В этом уроке мы отрендерим список категорий и добавим фильтр товаров по категориям.

Для начала давайте отрендерим их. Для этого давайте создадим новый компонент categories

import React from 'react'

const Categories = () => {

return (

<div>Categories</div>

)

}

export default Categories

И выведем его в sidebar

import React from 'react'

import BasketCart from 'components/basketCart'

import Search from 'components/search'

import Categories from 'components/categories'

const Sidebar = () => {

return (

<div>

<BasketCart />

<Search />

<Categories />

</div>

)

}

export default Sidebar

Как мы видим, у нас в браузер в sidebar отрисовался наш компонент. Теперь давайте подключим к нему connect.

import React from 'react'

import {connect} from 'react-redux'

const Categories = () => {

return (

<div>Categories</div>

)

}

const mapStateToProps = state => ({

categories: getCategories(state)

})

export default connect(mapStateToProps, null)(Categories)

Теперь давайте в селекторах опишем метод getCategories и импортируем его в компонент

import {getCategories} from 'selectors'

Используя метод R.values мы можем получить все категории в виде массива оз обьекта

export const getCategories = state => R.values(state.categories)

Теперь давайте отрисуем категории

import React from 'react'

import {connect} from 'react-redux'

import {Link} from 'react-router'

import {getCategories} from 'selectors'

const Categories = ({categories}) => {

const renderCategory = (category, index) => {

return (

<Link

to={`/categories/${category.id}`}

className='list-group-item'

key={index}

>

{category.name}

</Link>

)

}

return (

<div className='well'>

<h4>Brand</h4>

<div className='list-group'>

{categories.map((category, index) => renderCategory(category, index))}

</div>

</div>

)

}

const mapStateToProps = state => ({

categories: getCategories(state)

})

export default connect(mapStateToProps, null)(Categories)

Если мы посмотрим в браузер, то у нас вывелся список категорий. При клике на категорию даже меняется ссылка в урле, но товары пока что не фильтруются.

Сейчас нам нужно реализовать подсветку категорий. Для этого нам нужно из урла получить id категории и подсветить категорию, у которой такой же id активным классом.

Для этого мы можем обернуть наш компонент в withRouter так же, как мы его оборачивали в connect. Но так как мы хотим одновременно применить и withRouter и connect, то мы можем использовать метод compose из redux, так же, как мы использовали compose в Ramda.

import {Link, withRouter} from 'react-router'

import {compose} from 'redux'

export default compose(

withRouter,

connect(mapStateToProps, null)

)(Categories)

Метод withRouter дает нам возможность вторым параметром в mapStateToProps получить ownProps, которая содержит в себе id категории из url.

const mapStateToProps = (state, ownProps) => ({

categories: getCategories(state),

activeCategoryId: getActiveCategoryId(ownProps)

})

Теперь давайте опишем метод getActiveCategoryId в selectors.

export const getActiveCategoryId = ownProps => R.path(['params', 'id'], ownProps)

Для получения вложенных данных, которые могут быть undefined мы можем использовать метод R.path. Теперь если например ownProps будет undefined или ownProps.params будут undefined наш код не упадет

Давайте выведем в компоненте наш id

const Categories = ({categories, activeCategoryId}) => {

console.log('activeCategoryId', activeCategoryId)

...

}

Если мы посмотрим в браузер, то при смене категории у нас выводится активный id.

Теперь мы можем добавить класс active в метод renderCategory. Для добавления нескольких классов с условием можно использовать отличную библиотеку classNames.

const renderCategory = (category, index) => {

const getActiveState = R.propEq('id', activeCategoryId)

const linkClass = classNames({

'list-group-item': true,

'active': getActiveState(category)

})

return (

<Link

to={`/categories/${category.id}`}

className={linkClass}

key={index}

>

{category.name}

</Link>

)

}

Мы проверяем что activeCategoryId равняется id текущей категории и тогда добавляем класс active.

Не забываем также импортировать ramda и classnames.

import R from 'ramda'

import classNames from 'classnames'

Если мы посмотрим в браузер, то у нас подсвечивается активная категория.

Но теперь хотелось бы добавить ссылку Все товары, чтобы иметь возможность перейти на домашнюю страницу.

Для этого добавим метод renderAllCategory в return

return (

<div className='well'>

<h4>Brand</h4>

<div className='list-group'>

{renderAllCategory()}

{categories.map((category, index) => renderCategory(category, index))}

</div>

</div>

)

И опишем его

const renderAllCategory = () => {

const linkClass = classNames({

'list-group-item': true,

'active': R.isNil(activeCategoryId)

})

return (

<Link

to='/'

className={linkClass}

>

All

</Link>

)

}

Здесь мы ставим класс active, если у нас нет активной категории.

Если мы посмотрим в браузер, то у нас появилась ссылка ведущая на homepage с активным классом.

Теперь нам осталось только добавить фильтр товаров по категориям. Для этого нам нужно передать активную категорию в селектор getPhones. Для этого передадим ownProps, который у нас доступен вторым параметром в getPhones.

Если у вас возникает вопрос почему здесь ownProps доступны сразу, а в компоненте Categories нет, то потом, что это роут компонент и он рендерится прямо по роуту. Все вложенные компоненты нужно добавлять через withRouter.

const mapStateToProps = (state, ownProps) => {

return {

phones: getPhones(state, ownProps)

}

}

Теперь давайте модифицируем селектор getPhones. Мы хотим добавить фильтр по категориям, когда у нас есть активная категория. Для этого отлично подходим метод Ramda when. Первым параметром у нее должна идти функция, которая возвращает true или false и в зависимости от этого выполняется вторая функция.

Давайте сначала получим activeCategoryId

const activeCategoryId = getActiveCategoryId(ownProps)

const phones = R.compose(

R.filter(applySearch),

R.when(R.always(activeCategoryId), R.filter(applyCategory))

R.map(id => getPhoneById(state, id))

)(state.phonesPage.ids)

Это означает когда у нас есть activeCategoryId, мы хотим применить filter с функцие applyCategory. R.always превращает activeCategoryId в функцию, которая возвращает activeCategoryId. Это нужно только потому, что такое синтаксис when.

Теперь давайте опишем applyCategory. Здесь мы проверяем равняется ли текущая категория категории товара

const applyCategory = item => R.equals(

getActiveCategoryId(ownProps),

R.prop('categoryId', item)

)

Если мы посмотрим в браузер, то все работает и товары фильтруются в зависимости от категории.

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