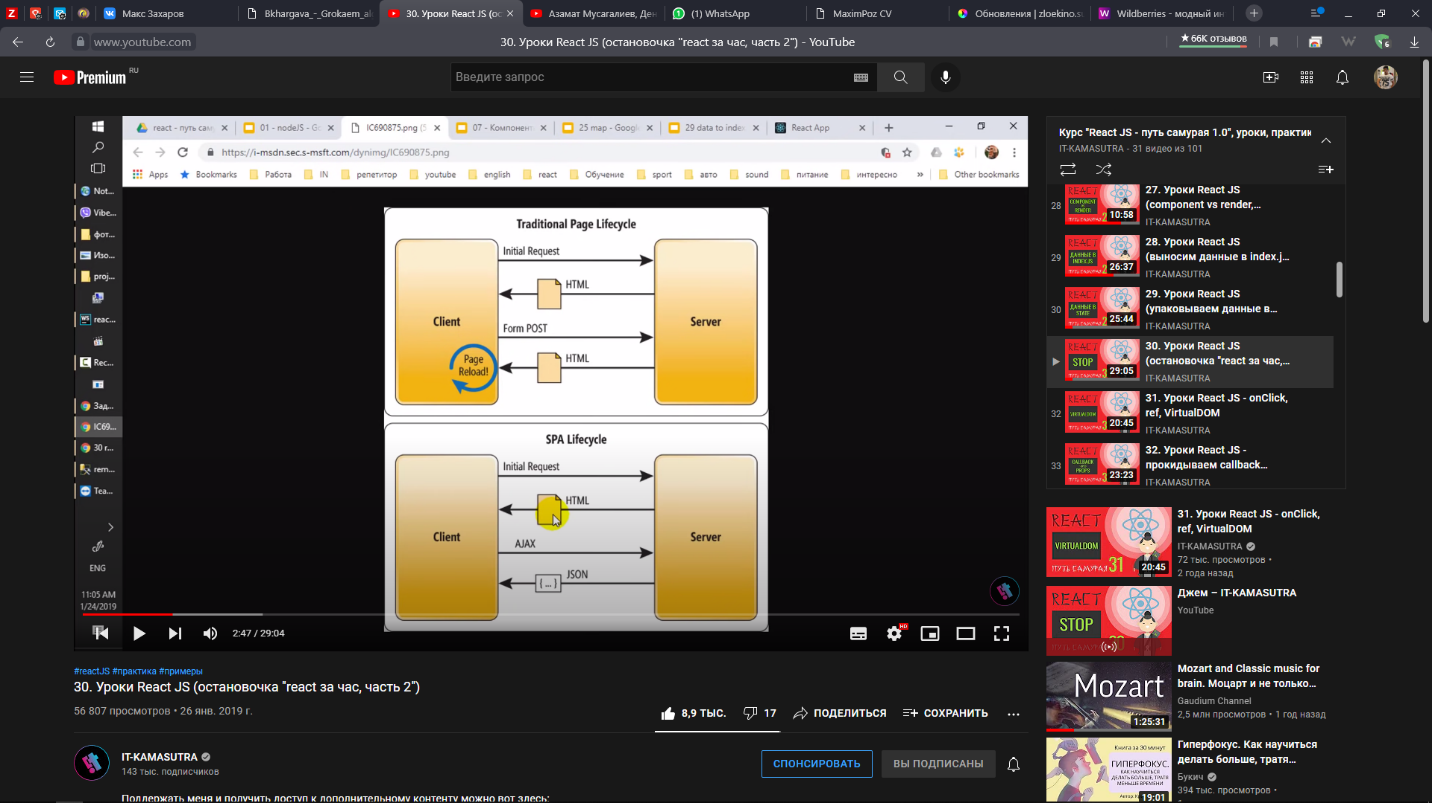
