1. Знакомство с React

Содержание урока

- Обзор;
- React, Virtual DOM и reconciliation;
- ReactElement;
- ReactComponent;
- Fiber;
- Подведём итоги.

Обзор

Привет! 👋🧑‍🎓 В этом уроке мы рассмотрим основы React, его концепцию, принципы работы и механики с высоты птичьего полёта. 🦅 «Под капотом» в React происходит немало интересной магии, с которой мы познакомим тебя в ближайшее время.

Так что же происходит на самом деле в то время, когда мы с помощью React создаём пользовательский интерфейс? Самое время узнать, так что — к делу! 💥👊🕵️

React, Virtual DOM и reconciliation

React — узкоспециализированный инструмент в виде библиотеки для создания пользовательских интерфейсов. React помогает выполнить одну задачу (строить пользовательский интерфейс) и делает это хорошо, словно скальпель 🔪 в арсенале хирурга. 👨‍⚕️

Говоря о React, мы не имеем в виду анимации 📽️ или навигацию в браузере, которую также называют «роутингом» — переходом от одной страницы к другой в рамках одного веб-приложения. Для подобных задач созданы отдельные библиотеки, о которых мы поговорим в последующих частях этого конспекта.

🔎 Хозяйке на заметку:

> В прошлых версиях React включал в себя инструменты для анимации, валидации типов данных, роутинга и даже тестирования. А также набор вспомогательных инструментов, например «миксинов», которые применялись как «плагины» или так называемые «расширители функциональных возможностей».
>
> Но миксины уже не применяются и считаются устаревшими.

> А сама библиотека React сильно «похудела»: все вспомогательные техники для анимации, роутинга и прочего вынесены в отдельные npm-пакеты, чтобы акцентировать узконаправленную концепцию React — «делай что-то одно, но делай это в совершенстве». 💎

За счёт чего React позволяет создать высокопроизводительный UI? 🧐 Почему React быстрый? 🏎 Ответ заключается во внутреннем механизме работы React и структуре данных, лежащей в основе этого механизма.

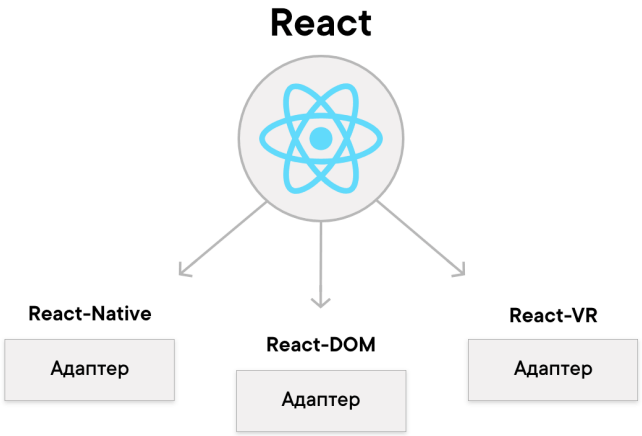
🔎 Хозяйке на заметку:

> UI — пользовательский интерфейс (от англ. User Interface).

Структура данных, используемая внутри React называется `Virtual DOM`, а механизм — `reconciliation`.

`Virtual DOM` или `VDOM` называют концепцию, когда желаемая модель представления UI в актуальном виде хранится в памяти и синхронизируется с `настоящей` моделью (то есть, DOM браузера) библиотекой `ReactDOM`. Этот процесс синхронизации и есть механизм, именуемый `reconciliation`.

React — универсальный инструмент, а алгоритм `reconciliation` (в переводе «согласование»), делающий React быстрым, является основой для его использования в других окружениях. Для использования React в окружении браузера существует адаптер — `React-DOM`. А, например, для постройки UI в окружении мобильных устройств используют адаптер `React-Native`. Адаптер под любое окружение использует React и его механику как основную зависимость.



Дело в том, что императивные операции над настоящим DOM слишком многословны и неэффективны в масштабных веб-приложениях. React берёт на себя работу с DOM, открывая декларативный API для комфортной разработки. 👨‍🎨

🔎 Хозяйке на заметку:

> В программировании, «императивный» подход — это когда ты детально и пошагово рассказываешь программе `как ей работать`. «Декларативный» подход — это когда ты говоришь программе только `что она должна делать`. То есть, «императивный» подход отвечает на вопрос «Как сделать?», а декларативный подход отвечает на вопрос «Что сделать?».

