMount` будет вызываться всегда перед `A.componentWillUnmount`. Ранее `A.componentWillUnmount` мог вызываться раньше в некоторых случаях.
- Ранее изменение ссылки на компонент вызывало всегда обнуление ссылки перед вызовом рендера. Теперь мы изменяем ссылку позднее, когда применяем изменения к DOM.
- Небезопасно вызывать рендер контейнера, если содержимое было изменено в обход React. Ранее это работало в некоторых случаях, но никогда не поддерживалось. Мы не вызываем варнинг в этом случае. Вместо этого вы сами должны чистить дерево вашего компонента используя `ReactDOM.unmountComponentAtNode`. Посмотрите на этот пример.
- Lifecycle-метод `componentDidUpdate` больше не получает параметр `prevContext`. (см.#8631)

- Shallow рендерер больше не вызывает `componentDidUpdate`, т.к. ссылки на DOM недоступны. Это изменение делает его консистентным с методом `componentDidMount` (который тоже не вызывался в предыдущих версиях).
- Shallow рендерер больше не имеет метода `unstable_batchedUpdates`.

Сборка

- Теперь недоступны `react/lib/*` и `react-dom/lib/*`. Даже для CommonJS окружений, React и ReactDOM теперь собраны в отдельные файлы (“flat bundles”). Если ваш проект ранее зависел от недокументированных внутренних возможностей React'a и они больше не работают, дайте нам об этом знать в новом тикете, а мы постараемся придумать способ миграции для вас.
- Больше нет билда `react-with-addons.js`. Все аддоны из этого билда уже опубликованы по отдельность в npm и имеют однофайловые браузерные версии, если они нужны вам.
- Методы и возможности, помеченные устаревшими в 15.x, теперь убраны из основного пакета. `React.createClass` теперь доступен как `create-react-class`, `React.PropTypes` как `prop-types`, `React.DOM` как `react-dom-factories`, `react-addons-test-utils` как `react-dom/test-utils`, а shallow рендерер как `react-test-renderer/shallow`. См. посты в блоге 15.5.0 и 15.6.0 для инструкций по миграции кода и автоматических кодемодов.

- Имя и путь до однофайловых браузерных билдов изменились для подчёркивания различий между разработческими и боевыми билдами. Например:
  - `react/dist/react.js` → `react/umd/react.development.js`
  - `react/dist/react.min.js` → `react/umd/react.production.min.js`
  - `react-dom/dist/react-dom.js` → `react-dom/umd/react-dom.development.js`
  - `react-dom/dist/react-dom.min.js` → `react-dom/umd/react-dom.production.min.js`

Требования к JavaScript-окружению:

React 16 зависит от коллекций Map и Set. Если вы поддерживаете старые браузеры и устройства, в которых нет этого нативно (напр. IE < 11), используйте полифилы, такие как core-js или babel-polyfill.

Окружение с полифилами для React 16 используя core-js для поддержки старых браузеров может выглядеть как-то так:

```
import 'core-js/es6/map';
import 'core-js/es6/set';

import React from 'react';
import ReactDOM from 'react-dom';

ReactDOM.render(
  <h1>Hello, world!</h1>,
  document.getElementById('root')
);
```

React также требует `requestAnimationFrame` (даже в тестовых средах). Простая заглушка для тестовых окружений может выглядеть так:

```
global.requestAnimationFrame = function(callback) {
  setTimeout(callback, 0);
};
```

Благодарности

Как обычно, этот релиз был бы невозможен без наших контрибьюторов-волонтёров (open source contributors). Спасибо всем, кто заводил баги, открывал пулл-реквесты, отвечал в тикетах, писал документацию.

Отдельное спасибо нашим корневым контрибьюторам, особенно за их героические усилия в последние несколько недель пререлизного цикла: Brandon Dail, Jason Quense, Nathan Hunzaker, и Sasha Aickin.

Комментарии (31)

Похожие публикации

Распознаем коды Морзе с использованием Rx.js

17

MostovenkoAlexander • 10 ноября 2015 в 01:53

Релиз TypeScript 1.6: не только React

13

0xy • 19 октября 2015 в 14:24

5 практических примеров для изучения фреймворка React

46

ПЕРЕВОД

jojo97 • 13 июля 2014 в 03:37

Популярное за сутки

Яндекс открывает Алису для всех разработчиков. Платформа Яндекс.Диалоги (бета)

69

BarakAdama • вчера в 10:52

Почему следует игнорировать истории основателей успешных стартапов

20

ПЕРЕВОД

m1rko • вчера в 10:44

Как получить телефон (почти) любой красотки в Москве, или интересная особенность MT_FREE

24

ИЗ ПЕСОЧНИЦЫ

cab404 • вчера в 20:27

Java и Project Reactor

10

zealot_and_frenzy • вчера в 10:56

Пользовательские агрегатные и оконные функции в PostgreSQL и Oracle

6

erogov • вчера в 12:46

Лучшее на Geektimes

Как фермеры Дикого Запада организовали телефонную сеть на колючей проволоке

31

NAGru • вчера в 10:10

Энтузиаст сделал новую материнскую плату для ThinkPad X200s

49

alizar • вчера в 15:32

Кто-то посылает секс-игрушки с Amazon незнакомцам. Amazon не знает, как их остановить

85

Pochtoycom • вчера в 13:06

Илон Маск продолжает убеждать в необходимости создания колонии людей на Марсе

140

marks • вчера в 14:19

Дела шпионские (часть 1)

16

TashaFridrih • вчера в 13:16