**Роутинг**

Роутинга нет в чистой библиотеке React, поэтому подключим стороннюю реализацию:

npm I react-router-dom

Для того чтоб включить роутинг, нужно обвернуть все приложение в отдельный компонент в index.js:

import { BrowserRouter } from 'react-router-dom'

ReactDOM.render(

<BrowserRouter>

<App />

</BrowserRouter>,

document.getElementById('root')

)

Теперь нужно объявить роуты:

import { Route } from 'react-router-dom'

<Route path='/' exact component={Home} />

<Route path='/about' exact component={About} />

<Route path='/cars' exact component={Cars} />

<Route path='/setting' exact component={Setting} />

Роутам при создании нужно указать путь и component, либо указать render функцию:

<Route path='/setting' exact render={() => <Setting/>} />

Параметр exact заставляет роут появляться лишь в строго, когда будет его путь.

Между роутами можно переключаться стандартными ссылками, но в таком случае обновляется страница, поэтому нужно воспользоваться отдельным компонентом:

import { NavLink } from 'react-router-dom'

<NavLink to='/'>Home</NavLink>

Ему нужно передать в параметр to строку путь или объект, описывающий путь:

<NavLink to={{ pathname: '/' }}>Home</NavLink>

Кроме того этот объект может описать search и hash:

<NavLink to={{ pathname: '/about', search: '?version=1.1&part=react', hash: '#hash' }}>

About

</NavLink>

Навигация может осуществляться и программно, но только через специальные свойства. Они по-умолчанию передаются в параметры, любому компоненту, указанному в component.

import React from 'react'

function Cars(props) {

function back() {

props.history.push('/')

}

return (

<div>

<button onClick={back}>Back</button>

</div>

)

}

export default Cars

Если же компонент не указан в параметре component, то нужно его обвернуть в withRouter:

import React from 'react'

import { withRouter } from 'react-router-dom'

function Car(props) {

return (

<div onClick={() => props.history.push('/cars/' + name)}>

<h1>{props.name}</h1>

<h4>{props.year}</h4>

</div>

)

}

export default withRouter(Car)

Так какие же это свойства?

* **locations** позволяет получить, то что передали в параметре to
* **match** позволяет получить данные для сравнения – путь, указанный в path и url
* **history** позволяет осуществить навигацию. Основной метод push позволяет перейти по ссылке

Допустим при клике на машину, должна открываться страница с детальной информацией. Это можно сделать, если указать имя машины, как параметр роута. Как раз таки этот параметр можно получить в match.

import React from 'react'

function CarDetail({ match }) {

return <h1>{match.params.name}</h1>

}

export default CarDetail

Только создать роут для этого компонента можно, если указать строго такое же название:

<Route path='/cars/:name' exact component={CarDefail} />

Имея все необходимые роуты вместе их объявления будет каша, упорядочить которую можно через компонент Switch:

<Switch>

<Route path='/' exact component={Home} />

<Route path='/cars/:name' component={CarDefail} />

<Route path='/cars' component={Cars} />

<Route path='/about' component={About} />

<Route path='/setting' component={Setting} />

</Switch>

Switch будет отражать первый компонент, соответствующий условию, поэтому exact тут можно не указывать. Данный подход хорош, тем, что заставляет объявлять роуты от частного к общему.

Теперь нужно обработать несуществующий url адрес. Для этого можно в конце создать роут без параметра to и с параметром render / component

<Route render={() => <h1>Error 404</h1>} />

Компонент Redirect позволяет осуществить редирект от одного роута в другой. Его особенность в том, что его нельзя в Switch (кроме одного случая).

<Redirect from='/' to='/about' />

Редирект используется по такому назначению редко. Часто его используют для обработки ошибки 404, в таком случае его можно обвернуть в Switch:

<Switch>

<Route path='/' exact component={Home} />

<Route path='/cars/:name' component={CarDefail} />

<Route path='/cars' component={Cars} />

<Route path='/about' component={About} />

<Route path='/setting' component={Setting} />

<Redirect to='/' />

</Switch>

* подключение
* роуты
* навигация между ними
* программная навигация
* динамические роуты
* switch
* обработка 404
* редирект