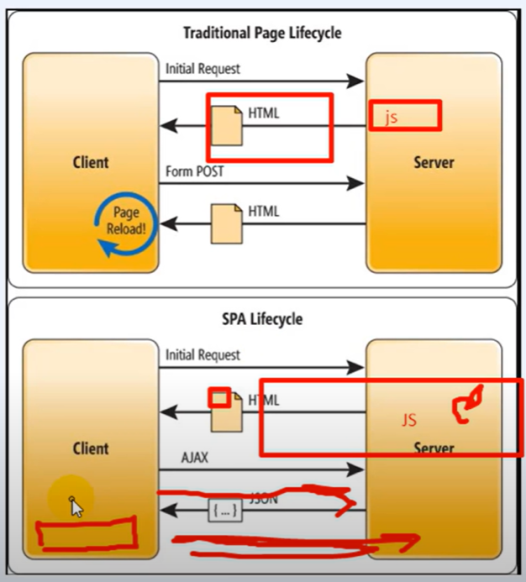
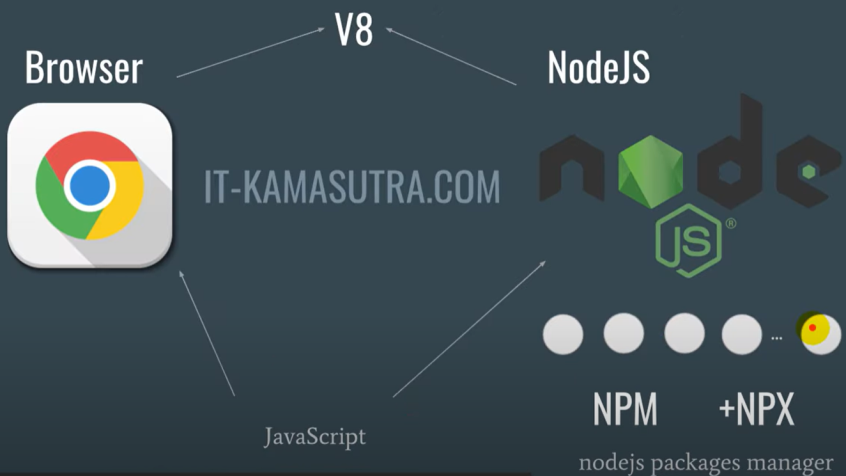
react shopping cart.

Есть такой чел max roslow на русском языке есть уроки по созданию интернт магазина с корзиной (react-shopping-cart) в англ ютубе так называют.

**(SPA: single page application)**

SPA - маленькая полупустая HTML страница, в которую приходит много JS. Этот JS работает на клиенте и создает динамически нужную страницу. Даже если мы переключаемся на другой раздел сайта - новый HTML не подгружается. JS ловит изменение url, отправляет AJAX запрос на сервер, возвращается JSON с данными и по ним JS рисует нужный HTML.

