ТЕХНОАТОМ

Лекция 5

Библиотеки. Фреймворки. React

Мартин Комитски

План на сегодня

ТЕХНОАТОМ

- Введение
- Теория
  - Библиотека
  - Фреймворк
  - Инструменты
  - Введение в React
  - React Компоненты
  - JSX
- Практика
  - Используемые инструменты
  - Разворачиваем свое окружение для разработки React приложений
- Теория & Практика
  - React basics (Разбор примеров по React (Компоненты и все остальное))
  - Context
  - Portals
  - Refs
  - Web Components
  - Prop Types
  - Hooks
- SSR
- React Native

Минутка бюрократии

ТЕХНОАТОМ

- Внимание
- Отметки о посещении занятий
- Обратная связь о лекциях

ТЕХНОАТОМ

Библиотеки

Библиотеки

ТЕХНОАТОМ

Библиотека — *это структурированный набор полезного функционала.*

Содержит функции для работы со следующими вещами:

- Строки
- Даты
- DOM-элементы
- События
- Cookie
- Анимации
- Запросы
- Многое другое

Библиотеки

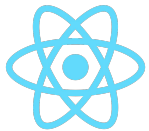
ТЕХНОАТОМ

Плюсы:

- Увеличивает скорость разработки
- Снижает порог входа в проект (и сами технологии)
- Кроссбраузерность

Минусы:

- Могут быть ошибки
- Быстрое устаревание/обновление



ТЕХНОАТОМ

Фреймворки

Фреймворки

ТЕХНОАТОМ

Фреймворк — *это каркас приложения*. Он обязывает разработчика выстраивать архитектуру приложения в соответствии с некоторой логикой.

Предоставляет функционал для:

- Событий
- Хранилищ
- Связывания данных

Плюсы:

- Быстрый старт
- Структурированный код, использование паттернов
- Много нужных инструментов идёт уже “в коробке”

Минусы:

- У каждого свои “идеология” и подход
- Надо учить

Фреймворки

Еще плюсы:

- Более высокий уровень абстракции
- Можно построить около 80% вашего приложения

Еще минусы:

- Немалые трудности из-за ограничений
- Быстрое устаревание/обновление

ТЕХНОАТОМ

Инструменты

Инструменты

ТЕХНОАТОМ

Инструмент — *это вспомогательное средство разработки, но он не является неотъемлемой частью проекта.*

Виды:

- системы сборки
- компиляторы
- транспайлеры
- механизмы развертывания
- препроцессоры
- линтеры
- тесты
- многое другое

Инструменты

ТЕХНОАТОМ

Плюсы:

- Невероятное упрощение работы
- Контроль разных этапов разработки
- Автоматизация

Минусы:

- Их много
- Надо разбираться
- Надо уметь настраивать

Введение в React. Термины

ТЕХНОАТОМ

React – *это декларативная, эффективная и гибкая JavaScript библиотека для разработки интерфейсов*. Она позволяет составлять сложные визуальные интерфейсы из атомарных кусочков, называемых “компонентами”.

Императивный – *как сделать*.
Декларативный – *что сделать*.

Появился в 2011 в Facebook, в ленте Instagram в 2012.

Virtual DOM - JSON, который описывает обычный DOM, делает diff в RAM
JSX - ...

Введение в React. Компоненты. JSX

ТЕХНОАТОМ

```
class ShoppingList extends React.Component {
  render() {
    return (
      <div className="shopping-list">
        <h1>Shopping List for {this.props.name}</h1>
        <ul>
          <li>Instagram</li>
          <li>WhatsApp</li>
          <li>Oculus</li>
        </ul>
      </div>
    );
  }
}

// Example usage: <ShoppingList name="Mark" />
```

Введение в React. Понятие JSX


```
const element = <h1>Hello, world!</h1>;
```

JSX - ни строка, ни HTML.
JSX - представляет собой расширение **J**ava**S**cript при помощи **X**ML синтаксиса.
JSX - нормальное, полноценное JS выражение.

```
const name = 'Martin Komitsky';
const element = <h1>Hello, { name }</h1>;

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Hello, Martin Komitsky

Введение в React. Понятие JSX. Выражения

ТЕХНОАТОМ

JSX может содержать и выполнять JS выражение внутри “{ }”.

```
function formatName(user) {
  return user.firstName + ' ' + user.lastName;
}

const user = {
  firstName: 'Martin',
  lastName: 'Komitsky'
};

const element = (
  <h1>
    Hello, Sir { formatName(user) }!
  </h1>
);

ReactDOM.render(
  element,
  document.getElementById('root')
);
```

Hello, Sir Martin Komitsky!

Введение в React. Понятие JSX. Дочерние эл., атр.

ТЕХНОАТОМ

JSX может содержать многострочные конструкции, тогда оборачивается в ( ).