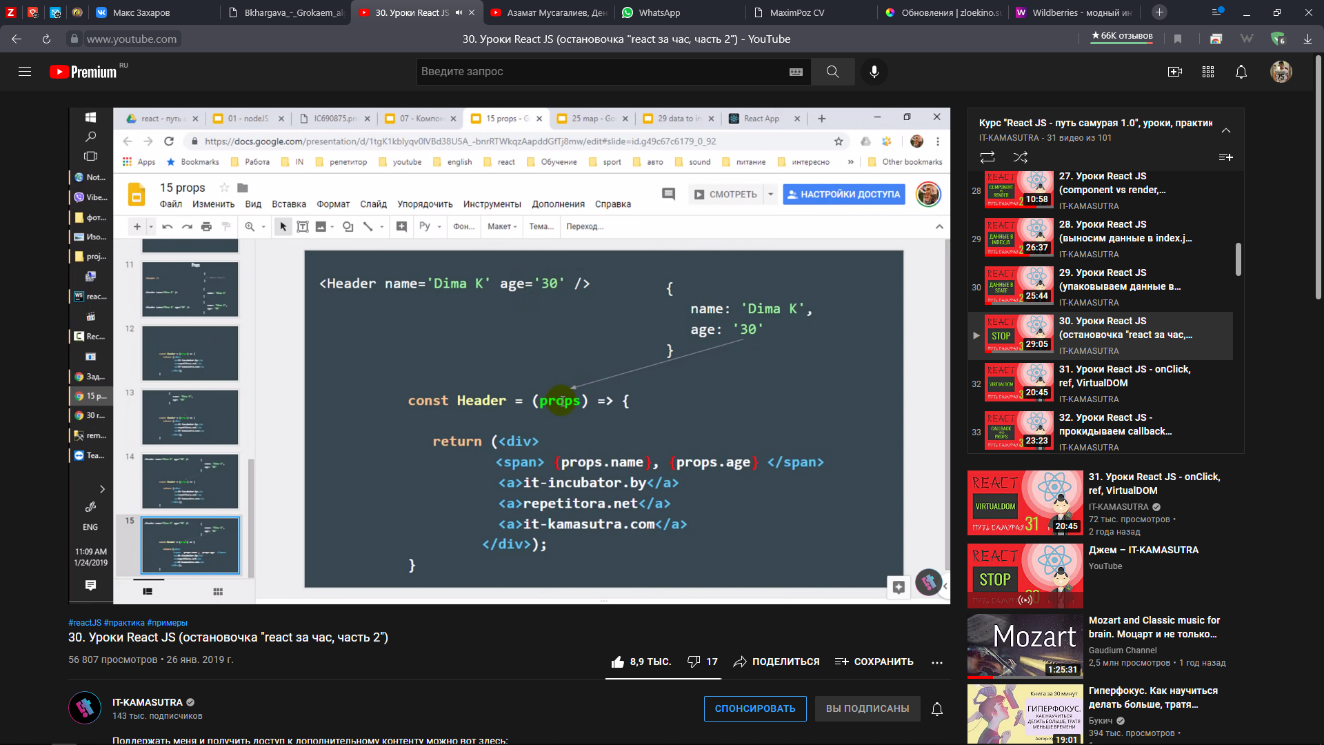