 a part of HTML

***и принимает параметр props***

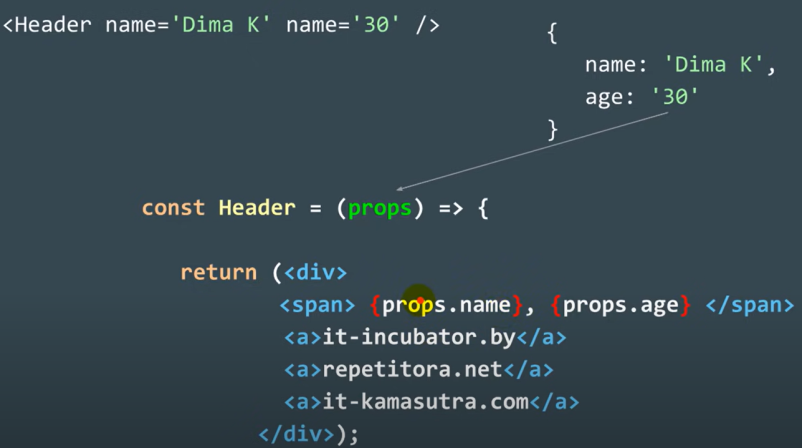
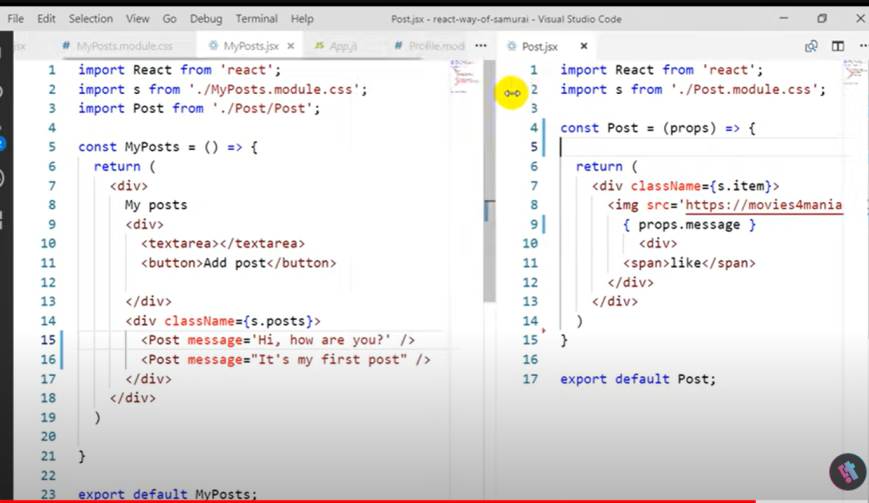
Компонента – это тег <App /> или <Header />

*пишу сниппет rafc нажимаю tab Компонент готов)*

*коммент –* ctrl+K+C

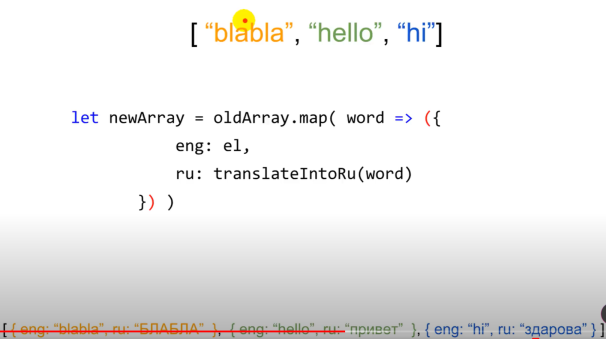
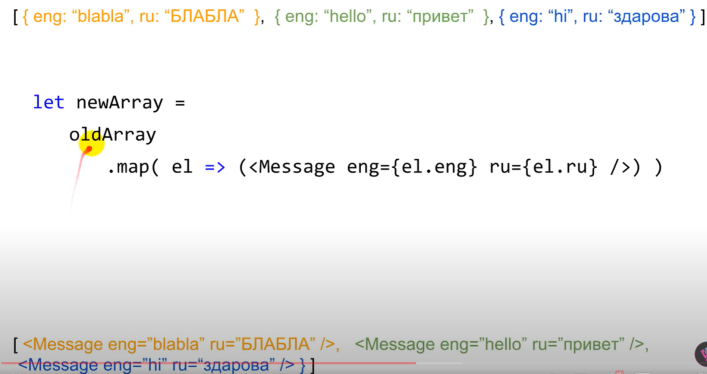


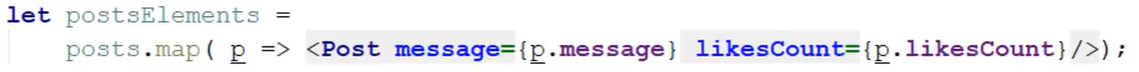
Если вы смотрите эти курсы в 2021, то, скорее всего, у вас может не работать npm start в файле Димы. Как я понял, это происходит из-за того, что его версия npm старее нынешних, и его проект подточен под старую версию. Именно поэтому, чтобы вы, как я, не сидели у компа чуть ли не день, чтобы выяснить в чем проблема, предлагаю решение): 1) Проверяете версию при помощи "npm -v"; 2) В случае если версия не 6, устанавливаете ту самую версию при помощи "npm install -g npm@6.14.10"; 3) Снова проверяете версию при помощи "npm -v"; 4) Если показывает 6.14.10, то удаляете имеющуюся папку node\_modules; 5) Переустанавливаете node\_modules под 6 версию при помощи "npm install"; 6) После установки вводете "npm fix audit"; 7) Запускаете npm "npm start". Удачи) Готово).