Иными словами, `Virtual DOM` — абстракция вокруг настоящего DOM браузера. `Virtual DOM` содержит описание того, что мы видим на странице в настоящее время. Распоряжаясь `Virtual DOM`, React запускает механизм `reconciliation` для сравнения предыдущей модели UI с новой моделью UI и определяет то, что изменилось (делает «дифф»).

`Virtual DOM` всегда держит ссылку на объект с предыдущей виртуальной моделью DOM, перед созданием новой модели. Затем, основываясь на результате сравнения меняет в настоящем DOM лишь то, что необходимо, а не весь DOM целиком. А если результат сравнения говорит о том, что изменений в VDOM не произошло (предыдущая и новая модели состояния UI — идентичны), то не производится никаких операций. 🎉

Цикл повторяется при каждом изменении состояния приложения.

В этом процессе участвуют следующие «ключевые игроки» VDOM:

- `ReactElement`;
- `ReactComponent`.

ReactElement

Структура VDOM в основе состоит из неких уникальных сущностей, именуемых `ReactElement`. `ReactElement` — основной тип элемента в React. Можешь думать о `ReactElement`, как о наименьшем строительном блоке приложения, написанного на React.

`ReactElement` описывает то, что ты видишь на экране.

🖥️ Пример кода 1.1:

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

const linkToGoogle = React.createElement(
    'a',
    {
        href: 'https://www.google.com'
    },
    'Google!'
);

ReactDOM.render(linkToGoogle, document.getElementById('app'))
```

Такая инструкция создаст элемент `a` (ссылка) в VDOM, с атрибутом `href`, ведущий на сайт Google. Напоминаем, что в VDOM создается лишь `модель`, а затем — эта `модель` «зеркальным отражением» отрисуется в настоящем DOM в виде HTML:

🖥️ Пример кода 1.2:

```
<a href="https://www.google.com">Google!</a>
```

В сравнении с DOM-элементами, React-элементы — простые JavaScript-объекты. React-элементы — «не дорогие» и не требуют большого количества системных ресурсов при создании, что делает их эффективными в контексте производительности. 🔥

`VDOM` и `reconciliation` позаботятся о своевременном и эффективном обновлении настоящего DOM, основываясь на имеющихся React-элементах. `ReactElement` — это легковесная, иммутабельная виртуальная модель DOM-элемента, не имеющая состояния.

ReactComponent

Теоретически, для постройки простого React-приложения достаточно React-элементов. Но полностью раскрыть потенциал React можно с использованием `ReactComponent` (далее — компонент). Компонент позволяет создать инкапсулированную сущность со встроенным состоянием и методами жизненного цикла. Компоненты также имеют механизмы передачи данных через «пропсы» (props), и «контекст» (context). Эти механизмы мы рассмотрим в следующих частях конспекта.

Компоненты бывают следующих типов:

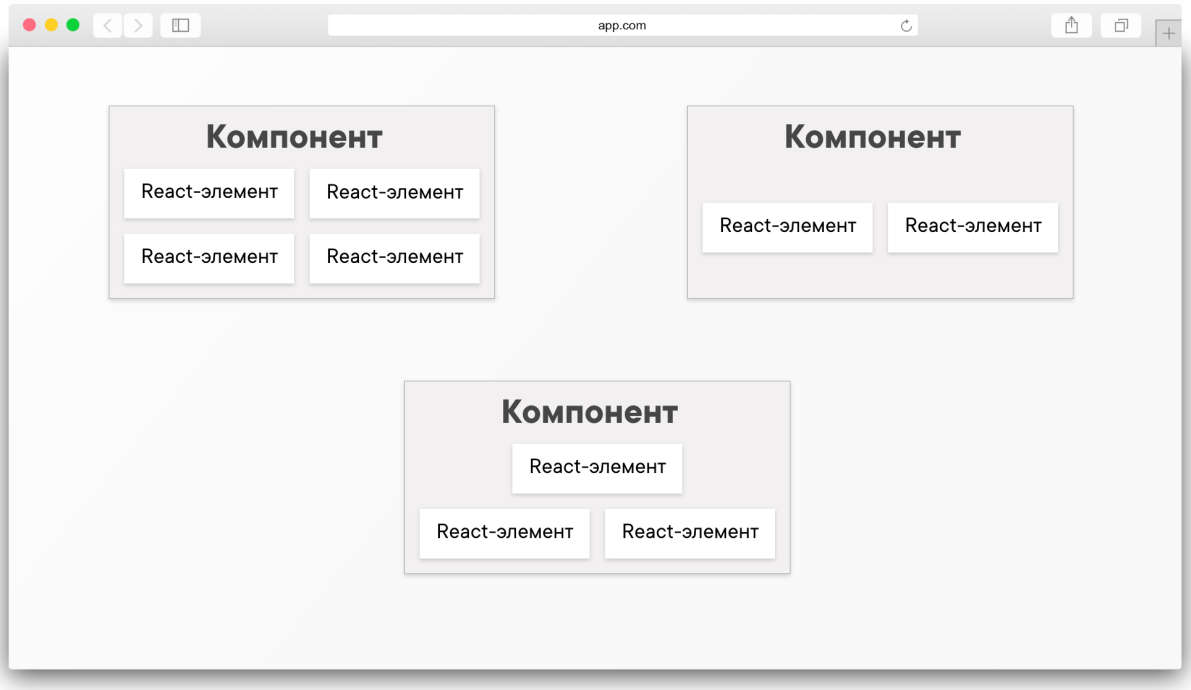
- `Классовые компоненты`: создаются с помощью ES2015-классов;
- `Функциональные компоненты`: создаются с помощью обычных функций.

Совет бывалых:

> На данный момент классовые компоненты обладают более широким спектром возможностей. Функциональные компоненты обладают преимуществом лишь в виде своей компактности. Поэтому, несмотря на возможность использовать любой тип компонента, рекомендую использовать классовые.

Можешь думать о компоненте React как о грузовике, везущем кирпичи (строительные блоки), где каждый строительный блок — React-элемент.

Схематически, отношение компонентов и React-элементов в рамках приложения можно описать так:



Компоненты — сердце и душа React, но об этом чуть позже. 👰🙌🏼

Fiber

`Fiber` — новый движок `reconciliation`, представленный в React v16. Главная цель `Fiber` — позволить React осуществлять инкрементальный, запланированный рендер виртуального DOM, что значительно повысит производительность и отзывчивость UI. На данный момент файбер находится в стадии разработки, и доступен к использованию пока еще как экспериментальная фича. 🚧 Полноценный релиз `Fiber` ожидается в ближайших версиях React v16.x или React v17. 🚀

Подведём итоги

Мы рассмотрели много новой и базовой информации о React. Эти знания позволят нам продвинуться к более глубоким концептам и ощутить магию React в полной мере. 🔮

На данный момент мы знаем, что React умеет эффективно 🚀 обновлять UI за счёт:

- `Virtual DOM` — виртуальной модели настоящего DOM, в виде огромного JavaScript-объекта.
- `reconciliation` — механизма, работающего в паре с `Virtual DOM`. Когда состояние React-приложения меняется, этот механизм проходит по виртуальному дереву, определяя какие конкретно его части изменились. С диффом, полученным в результате, React обновляет настоящий DOM браузера (минимальное количество DOM-узлов, необходимое для того, чтобы отразить изменения `Virtual DOM`).

`Virtual DOM` состоит из следующих сущностей: 👷

- `ReactElement` — элементарный строительный блок. По своей природе React-

элемент — обычный иммутабельный JavaScript-объект, чаще всего описывающий будущий HTML-элемент. React-элемент легковесный и производительный.
- `ReactComponent` — абстрактная сущность-контейнер для `ReactElement`. Обладает инкапсулированным состоянием, методами жизненного цикла, механизмом передачи данных, а также еще несколькими интересными особенностями, которые мы рассмотрим далее.

В следующей части конспекта мы рассмотрим `JSX` — синтаксис, позволяющий создавать элементы и компоненты React лаконичным, удобным и красивым путем. 🎈 Увидимся в следующей части! 🤺

> Мы будем очень признательны, если ты оставишь свой фидбек в отношении этой части конспекта на нашу электропочту `hello@lectrum.io`.