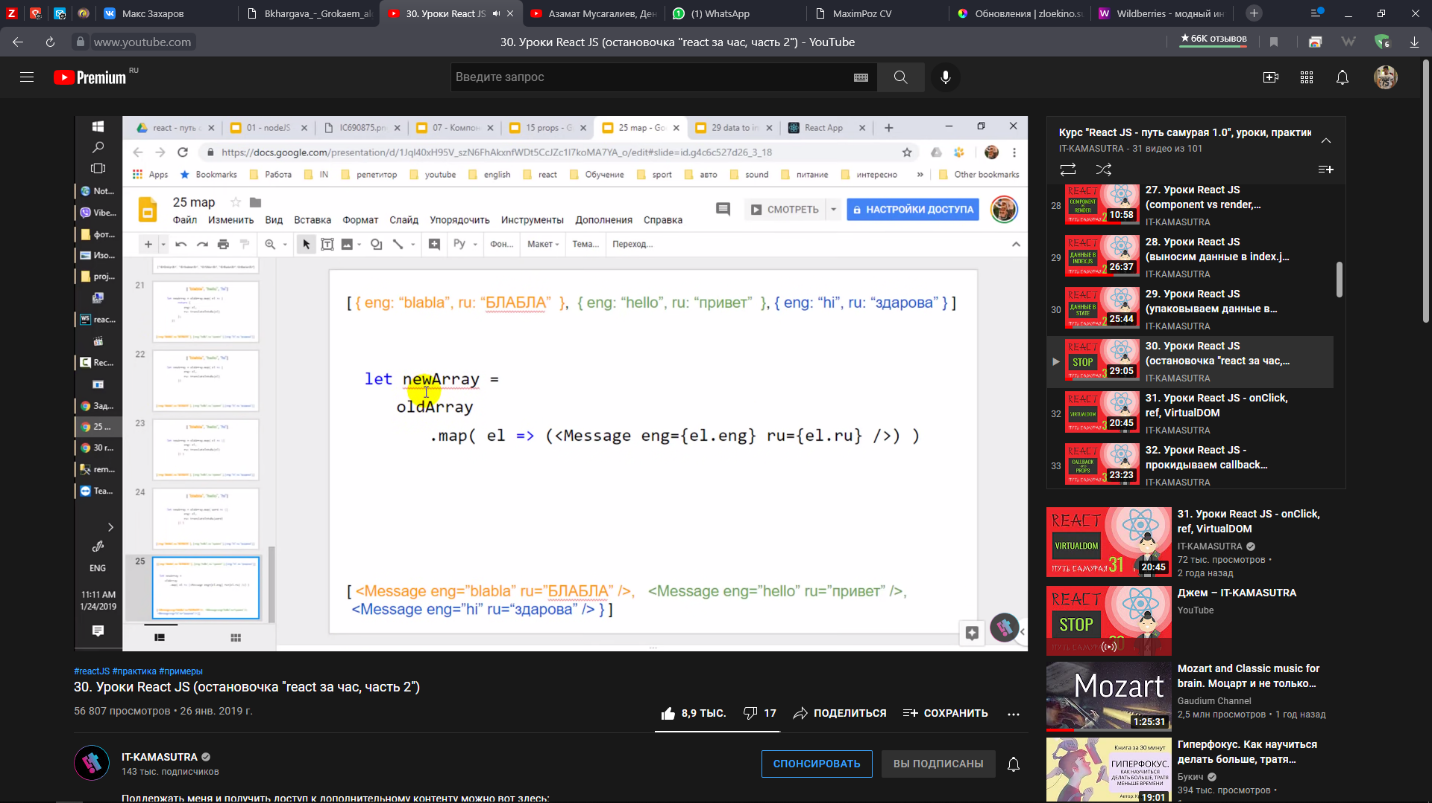
**31 лекция.**