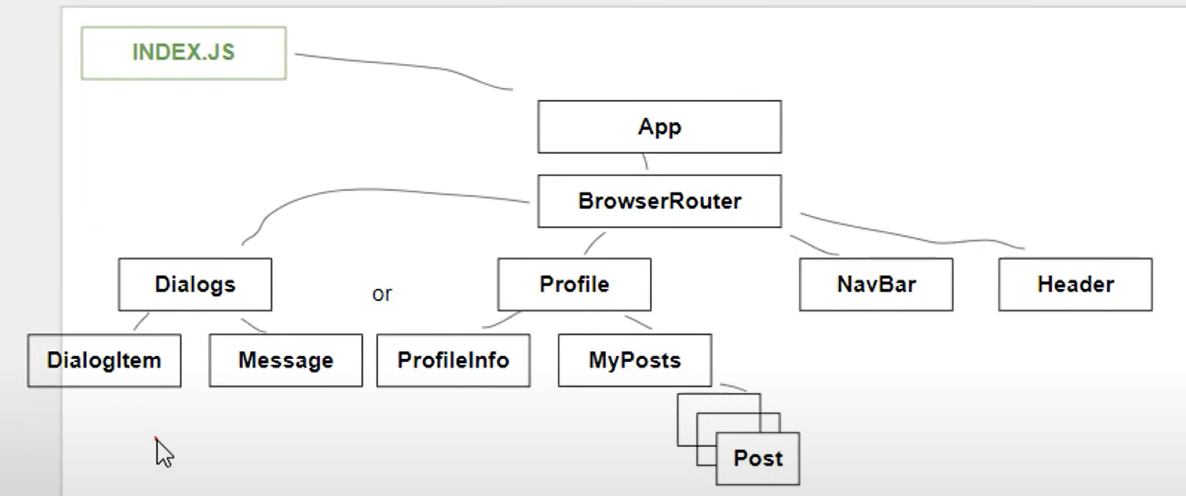
 

из профайл убрать ссс для фпп-враппер-контент

   map принимает стрелочную функцию и преобразует элементы в другой массив





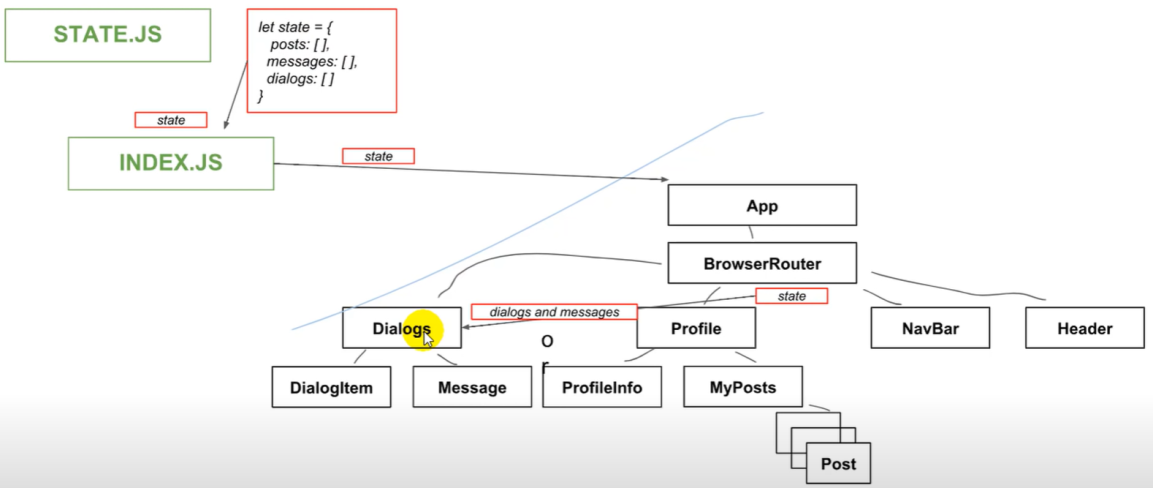
**<App dialogs={dialogs} messages={messages} posts={posts} />**

