В этом уроке мы с вами подготовим наши данные для страницы товаров. Наши екшены уже готовы и нам осталось написать редьюсеры и подключить store к компоненту.

Итак сейчас у нас выстреливает екшен FETCH\_PHONES\_SUCCESS, который содержит в себе полученые данные.

Давайте теперь поговорим о редьюсерах, какие нам нужны. У нас интернет магазин телефонов и у нас есть специальный редьюсер phones, куда мы будем записывать все полученные телефоны и неважно из какого запроса они пришли. Эти данные мы будем хранить в виде обьекта с id телефона, как ключ. Это даст нам возможность легко вытаскивать необходимый нам телефон из обьекта и все телефоны хранятся уникально, так как в случае чего, если нам прийдут новые данные, то телефон просто перетрется.

Итак мы хотим в редьюсере phones отловить екшен FETCH\_PHONES\_SUCCESS и обновить данные в store. Давайте импортируем наш actionType и напишем switch на type екшена внутри.

import {

FETCH\_PHONES\_SUCCESS

} from 'actionTypes'

const initialState = {}

export default (state = initialState, {type, payload}) => {

switch (type) {

case FETCH\_PHONES\_SUCCESS:

default:

return state

}

}

Мы с вами получаем данные в виде массива обьектов. И нам нужно перезаписать каждый обьект в нашем стейте новыми данными.

Для работы и транформации данных я использую библиотеку Ramda. Это отличная библиотека в функциональном стиле, где из множества простых функций можно строить как из кирпичиков более сложные функции и их использовать.

В Ramda есть отличная функция подходящая под наш случай. Сначала нам нужно превратить массив обьектов в обьект с ключами. Для этого нам поможет метод indexBy

import R from 'ramda'

import {

FETCH\_PHONES\_SUCCESS

} from 'actionTypes'

const initialState = {}

export default (state = initialState, {type, payload}) => {

switch (type) {

case FETCH\_PHONES\_SUCCESS:

const newValues = R.indexBy(R.prop('id'), payload)

default:

return state

}

}

Что мы здесь делаем? Мы вызываем функцию Ramda indexBy, которая первым аргументом получает R.prop('id'), а вторым аргументом наши данные. R.prop мы используем, чтобы указать какое поле мы хотим видеть как ключ обьекта.

То есть теперь в newValues у нас записать обьект с ключами idшниками, каждый из которых содержит соответствующий телефон.

Теперь мы хотим померджить два обьекта между собой, то есть все значение обьекта newValues должны перетереть ключи обьекта state, если такие есть, а если нет, то добавить.

В это нам поможет функция merge.

import R from 'ramda'

import {

FETCH\_PHONES\_SUCCESS

} from 'actionTypes'

const initialState = {}

export default (state = initialState, {type, payload}) => {

switch (type) {

case FETCH\_PHONES\_SUCCESS:

const newValues = R.indexBy(R.prop('id'), payload)

return R.merge(state, newValues)

default:

return state

}

}

Ramda хороша тем, что все ее функции возвращают новые данные и никогда не мутируют старые. В случае с redux это особенно важно, так как стейт нельзя мутировать.

Если мы посмотрим в браузер, то в наш редьюсер phones записались телефоны в нужном нам виде.

Но это еще не все. Мы теперь сохраняем все полученные телефоны, но мы не можем просто взять их отсюда и все отрисовать, так как это все телефоны, которые мы могли получить из разных мест на сайте. Мы же хотим где-то хранить именно телефоны, которые мы получили на странице телефонов.

Для этого давайте создадим новый редьюсер phonesPage, где мы будем хранить все данные, относящиеся к странице телефонов, но которые на других страницах нам не пригодятся.

const initialState = {

ids: []

}

export default (state = initialState, {type, payload}) => {

switch (type) {

default:

return state

}

}

Здесь initialState у нас будет обьект с полем ids, где мы будем хранить idшники телефонов, которые мы будем отображать на этой странице. Теперь мы хотим тут слушать екшен FETCH\_PHONES\_SUCCESS и перезаписывать все idшники полученных треков в массив.

Для этого мы можем использовать функци pluck из Ramda, которая возьмет из массива обьектов все id и положит в новый массив

import R from 'ramda'

import {

FETCH\_PHONES\_SUCCESS

} from 'actionTypes'

const initialState = {

ids: []

}

export default (state = initialState, {type, payload}) => {

switch (type) {

case FETCH\_PHONES\_SUCCESS:

return R.merge(state, {

ids: R.pluck('id', payload)

})

default:

return state

}

}

То есть мы померджили наш state с новым обьектом, в который мы записали массив ids.

Теперь давайте добавим этот редьюсер в combineReducers.

import {combineReducers} from 'redux'

import {routerReducer} from 'react-router-redux'

import phones from './phones'

import phonesPage from './phonesPage'

export default combineReducers({

routing: routerReducer,

phones,

phonesPage

})

Если мы посмотрим в браузер, то в redux-devtools увидим новый редьюсер.

Если у вас что-то не получается, либо возникли вопросы и комментарии, пишите их прямо под этим видео и я обязательно на них отвечу.