В этом уроке мы с вами отрендерим страницу отдельного товара, по данным, которые мы приготовили в прошлом уроке.

Для начала давайте добавим mapStateToProps в connect на phonePage.

const mapStateToProps = state => {

return {

phone: getPhoneById(state, state.phonePage.id)

}

}

const mapDispatchToProps = {

fetchPhoneById

}

export default connect(mapStateToProps, mapDispatchToProps)(Phone)

Мы хотим получить телефон по id, который мы записали в phonePage редьюсер. Для этого мы используем селектор getPhoneById. Этот метод мы уже создали с вами ранее, когда писали селектор getPhones. Нам нужно только его експортировать.

import R from 'ramda'

export const getPhoneById = (state, id) => R.prop(id, state.phones)

export const getPhones = state => {

const phones = R.map(id => getPhoneById(state, id), state.phonesPage.ids)

return phones

}

export const getRenderedPhonesLength = state => R.length(state.phonesPage.ids)

Теперь мы можем его импортировать в нашем компоненте

import {getPhoneById} from 'selectors'

Давайте напишем console.log в render методе, чтобы проверить что мы получили телефон и он доступен нам в props.

render () {

console.log(this.props.phone)

return (

<div>Phone</div>

)

}

Если мы посмотрим в браузер, то в консоли у нас вывелся наш обьект.

Теперь давайте напишем разметку. Так как Layout мы сейчас не используем, то нам нужно написать всю разметку с самого начала, как было в Layout.

render () {

return (

<div className='view-container'>

<div className='container'>

<div className='row'>

<div className='col-md-9'>

{this.renderContent()}

</div>

<div className='col-md-3'>

{this.renderSidebar()}

</div>

</div>

</div>

</div>

)

}

Теперь давайте добавим методы renderSidebar и renderContent.

renderContent () {

return (

<div>Content</div>

)

}

renderSidebar () {

return (

<div>Sidebar</div>

)

}

Если мы посмотрим в браузер, то у нас отрендерилось 2 блока.

Теперь давайте в методе renderContent отрендерим все поля.

renderContent () {

const {phone} = this.props

return (

<div className='thumbnail'>

<div className='row'>

<div className='col-md-6'>

<img

className='img-thumbnail'

src={phone.image}

alt={phone.name}

/>

</div>

<div className='col-md-6'>

Our fields

</div>

</div>

<div className='caption-full'>

<h4 className='pull-right'>${phone.price}</h4>

<h4>{phone.name}</h4>

<p>{phone.description}</p>

</div>

</div>

)

}

Как мы видим, мы просто вставили поля обьекта в разметку. Если мы посмотрим в браузер, то увидим в консоли ошибку cannot read property of undefined. И это естественно, так как фетч данных у нас асинхронный и поэтому phone при загрузке страницы undefined, а через какое-то время заполняется. Для того, чтобы убрать эту ошибку, мы можем рендерить content и sidebar, только если у нас есть телефон. Давайте добавим проверку.

render () {

const {phone} = this.props

return (

<div className='view-container'>

<div className='container'>

<div className='row'>

<div className='col-md-9'>

{phone && this.renderContent()}

</div>

<div className='col-md-3'>

{phone && this.renderSidebar()}

</div>

</div>

</div>

</div>

)

}

Как мы видим, ошибка исчезла и у нас отрендерился наш телефон.

Теперь давайте вместо текста OurFields вызовем this.renderFields() Это метод, который будем рендерить таблицу с полями товара. Мы бы могли конечно вывести их руками, но так как полей много, лучше написать метод с циклом.

Для начала импортируем Ramda.

import R from 'ramda'

И получим телефон из props.

renderFields () {

const {phone} = this.props

}

Теперь мы хотим получить из этого обьекта только нужные нам поля и превратить обьект в массив, который мы сможем вывести маппом.

R.compose делает композицию из функций и они выполняются справа налево. То есть мы сначала используем метод pick, чтобы оставить только необходимые нам поля, а потом toPairs, чтобы превратить каждую пару ключ значение в елемент массива.

renderFields () {

const {phone} = this.props

const columnFields = R.compose(

R.toPairs,

R.pick([

'cpu',

'camera',

'size',

'weight',

'display',

'battery',

'memory'

])

)(phone)

console.log(columnFields)

}

Если мы посмотрим в браузер, то в консоль у нас вывелся массив массивов, где первый елемент у нас ключ, а второй значение. И теперь мы можем их отрендерить.

Для этого напишем map с деструктуризацией и выведем ключ и значение.

renderFields () {

const {phone} = this.props

const columnFields = R.compose(

R.toPairs,

R.pick([

'cpu',

'camera',

'size',

'weight',

'display',

'battery',

'memory'

])

)(phone)

return columnFields.map(([key, value]) => (

<div className='column' key={key}>

<div className='ab-details-title'>

<p>{key}</p>

</div>

<div className='ab-details-info'>

{value}

</div>

</div>

))

}

Если мы посмотрим в браузер, то у нас вывелась таблица с полями.

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