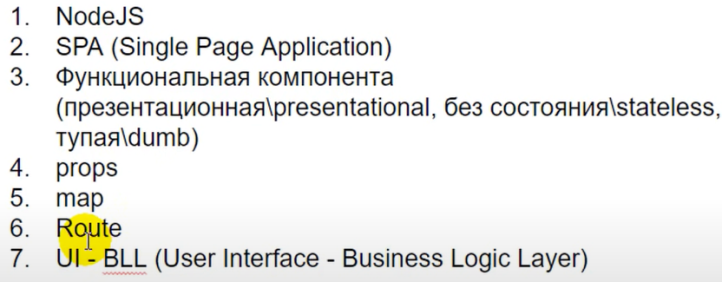
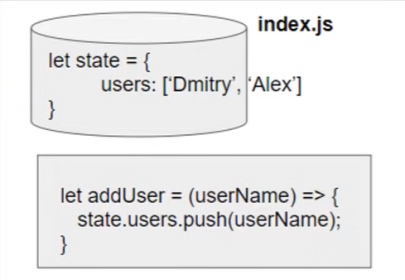
**И затем в роутах:**

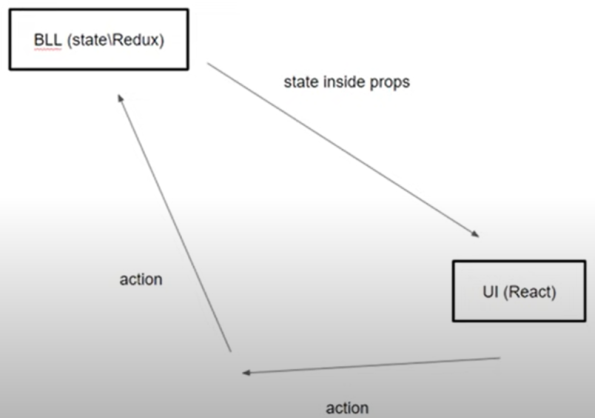
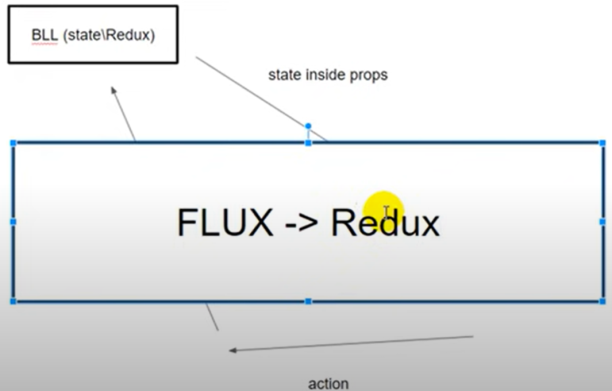
**...**

