Лекция 6
React. Дополнительные возможности

Мартин Комитски

План на сегодня

- Context
- Portals
- Refs
- Web Components
- Prop Types
- Hooks
- SSR
- React Native

Минутка бюрократии

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

React. Context

React. Context

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.
```

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.

```
1.  render() {
2.    // React *не* создаёт новый div. Он рендерит дочерние элементы в `domNode`.
3.    // `domNode` — это любой валидный DOM-узел, находящийся в любом месте в DOM.
4.    return ReactDOM.createPortal(
5.      this.props.children,
6.      domNode
7.    );
8.  }
9.
```

React. Refs

React. Refs

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

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

```
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);
```

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 проп имеет некорректное значение, в консоли будет выведено предупреждение. По соображениям производительности propTypes проверяются только в режиме **разработки**.

React. Prop Types

```
1.  import PropTypes from 'prop-types';
2.
3.  class Greeting extends React.Component {
4.    render() {
5.      return (
6.        <h1>Привет, {this.props.name}</h1>
7.      );
8.    }
9.  }
10.
11. Greeting.propTypes = {
12.   name: PropTypes.string
13. };
14.
```

React?

Вопросы?

“Перерыв! (10 минут)

Препод (с)

React. Hooks

React. Hooks

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:

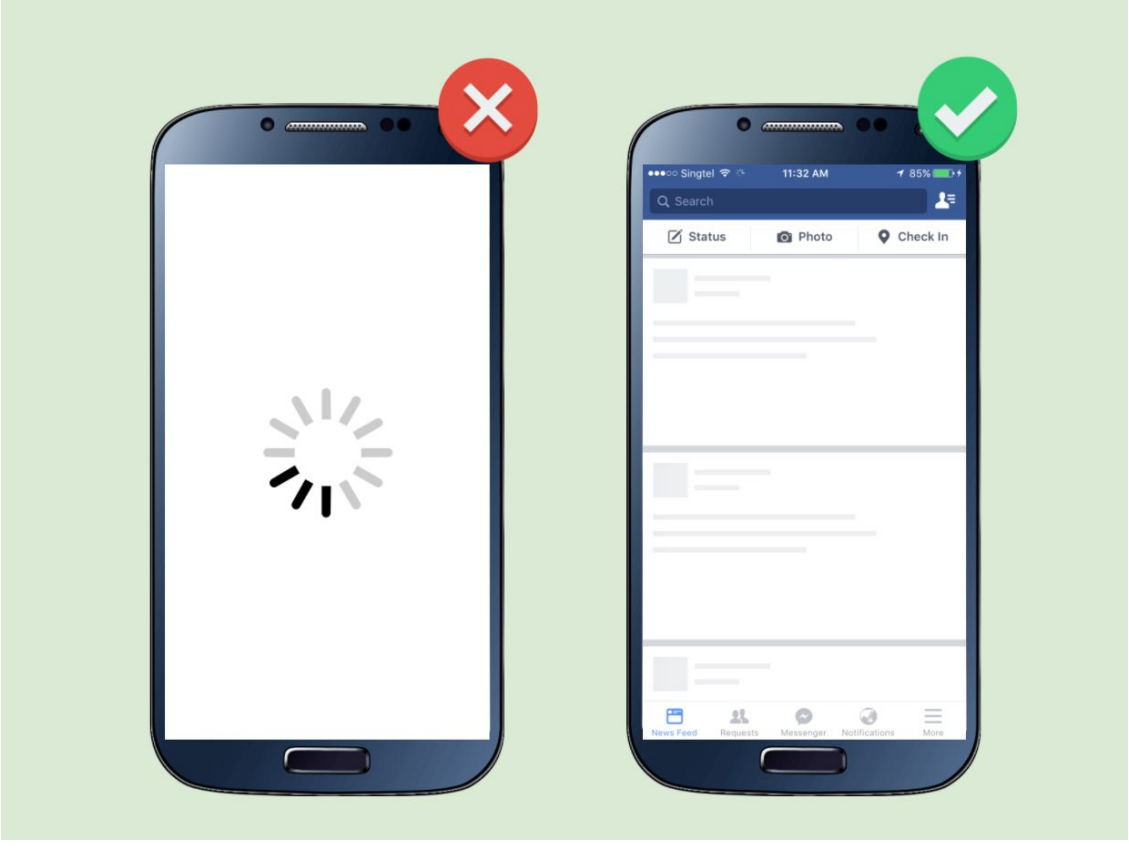
- Хуки следует вызывать только на верхнем уровне. Не вызывайте хуки внутри циклов, условий или вложенных функций.
- Хуки следует вызывать только из функциональных компонентов React. Не вызывайте хуки из обычных JavaScript-функций. Есть только одно исключение, откуда можно вызывать хуки — это ваши пользовательские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

React. Hooks

- 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”.

1.

React. SSR. Настройка

1. https://medium.freecodecamp.org/demystifying-reacts-server-side-render-de335d408fe4

React. React Native

React. React Native

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

```
import React, {Component} from 'react';
import {Text, View} from 'react-native';

class HelloReactNative extends Component {
  render() {
    return (
      <View>
        <Text>
          If you like React, you'll also like React Native.
        </Text>
        <Text>
          Instead of 'div' and 'span', you'll use native components
          like 'View' and 'Text'.
        </Text>
      </View>
    );
  }
}

```

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

Домашнее задание № 6

1.

Срок сдачи

FS-21: ? апреля

Спасибо за внимание!