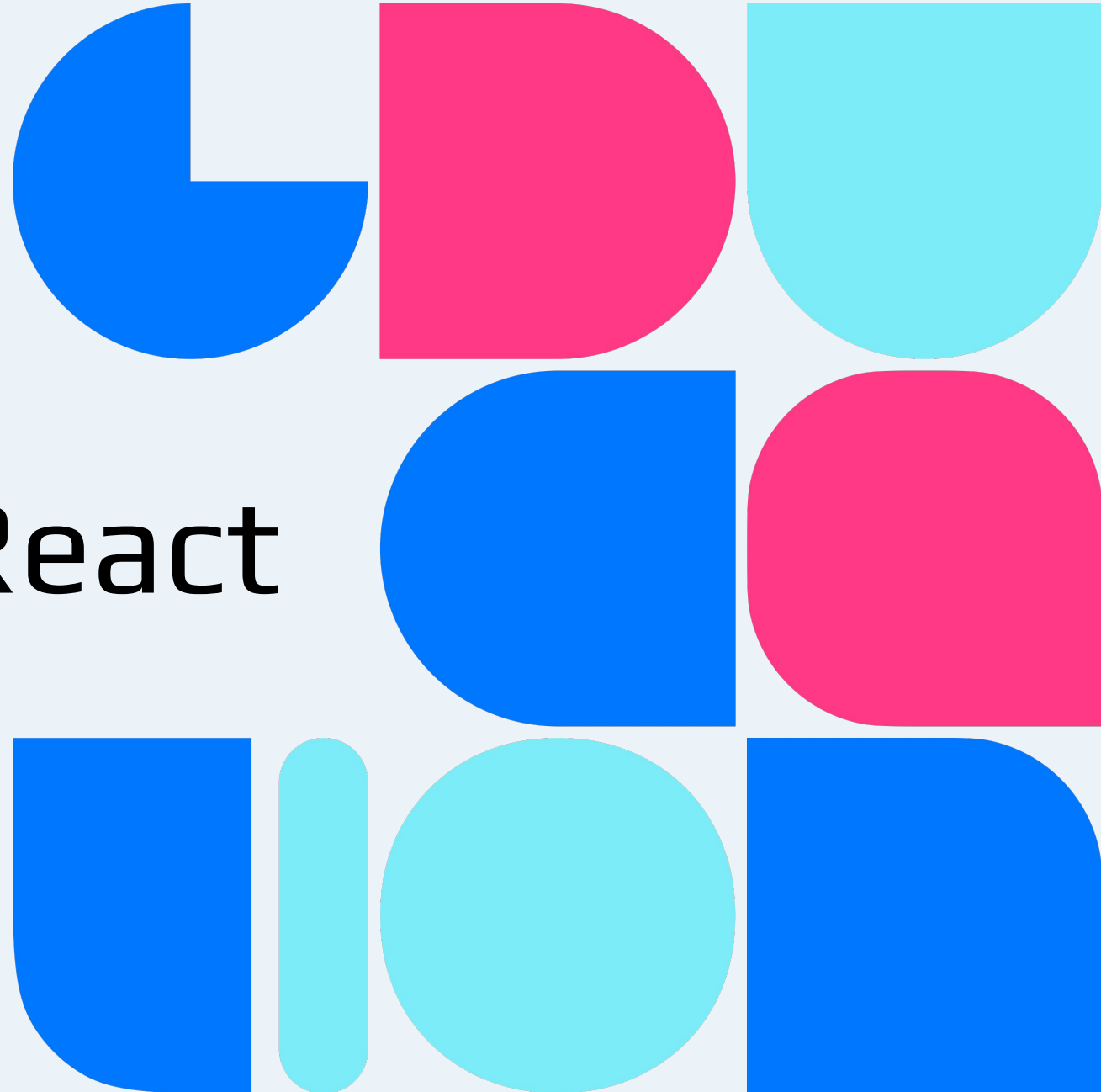
education

Лекция 5 Библиотеки. Фреймворки. React

Мартин Комитски



План на сегодня

- Введение
- Теория
  - Библиотека
  - Фреймворк
  - Инструменты
  - Введение в React
  - React Компоненты
  - JSX
- Практика
  - Используемые инструменты
  - Разворачиваем свое окружение для разработки React приложений
  - Квиз
- 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

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

Библиотеки

Библиотека — *это структурированный набор полезного функционала.*

Содержит функции для работы со следующими вещами:

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

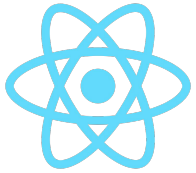
Библиотеки

Плюсы:

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

Минусы:

- Могут быть ошибки
- Быстрое устаревание/обновление



Библиотеки

Фреймворки

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

Фреймворки


SVELTE
ANGULAR
REACT
VUE
JQUERY BEING USED ON 70%
OF SITES IN 2022

Инструменты

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

Плюсы:

- Невероятное упрощение работы
- Контроль разных этапов разработки
- Автоматизация

Минусы:

- Их много
- Надо разбираться
- Надо уметь настраивать

Введение в React. Термины

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, приказывать по шагам*.
Декларативный – *что сделать, описывая проблему и ожидаемый результат*.

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

```
const ShoppingList = ({ name }) => {
 return (
   <div className="shopping-list">
     <h1>ShoppingList for {name}</h1>
     <ul>
       <li>Instagram</li>
       <li>WhatsApp</li>
       <li>Oculus</li>
     </ul>
   </div>
 );
};

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

Введение в React. Понятие JSX

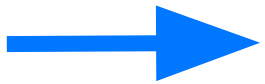
```
const element = <h1>Hello World!</h1>;
```

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

```
import { createRoot } from 'react-dom';

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

const root = createRoot(
  document.getElementById('root')
);
root.render(element);
```



Hello, Martin Komitsky

Введение в React. Понятие JSX. Выражения

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

```
import { createRoot } from 'react-dom';

const formatName = (user) => {
 return `${user.firstName} ${user.lastName}`;
};

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

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

const root = createRoot(
  document.getElementById('root')
);
root.render(element);
```

Hello, Sir Martin Komitsky!

Введение в React. Понятие JSX. Дочерние элементы, атрибуты

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} />
```

JSX защищает от внедрения опасного кода.

```
const title = response.xss;
// Это безопасно!
const element = <h1>{title}</h1>;
```

Введение в React. Компоненты. JSX

Вопросы?

Введение в React. Используемые инструменты

- React DevTools
- Webpack
- Babel
- Create React App

Или современный Create Vite App

Введение в React. Отрисовка элементов

Довольно! Перемещаемся в текстовый редактор.

Все примеры в репозитории.

npm create vite@latest

Практическая часть
~ 0 сек

Квиз

https://forms.gle/2wrK9Ti2SGejLF6M9

Разбор примеров

Перерыв! (10 минут)

Препод (с)

React. Context

React. Context

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

```
import { createContext } from 'react';

export const MyContext = createContext(defaultDataValue);

const MyParentComponent = () => {
 const myData = 'hello!';

 return (
   <MyContext.Provider value={myData}>
     ...
     <MyChildComponent />
     ...
   </MyContext.Provider>
 );
};
```

```
import { MyContext } from 'MyParentComponent.jsx'

const MyChildComponent = () => {
 const myData = useContext(MyContext);

 return (
   <div>{myData}</div>
 );
};
```

React. Portals

React. Portals

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.

```
import { createPortal } from 'react-dom';

// ...

<div>
 <p>Размещается стандартно, в родительском div</p>
 {createPortal(
   <p>А этот параграф расположится прямиком в body</p>,
   document.body
 )}
</div>
```

React. Refs

React. Refs

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

```
import { useRef } from 'react';

const MyComponent = () => {
 const wrapperRef = useRef(null);

 logElement () {
   if (wrapperRef.current) {
     // обращаемся к Hello!
     console.log(wrapperRef.current.innerText);
   }
 }

 return (
   <div ref={wrapperRef}>Hello!</div>
 );
};
```

React. Refs. Когда использовать Ref

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.

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

```
import { createRoot } from 'react-dom';

// Использование веб-компонентов в React
const HelloMessage = ({ name }) => {
 return <div>Привет, <x-search>{name}</x-search>!</div>;
};

// Использование React в веб-компонентах
class XSearch extends HTMLElement {
 connectedCallback() {
   const mountPoint = document.createElement('span');
   this.attachShadow({ mode: 'open' }).appendChild(mountPoint);

   const name = this.getAttribute('name');
   const url = 'https://www.google.com/search?q='+ encodeURIComponent(name);

   const root = createRoot(mountPoint);
   root.render(<a href={url}>{name}</a>);
 }
}

customElements.define('x-search', XSearch);
```

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

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

class Greeting extends React.Component {
 static propTypes = {
   name: PropTypes.string
 };

 render() {
   return (
     <h1>Привет, {this.props.name}</h1>
   );
 }
}
```

React. Hooks

React. Hooks

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