1. NodeJS – для того что бы компилировать то что мы пишем в нормальный JS для браузера. Мы пишем браузерное приложение.



1. SPA (single page application) – это одна страница HTML и подгружаемый к ней JSon код. То есть мы не каждый раз запрашиваем новую страничку, а запрашиваем код, который рисует нам на одной и той же странице HTML.
2. Функциональная компонента (презентационная\presentational, без состояния\stateless, тупая\dumb). Компонент – это ф-ция которая возвращает JSX разметку.



1. Props – это объект в который мы с помощью тега (в другом JSX файле) можем передать атрибуты, которые станут свойством данного объекта props, а значение этого атрибута станет значением объекта
2. map - метод массива. Он нужен для преобразования одного массива в другой массив. Мы пробегаемся по каждому элементу, для этого вызывается callback ф-ция (стрелочная ф-ция), с помощью стрелочной ф-ции вместо el вызываются поочерёдно старые исходные элементы и мы возвращаем новый объект который возвращается в новый массив.
3. Route – анализирует URL и отрисовывает нужный нам JSX. Следит за адресной строкой
4. UI – BLL (User Interface – Business Logic Layer) - react отрисовывает пользовательский интерфейс и организовывает взаимодействие. BLL это redux в котором у нас сидит вся логика, которая отвечает за сложные изменения с нашей базой данных.

BrowserRouter — следует использовать, когда вы обрабатываете на сервере динамические запросы.

Route – следит за тем что в адресной строке URL, и если там path который соответствует данному Route, то там отрисовывется то что записано в render. Как только в URL пропадает данный path (к примеру /Dialogs) тогда всё стирается.

**onClick – при нажатии на кнопку выполнить функцию.**

Можно записать двумя способами:

1. let addPost = () => {

alert ('Its a live!!!!')}

<button onClick = { addPost }> Add post </button>

1. <button onClick = {() => (alert ( "its a live!!!" )) }> Add post </button>

В этих примерах, в onClick мы передаём callback ф-цию. Мы отдаём свою ф-цию, а когда кнопка посчитает нужным (при нажатии) она вернёт нам нашу ф-цию.

Для создания ссылок

let newPostElement = React.createRef(); \\создаём ссылку на textarea

let addPost =()=>{ \\ создаём ф-цию которая

let text = newPostElement.current.value; \\из current’a (textarea) newPostElement будет \\брать value (то что мы вводим в посте)

alert(text) } \\ и выводить его в модально окне

return (

<div className={s.postsBlock}>

<h3>My posts </h3>

<div><textarea ref={newPostElement}>123</textarea></div> \\ указываем что значение (value) будем брать отсюда

<div><button onClick={ addPost }> Add post</button></div> \\callback ф-ция

**32 лекция. Прокидываем callback через props**

Для того что бы наша информация в постах могла изменять посты по нажатию кнопки мы должны создать функцию в BLL (state.js) и прокинуть её до MyPosts через пропс.

In State

export let addPost = (postMessage) => {

    debugger;

    let newPost = {

        id: 5,

        message: postMessage,

        likesCount: "God"

    }

    state.profilePage.posts.push(newPost);

}

In index

 <App state={state} addPost={addPost}/>

In App

 <Route path='/Profile' render={ () => <Profile

        state = {props.state.profilePage}

        addPost = {props.addPost}/>} />

In Profile

addPost={props.addPost}/>

In MyPosts

let addPost =()=>{

        debugger;

    let text = newPostElement.current.value;

    props.addPost(text)

    }

43 Container component (контейнерная компонента)

Компонента – это функция которая принимает props и возвращает JSX разметку.

Контейнерная компонента – это компонента которая обслуживает презентационную (функциональную) компоненту, даёт нужные пропсы. Берёт на себя всю грязную работу.

49 Страница пользователя

Reducer – это чистая ф-ция которая принимает старый state и action, если надо то изменяет его и возвращает изменённую копию state’a. Если ничего не изменилось, то возвращает старый state.

50 REST API (краткая теория)

Get – получить от сервака данные (но на сервак много данных отправлять нельзя, т.е. нагрузки быть не должно) отправляем только к примеру id того элемента который хотим получить.