**<Route path="profile" element={<Profile posts={props.posts} />} />**

**<Route path="dialogs/\*" element={<Dialogs dialogs={props.dialogs} messages={props.messages} />} />**

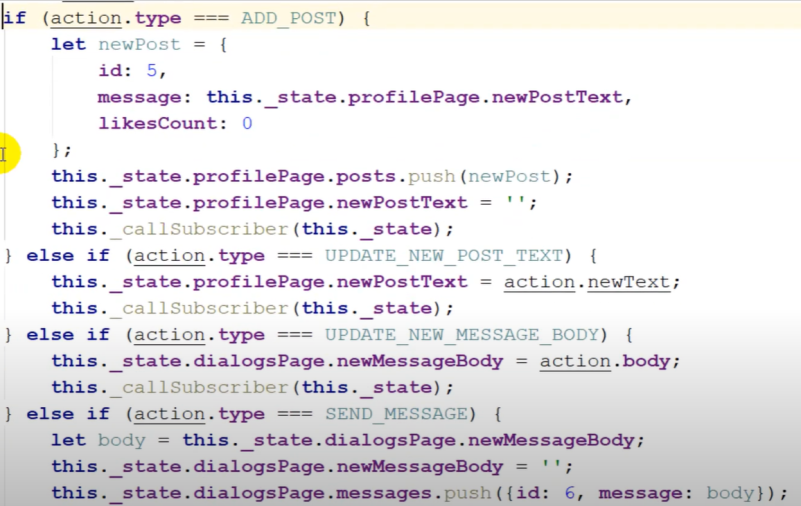


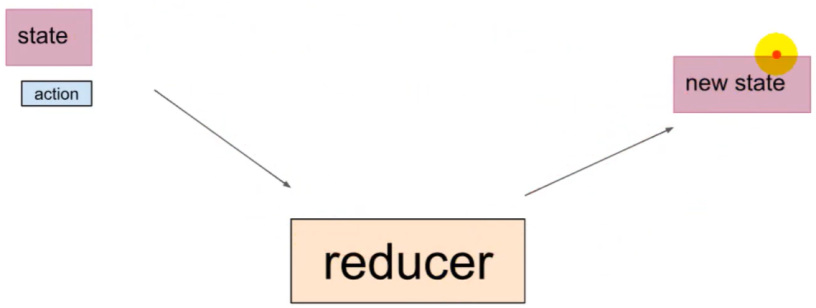


 ===

* **value**, то бишь каждый символ **в textarea**, мы берем **из BLL, в стейте**. Делаем мы это **через props**.
* Чтобы добавить каждый символ в стейт, т.е. наше value, мы используем обработчик onChange.
* Программируем наш onChange, чтобы value (символ который мы нажали) передавался в стейт. Делаем это через функцию **update, которая должна лежать со стейтом в BLL**.
* Прокидываем эту функцию через props в нашу компоненту. В обработчике пишем, вызови update(со значением value(символ)). т. е. то, что мы ввели, через функцию записывается в какой-то массив в стейте.
* А textarea говорит: О! Сейчас кто-то ввел символ и мой value стал тем, что ввели. Быстренько отображаю это, в поле ввода.
* Получается, **сначала поменялся state в BLL, а потом Ui в textarea. Это концепция Flux архитектуры**.
* 1.render находится в index,но его нужно передать в state,но передать не можем т.к
* a)это будет цикл.зависимость
* b) **index импортирует state**. и смотри a)
* 2.создали ф-ц **subscribe с параметр observer**,которая даёт себя на export и ждёт пока кто-то её пригласит (*обсервер является паттерном*)
* 3.эту ф-ц принимает в гости index и даёт ей подарочек в виде render
* 4.**subscribe возвращается домой c render(подарок)** и распаковывает его