```
import { useState } from 'react';

const Example = () => {
 // Объявление переменной состояния, которую мы назовём "count"
 const [count, setCount] = useState(0);

 return (
   <div>
     <p>Вы кликнули {count} раз</p>
     <button onClick={() => setCount(count + 1)}>
       Нажми на меня
     </button>
   </div>
 );
};
```

React. Hooks

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

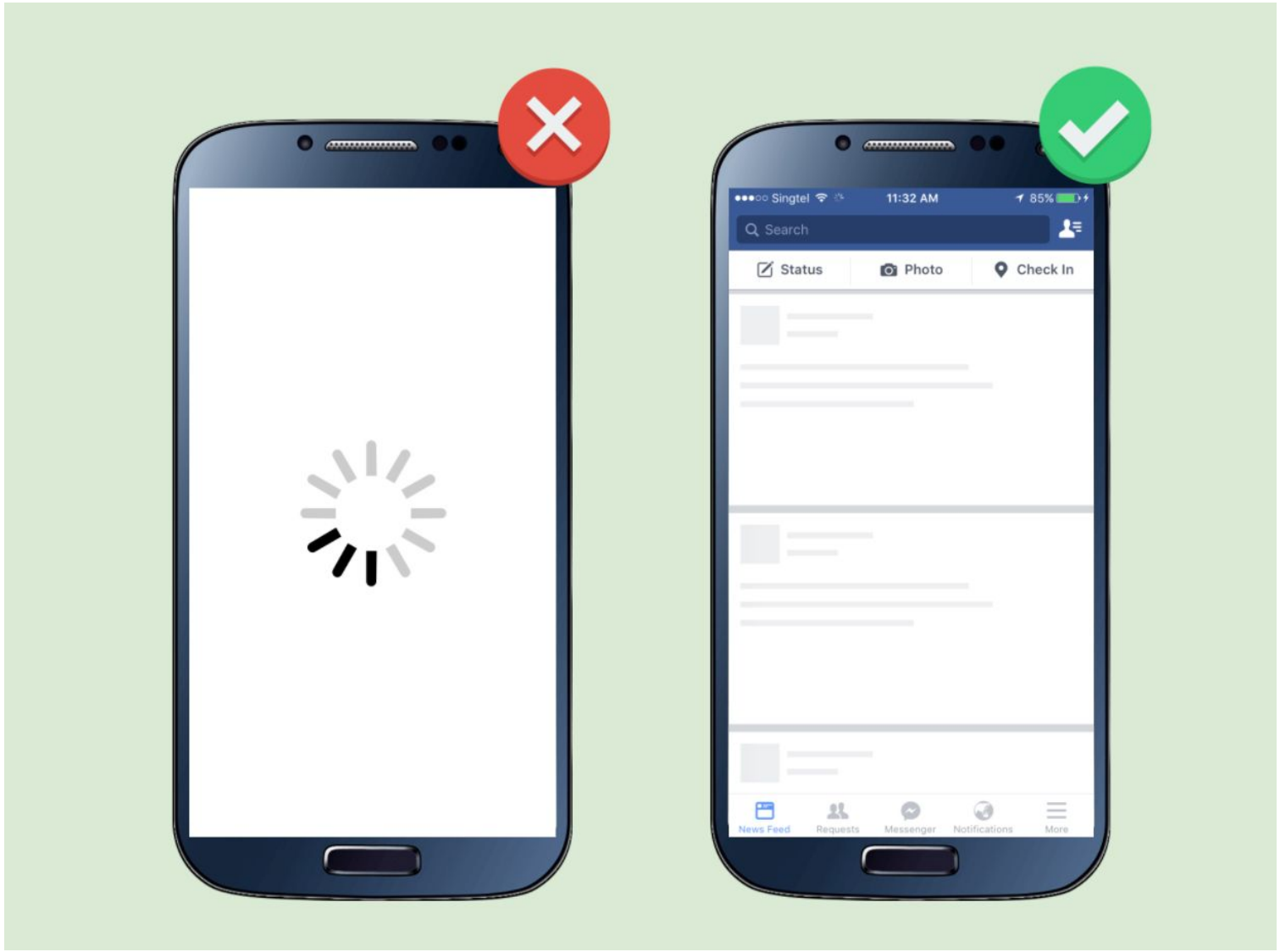
- Хуки следует вызывать только на верхнем уровне. Не вызывайте хуки внутри циклов, условий или вложенных функций.
- Хуки следует вызывать только из функциональных компонентов React. Не вызывайте хуки из обычных JavaScript-функций. Есть только одно исключение, откуда можно вызывать хуки — это ваши пользовательские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

React. Hooks

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

React. SSR

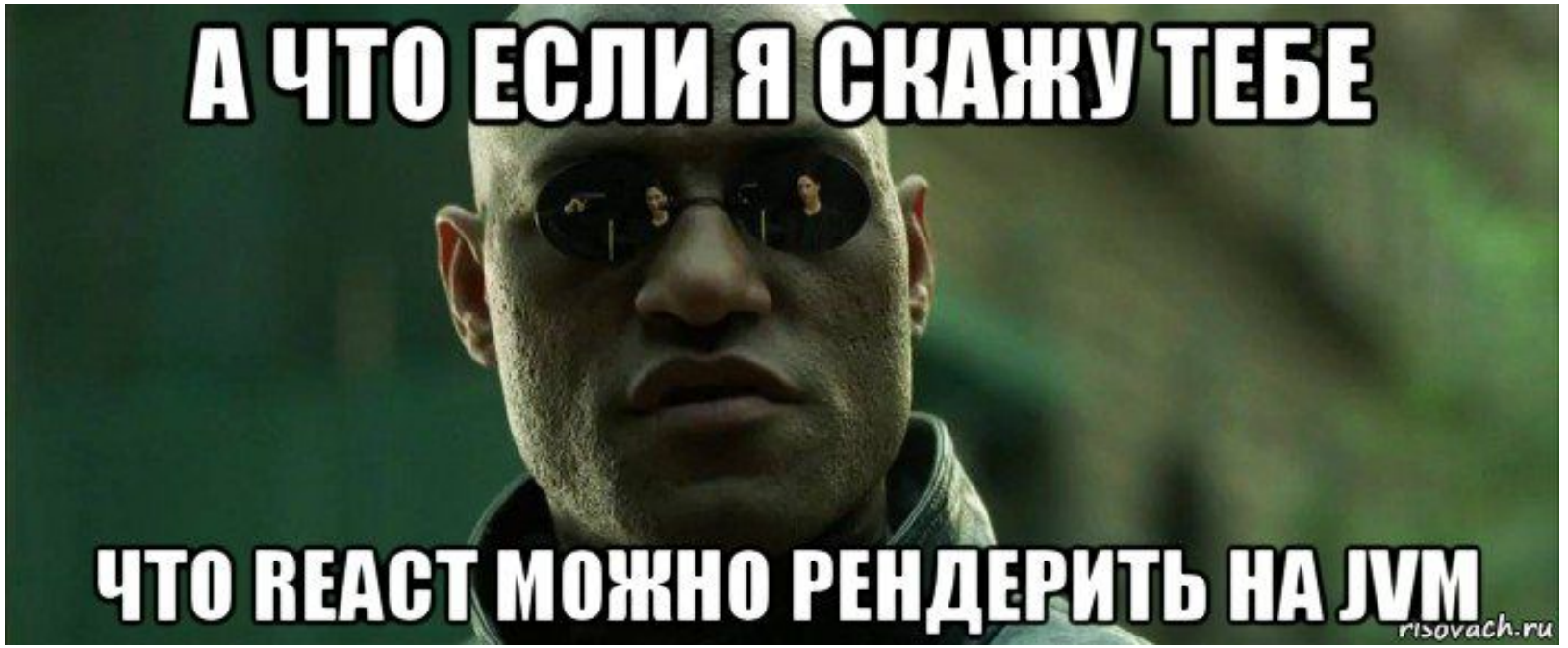
React. SSR


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

React. SSR

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

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

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

- 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 №5

1. Закрепить знания React.js
2. Изучить Create React App/Vite
3. Сгенерировать проект с помощью генератора
4. Переписать компоненты на React
5. Восстановить стили для всех компонентов
6. Восстановить функциональность всех компонентов

Расширенное описание задания, подсказки, а также презентации с лекций всегда есть в репозитории.

Срок сдачи

24 октября

Мем дня


Donald J. Trump @realDonaldTrump · 16h
@bunjavascript
I'll send you IMPORTANT ELECTION UPDATES for North Carolina. Make sure you are ready to VOTE FOR DONALD J. TRUMP by November 5th.
Reply #stop to opt-out.
2
4
11
1K
Bun
@bunjavascript
i am a javascript runtime
5:21 AM · Oct 3, 2024 · 3,262 Views
2
10
45
3

Спасибо за внимание!

education

Пока!

Присоединяйтесь к сообществу про образование в VK

- VK Образование

education