Post – отправляем на сервак данные

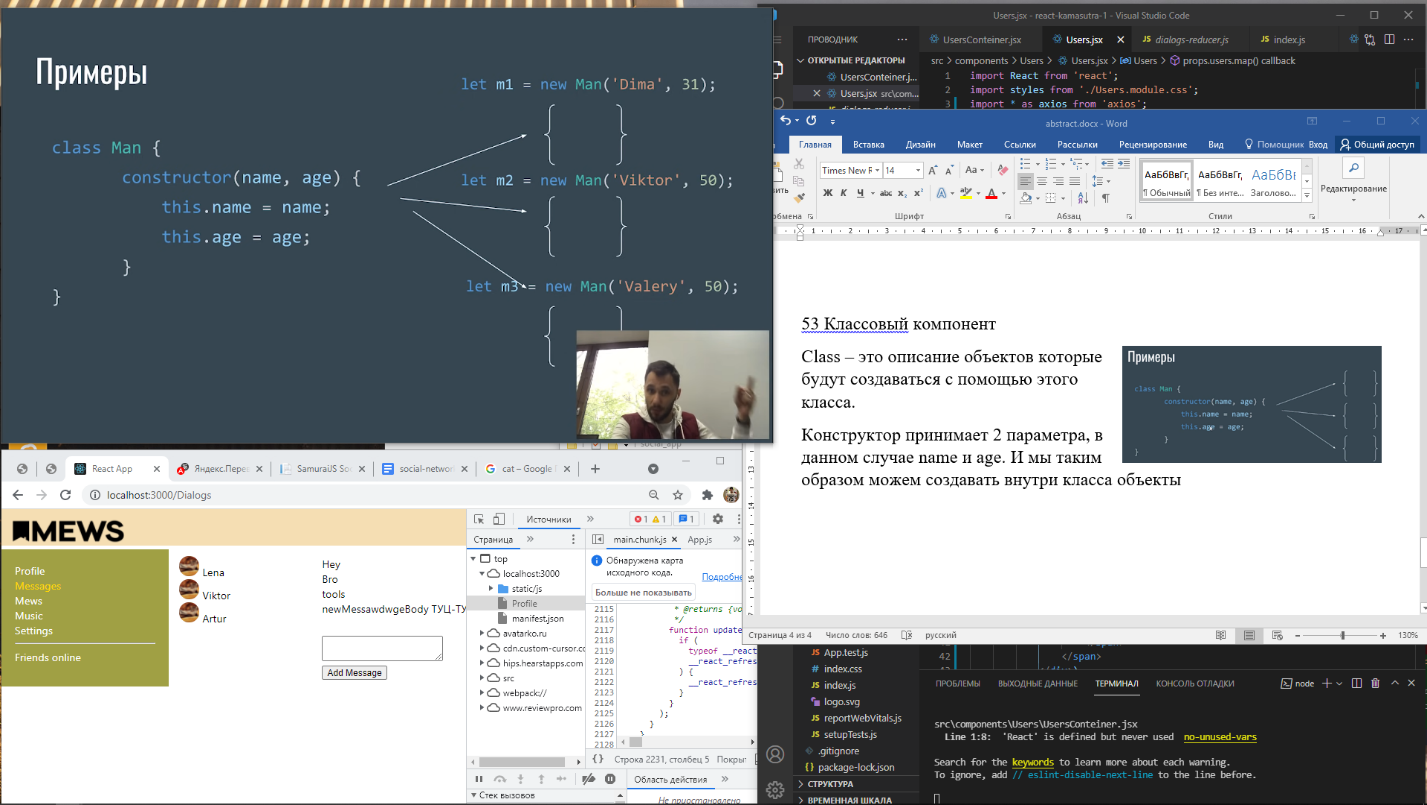
Put – обновляем что то (загружаем новую картинку, новый профиль заполнить и т.д.) тем самым создаём нагрузку на сервер (request payload)

Delete – при удалении на сервак отправляем только ID того элемента который хотим удалить.

**53 Классовый компонент**

**side effect -** это что-то, что может повлиять на "чистоту" вашей функции.

**Class** – это инструкция которая говорит из чего объекты будут состоять (name, age).

Конструктор принимает 2 параметра, в данном случае name и age. И мы таким образом можем создавать внутри класса объекты с помощью new className.