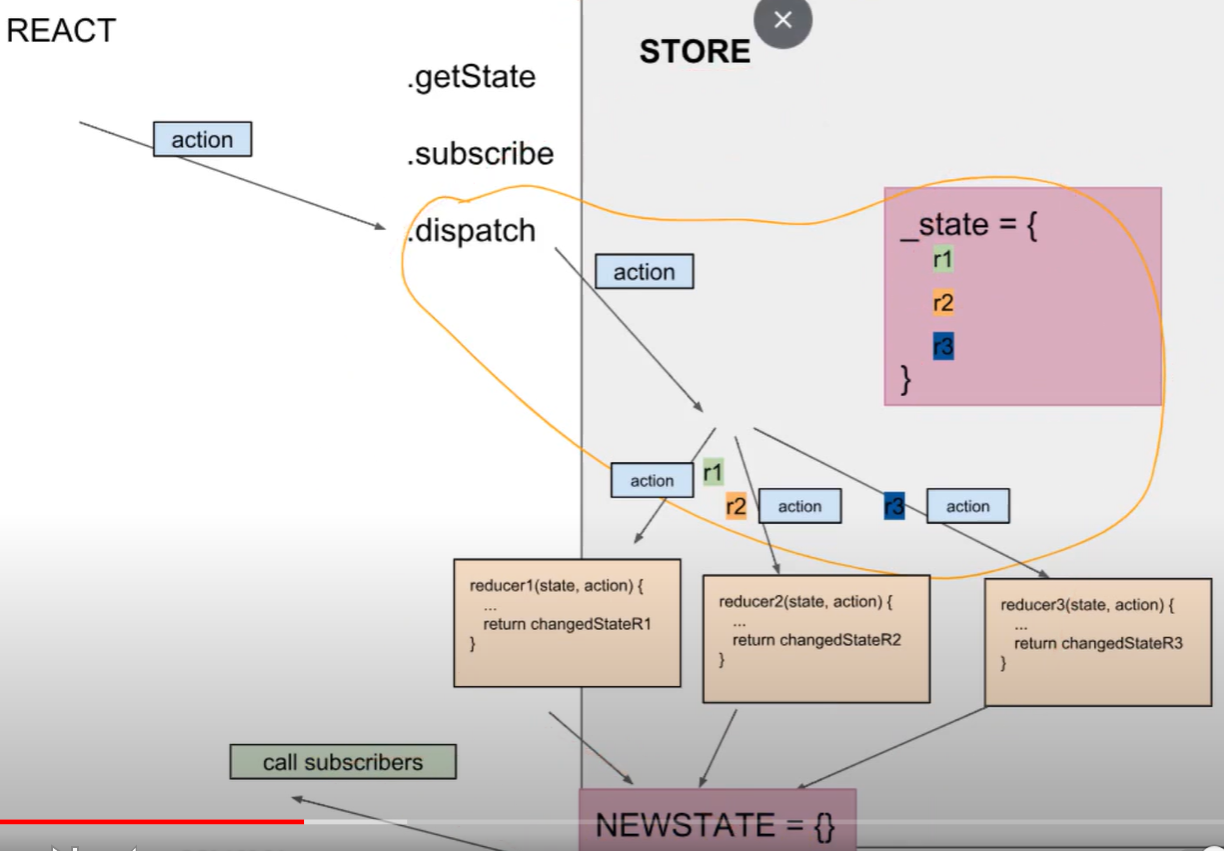
 **action** – объект со свойством *type*

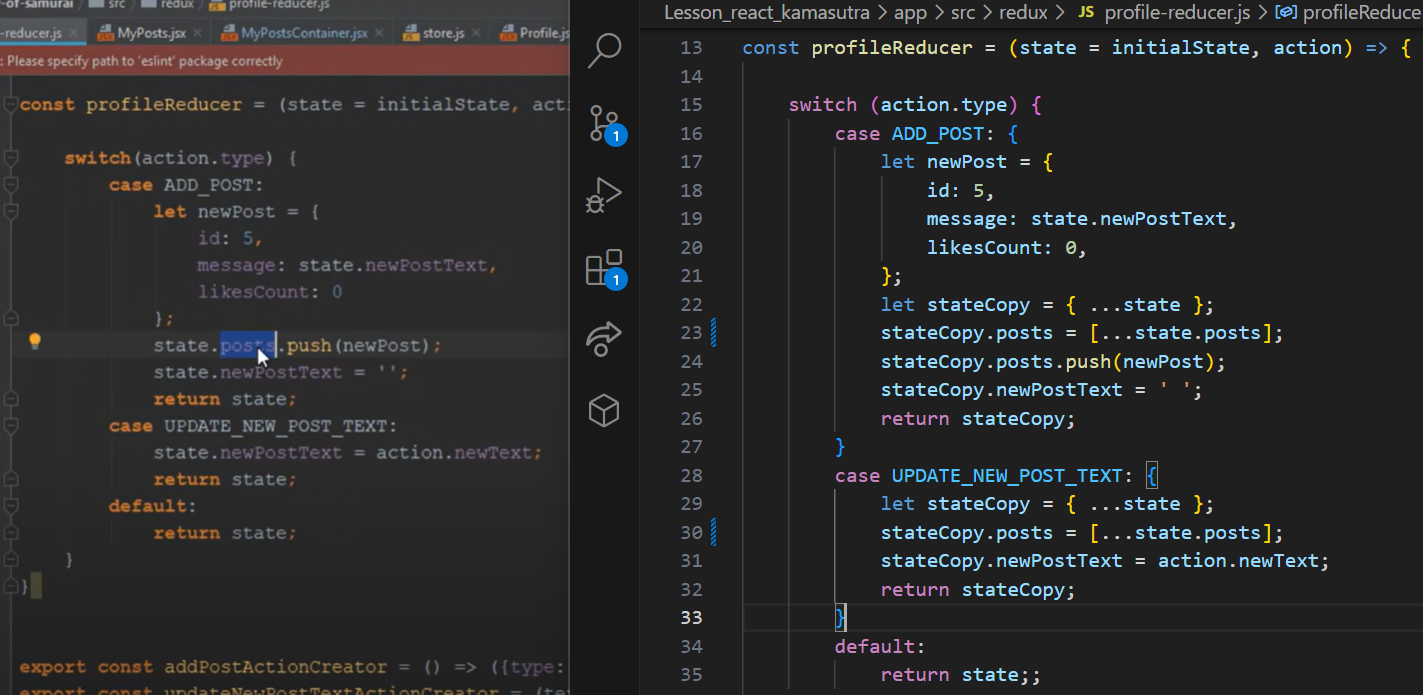
 чистая f принимает state+action => new state

