В этом уроке мы с вами подготовим данные для нашей страницы отдельного товара.

Для начала давайте добавим новый роут в файл index. Но так как наш Layout не подходит под разметку страницы отдельного товара, то мы его не будем класть внутрь роута Layout

Импортируем новый контейнер

import Phone from 'containers/phone'

И добавим роут

ReactDOM.render(

<Provider store={store}>

<Router history={history}>

<Route component={Layout}>

<Route path='/' component={Phones} />

</Route>

<Route path='/phones/:id' component={Phone} />

</Router>

</Provider>,

document.getElementById('root')

);

Теперь давайте добавим этот контейнер в папку containers/phone.

import React, {Component} from 'react'

class Phone extends Component {

render () {

return (

<div>Phone</div>

)

}

}

export default Phone

Точно также, как мы фетчили телефоны на странице списка, нам нужно фетчить отдельный телефон в этом компоненте.

componentDidMount () {

this.props.fetchPhoneById()

}

В этот метод нам нужно передать id телефона из урл. Сделать это очень легко, так как в props у нас есть свойсто params, которое содержит все параметры урла. В нашем случае это id.

componentDidMount () {

this.props.fetchPhoneById(this.props.params.id)

}

Для того, чтобы fetchPhoneById был нам доступен в props, давайте подключим connect и добавим туда этот екшен

Импортируем connect и екшен и подключаем connect

import React, {Component} from 'react'

import {connect} from 'react-redux'

import {fetchPhoneById} from 'actions'

class Phone extends Component {

componentDidMount () {

this.props.fetchPhoneById(this.props.params.id)

}

render () {

return (

<div>Phone</div>

)

}

}

const mapDispatchToProps = {

fetchPhoneById

}

export default connect(null, mapDispatchToProps)(Phone)

Давайте теперь добавим actionTypes.

export const FETCH\_PHONE\_BY\_ID\_START = 'FETCH\_PHONE\_BY\_ID\_START'

export const FETCH\_PHONE\_BY\_ID\_SUCCESS = 'FETCH\_PHONE\_BY\_ID\_SUCCESS'

export const FETCH\_PHONE\_BY\_ID\_FAILURE = 'FETCH\_PHONE\_BY\_ID\_FAILURE'

Импортируем екшены

import {

FETCH\_PHONES\_START,

FETCH\_PHONES\_SUCCESS,

FETCH\_PHONES\_FAILURE,

LOAD\_MORE\_PHONES\_START,

LOAD\_MORE\_PHONES\_SUCCESS,

LOAD\_MORE\_PHONES\_FAILURE,

FETCH\_PHONE\_BY\_ID\_START,

FETCH\_PHONE\_BY\_ID\_SUCCESS,

FETCH\_PHONE\_BY\_ID\_FAILURE

} from 'actionTypes'

И создадим этот екшен, передав ему на вход id и вызвав fetchPhoneByIdApi

export const fetchPhoneById = (id) => async dispatch => {

dispatch({type: FETCH\_PHONE\_BY\_ID\_START})

try {

const phone = await fetchPhoneByIdApi(id)

dispatch({

type: FETCH\_PHONE\_BY\_ID\_SUCCESS,

payload: phone

})

} catch (err) {

dispatch({

type: FETCH\_PHONE\_BY\_ID\_FAILURE,

payload: err,

error: true

})

}

}

Теперь давайте импортируем fetchPhoneByIdApi.

import {

fetchPhones as fetchPhonesApi,

loadMorePhones as loadMorePhonesApi,

fetchPhoneById as fetchPhoneByIdApi,

} from 'api'

И опишем этот метод. В нем мы хотим найти телефон в массиве наших телефонов. Для этого я использую метод Ramda find, куда я передаю метод propEq, который находит елемент в массиве по id.

export const fetchPhoneById = async (id) => {

return new Promise((resolve, reject) => {

const phone = R.find(R.propEq('id', id), phones)

resolve(phone)

})

}

И конечно, нужно не забыть импортировать Ramda

import R from 'ramda'

Если мы посмотрим в браузер, то у нас выстреливают 2 правильных екшена и второй из них идет с данными в payload.

Теперь давайте добавим обработку этих actionTypes в редьюсеры.

Сначала давайте добавим FETCH\_PHONE\_BY\_ID\_SUCCESS в наш редьюсер phones. Для того, чтобы добавить в обьект новый ключ, мы можем использовать метод Ramda assoc. Первым параметром идет ключ, а вторым значение. Третьим параметром идет наш state.

import R from 'ramda'

import {

FETCH\_PHONES\_SUCCESS,

FETCH\_PHONE\_BY\_ID\_SUCCESS,

LOAD\_MORE\_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)

case LOAD\_MORE\_PHONES\_SUCCESS:

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

return R.merge(state, moreValues)

case FETCH\_PHONE\_BY\_ID\_SUCCESS:

return R.assoc(payload.id, payload, state)

default:

return state

}

}

Теперь давайте создадим редьюсер phonePage. В нем у нас будет initialState обьект с ключем id. И соотвественно мы будем получать телефон из редьюсера phones по id.

const initialState = {

id: null

}

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

switch (type) {

default:

return state

}

}

Теперь давайте добавим FETCH\_PHONE\_BY\_ID\_SUCCESS, который будет заменять id в нашем store.

import R from 'ramda'

import {

FETCH\_PHONE\_BY\_ID\_SUCCESS

} from 'actionTypes'

const initialState = {

id: null

}

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

switch (type) {

case FETCH\_PHONE\_BY\_ID\_SUCCESS:

return R.merge(state, {

id: R.prop('id', payload)

})

default:

return state

}

}

Теперь давайте подключим phonePage редьюсер в combineReducers.

import {combineReducers} from 'redux'

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

import phones from './phones'

import phonesPage from './phonesPage'

import phonePage from './phonePage'

export default combineReducers({

routing: routerReducer,

phones,

phonesPage,

phonePage

})

Если мы посмотрим в браузер, то в store у нас появился phonePage и id в нем, а в phones один елемент, который мы зафетчили.

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