```
const element = (
  <div>
    <h1>Hello!</h1>
    <h2>Good to see you here.</h2>
  </div>
);
```

В **JSX** можно выражениях выставлять любые **HTML** атрибуты, но в *camelCase* (кроме class – там будет className).

```
const element = <img src={ user.avatarUrl }></img>;
```

JSX защищает от внедрения опасного кода.

```
const title = response.xss;
// This is safe:
const element = <h1>{ title }</h1>;
```

Вопросы?

Введение в React. Используемые инструменты

ТЕХНОАТОМ

- React DevTools
- Webpack
- Babel
- Create React App

Довольно! Перемещаемся в текстовый редактор.

Все примеры в репозитории.

ТЕХНОАТОМ

Практическая часть
~ 10 мин

ТЕХНОАТОМ

Разбор примеров ч. 1

Перерыв! (10 минут)

Препод (с)


TEXHOATOM
React. Context

React. Context

ТЕХНОАТОМ

Контекст позволяет передавать данные через дерево компонентов без необходимости передавать props на промежуточных уровнях.

```
1.  // data provider
2.  const MyContext = React.createContext(defaultValue);
3.
4.  <MyContext.Provider value={‘test’}>
5.
6.  // consumer 1
7.  MyClass.contextType = MyContext;
8.  const value = this.context;
9.
10. // consumer 2
11. <MyContext.Consumer>
12.   {value => <div>{value}</div>} // выведет test
13. </MyContext.Consumer>
14.
```

TEXHOATOM

React. Portals

React. Portals

ТЕХНОАТОМ

Порталы позволяют рендерить дочерние элементы в DOM-узел, который находится вне DOM-иерархии родительского компонента.

Типовой случай применения порталов — когда в родительском компоненте заданы стили overflow: hidden или z-index, но вам нужно чтобы дочерний элемент визуально выходил за рамки своего контейнера. Например, диалоги, всплывающие карточки и всплывающие подсказки.

```
render() {
  // React *не* создаёт новый div. Он рендерит дочерние элементы в `domNode`.
  // `domNode` — это любой валидный DOM-узел, находящийся в любом месте в DOM.
  return ReactDOM.createPortal(
    this.props.children,
    domNode
  );
}

```

TEXHOATOM

React. Refs

React. Refs

ТЕХНОАТОМ

Рефы дают возможность получить доступ к DOM-узлам или React-элементам, созданным в рендер-методе.

```
1.  // Создание ссылки
2.  class MyComponent extends React.Component {
3.    constructor(props) {
4.      super(props);
5.      this.myRef = React.createRef();
6.    }
7.    render() {
8.      return <div ref={this.myRef} />;
9.    }
10. }
11.
12. // Обращение к ссылке
13. const node = this.myRef.current;
14.
```

React. Refs. Когда использовать Ref

ТЕХНОАТОМ

Ситуации, в которых использования рефов является оправданным:

- Управление фокусом, выделение текста или воспроизведение медиа.
- Императивный вызов анимаций.
- Интеграция со сторонними DOM-библиотеками.

Не злоупотребляйте рефами. Чаще всего можно обойтись обычным способом.

TEXHOATOM

React. Web Components

React. Web Components

React и веб-компоненты созданы для решения самых разных задач. Веб-компоненты обеспечивают надёжную инкапсуляцию для повторно используемых компонентов, в то время как React предоставляет декларативную библиотеку для синхронизации данных с DOM. Две цели дополняют друг друга. Как разработчик, вы можете использовать React в своих веб-компонентах, или использовать веб-компоненты в React, или и то, и другое.

React. Web Components

ТЕХНОАТОМ

```
1.  // Использование веб-компонентов в React
2.  class HelloMessage extends React.Component {
3.    render() {
4.      return <div>Привет, <x-search>{this.props.name}</x-search>!</div>;
5.    }
6.  }
7.  // Использование React в веб-компонентах
8.  class XSearch extends HTMLElement {
9.    connectedCallback() {
10.     const mountPoint = document.createElement('span');
11.     this.attachShadow({ mode: 'open' }).appendChild(mountPoint);
12.
13.     const name = this.getAttribute('name');
14.     const url = 'https://www.google.com/search?q=' + encodeURIComponent(name);
15.     ReactDOM.render(<a href={url}>{name}</a>, mountPoint);
16.   }
17. }
18. customElements.define('x-search', XSearch);
```

TEXHOATOM

React. Prop Types

React. Prop Types

По мере роста вашего приложения вы можете отловить много ошибок с помощью проверки типов. Для этого можно использовать расширения JavaScript вроде Flow и TypeScript. Но, даже если вы ими не пользуетесь, React предоставляет встроенные возможности для проверки типов. Для запуска этой проверки на свойствах компонента вам нужно использовать специальное свойство propTypes.

PropTypes предоставляет ряд валидаторов, которые могут использоваться для проверки, что получаемые данные корректны. В примере мы использовали PropTypes.string. Когда какой-то prop имеет некорректное значение, в консоли будет выведено предупреждение. По соображениям производительности propTypes проверяются только в режиме **разработки**.

React. Prop Types

ТЕХНОАТОМ