***Функционал. программ. предполаг. разделение больших участков на раздельные маленькие функции***

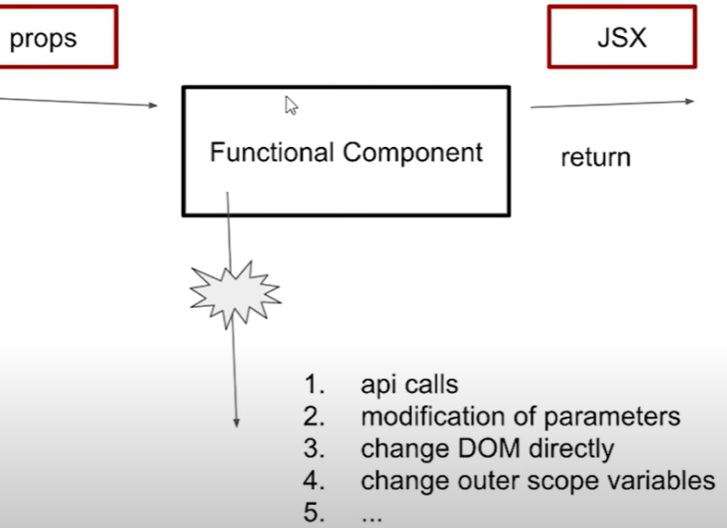
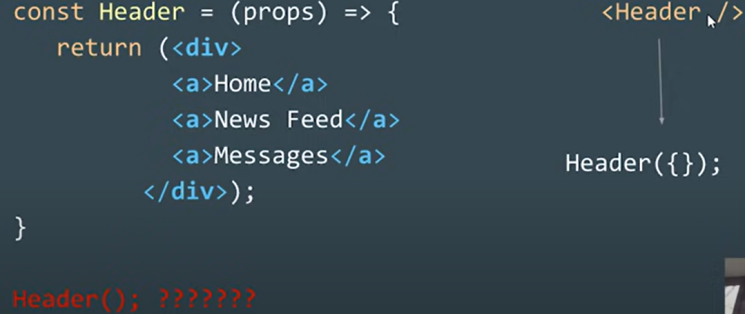
***Reduser – чистая функция принимает state+action returnes new state***

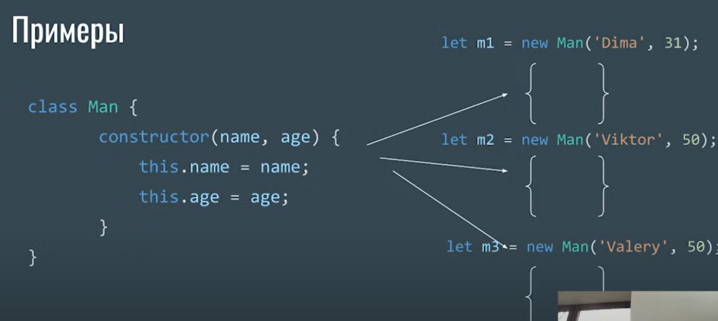
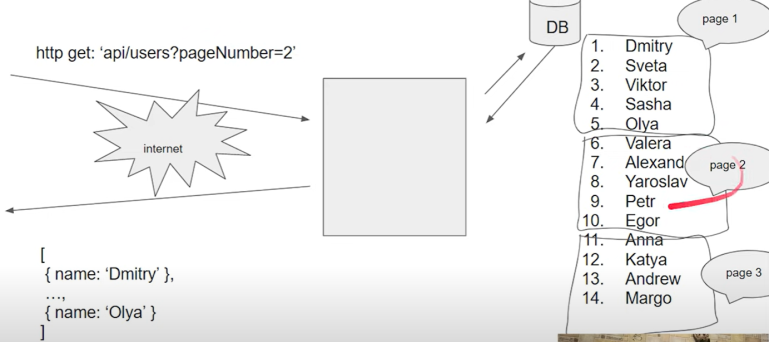
*** connect() вернет f1, и уже для f1 мы даем вторые скобки = f1()***

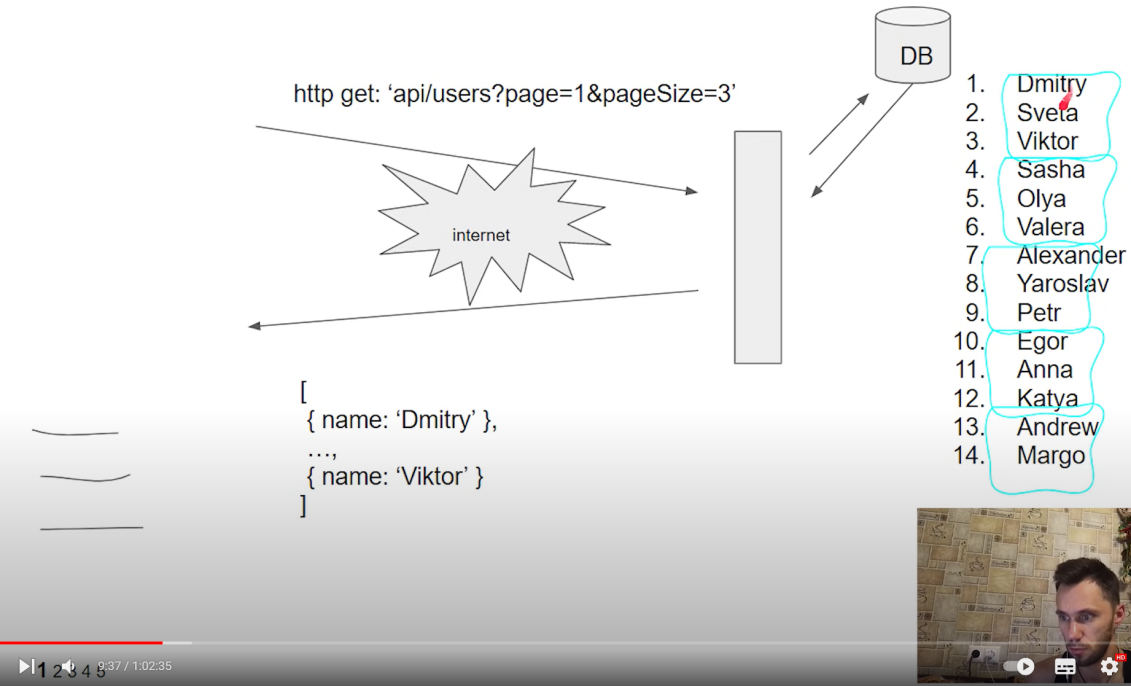


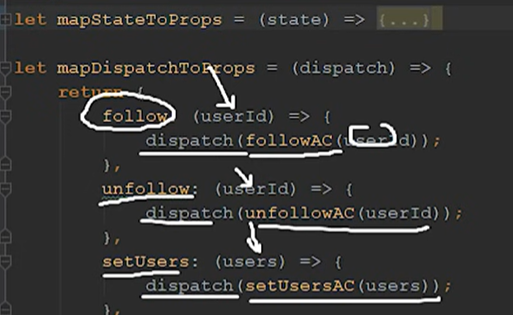






* Callback передаем в компоненту –
* Component вызывает callback передавая параметр ID –
* вызываем AC с этим ID –
* AC return action –
* диспатчим action