```
import PropTypes from 'prop-types';

class Greeting extends React.Component {
  render() {
    return (
      <h1>Привет, {this.props.name}</h1>
    );
  }
}

Greeting.propTypes = {
  name: PropTypes.string
};

```

TEXHOATOM

React. Hooks

React. Hooks

ТЕХНОАТОМ

Хуки — нововведение в React 16.8, которое позволяет использовать состояние и другие возможности React без написания классов.

```
1.  import React, { useState } from 'react';
2.
3.  function Example() {
4.    // Объявление переменной состояния, которую мы назовём "count"
5.    const [count, setCount] = useState(0);
6.
7.    return (
8.      <div>
9.        <p>Вы кликнули {count} раз</p>
10.       <button onClick={() => setCount(count + 1)}>
11.         Нажми на меня
12.       </button>
13.     </div>
14.   );
15. }
16.
```

React. Hooks

Хуки — это функции JavaScript, которые налагают два дополнительных правила:

- Хуки следует вызывать только на верхнем уровне. Не вызывайте хуки внутри циклов, условий или вложенных функций.
- Хуки следует вызывать только из функциональных компонентов React. Не вызывайте хуки из обычных JavaScript-функций. Есть только одно исключение, откуда можно вызывать хуки — это ваши пользовательские хуки.

React. Hooks

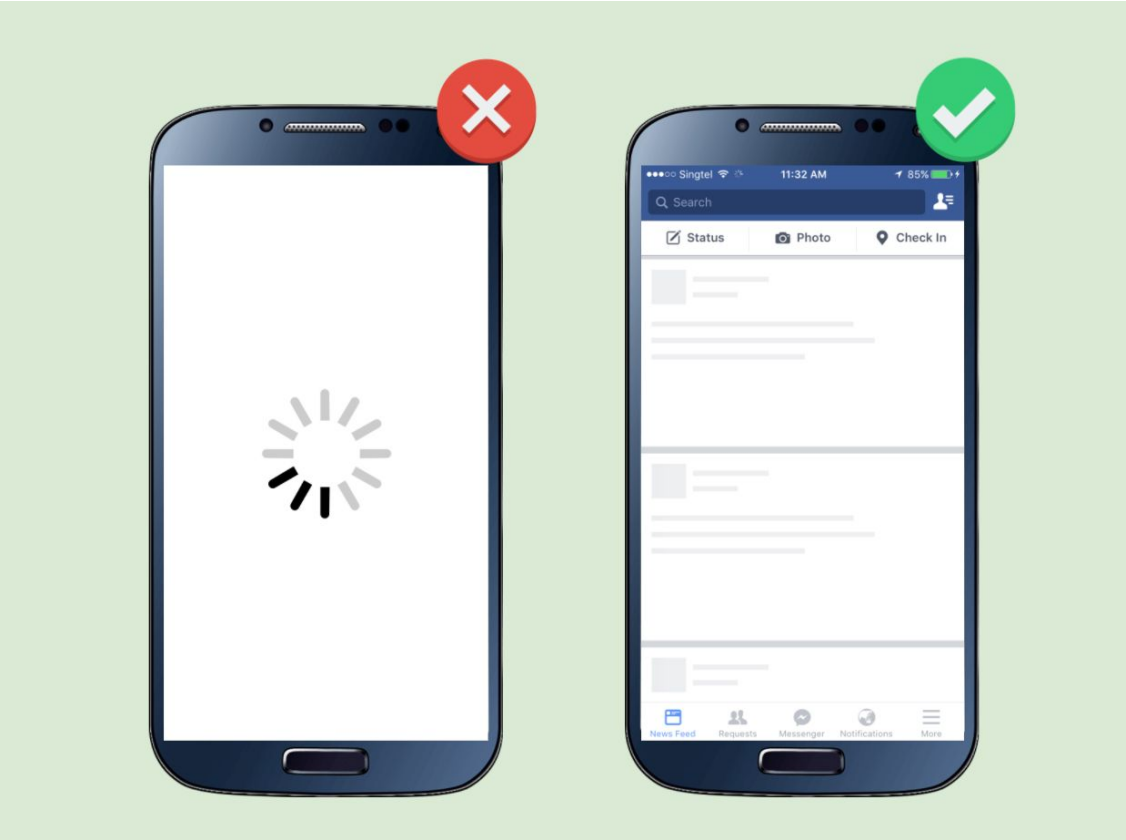
ТЕХНОАТОМ

- Основные хуки
  - useState
  - useEffect
  - useContext
- Дополнительные хуки
  - useReducer
  - useCallback
  - useMemo
  - useRef
  - useImperativeHandle
  - useLayoutEffect
  - useDebugValue

TEXHOATOM

React. SSR

React. SSR

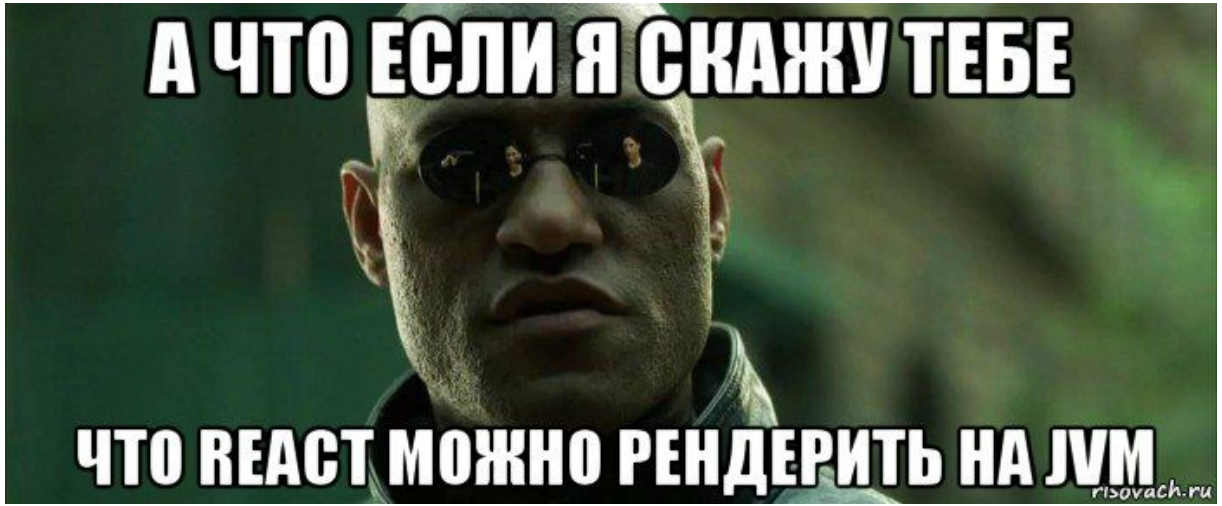

Singtel
11:32 AM
85%
Search
Status
Photo
Check In
News Feed
Requests
Messenger
Notifications
More

React. SSR

ТЕХНОАТОМ

Server-Side Rendering - возможность *Frontend* фреймворка отрисовывать *HTML* разметку, работая через системы *Backend*.

SSR + SPA = *Универсальное приложение* (работает как на *front*, так и на *back*). Можно встретить под названием “изоморфические приложения”.



React. SSR. Настройка

ТЕХНОАТОМ

1. https://reactjs.org/docs/react-dom-server.html
2. https://medium.freecodecamp.org/demystifying-reacts-server-side-render-de335d408fe4
3. https://flaviocopes.com/react-server-side-rendering/
4. https://nextjs.org/features/server-side-rendering

TEXHOATOM

React. React Native

React. React Native

React-native позволяет разрабатывать *нативные* мобильные приложения на Android и iOS при помощи javascript и React.

```
1.  import React, { Component } from 'react';
2.  import { Text, View } from 'react-native';
3.
4.  class HelloReactNative extends Component {
5.    render() {
6.      return (
7.        <View>
8.          <Text>
9.            If you like React, you'll also like React Native.
10.         </Text>
11.         <Text>
12.           Instead of 'div' and 'span', you'll use native components
13.           like 'View' and 'Text'.
14.         </Text>
15.       </View>
16.     );
17.   }
18. }
19.
```

React. Полезные ссылки

ТЕХНОАТОМ

- https://reactpatterns.com/
- https://www.hooks.guide/
- https://reactjs.org/docs/hooks-faq.html
- Просто про React Context
- React Native
- Web components in React
- SSR
- Еще про SSR

Домашнее задание № 4

ТЕХНОАТОМ

1. Закрепить знания React.js
2. Изучить Create React App
3. Сгенерировать проект при помощи generator-track-mail
4. Переписать компоненты на React
5. Восстановить стили для всех компонентов
6. Восстановить функциональность всех компонентов

Срок сдачи:

12 ноября

ТЕХНОАТОМ

Спасибо за внимание!