React Router 3.0

Internal Training Web

EPAM Ryazan

Dmitrii Pikulin – About me

Dmitrii Pikulin

EPAM Systems, Software Engineer

- I am a front-and developer
- Work on
  - https://learn.epam.com/
  - https://grow.epam.com/
  - https://competency.epam.com/
- I am an RD trainer

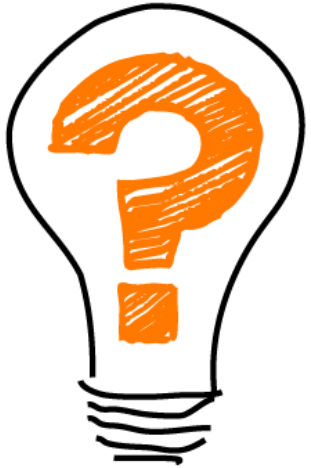
Agenda

1. **Overview**
   - History module
   - Handle 404 page
   - Nested routes
   - Index route
2. **Route configuration**
   - Params
   - Query
3. **Advanced**
   - Lazy routes
   - Redirect
   - Page transition animation
   - Navigation programmatically

Overview

Why use React Router?

- Sync URL with UI
- Bookmark Web Apps
- Keep The Back Button Useful



Star 17,706 Fork 4,146

REACT ROUTER

React Router features overview

1. **Cross platform**
   - Browser
   - Node.js
   - React Native
2. **Double configuration format**
   - JSX
   - Plain JavaScript object
3. **Rich API**
   - Routes can be loaded asynchronously
   - Redirects
   - Lifecycle hooks

React Router installation

```
npm i -S react-router
```

History - https://github.com/mjackson/history

01. history

History is a library that has minimal API that lets you manage the history stack, navigate, confirm navigation, and persist state between sessions.

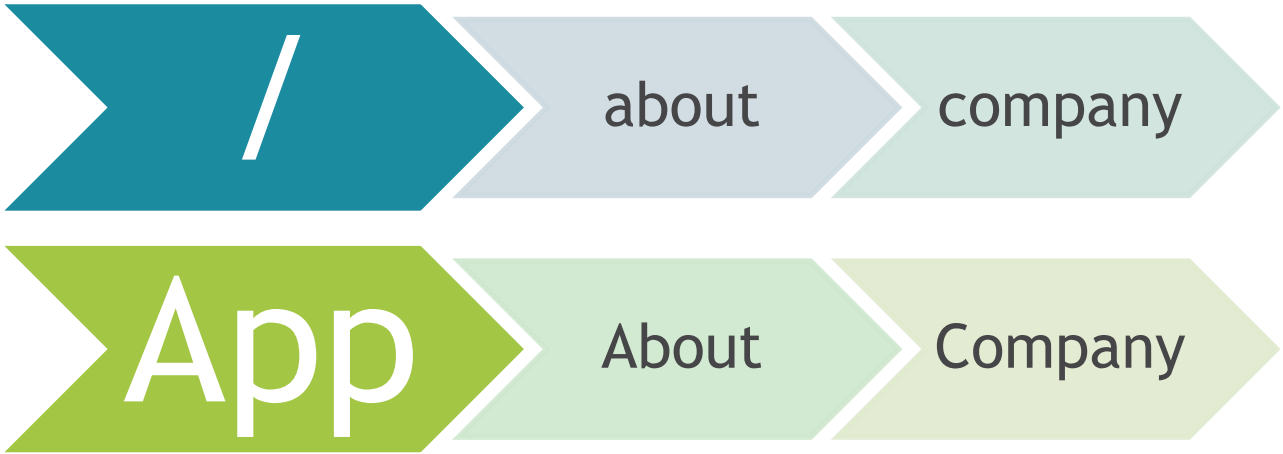
- `browserHistory` - is for use in modern web browsers that support the HTML5 history API
- `memoryHistory` - may be used in non-DOM environments, like React Native
- `hashHistory` - is for use in legacy web browsers

You can import them from the React Router package:

```
import { browserHistory } from 'react-router‘;
```

Parsing route

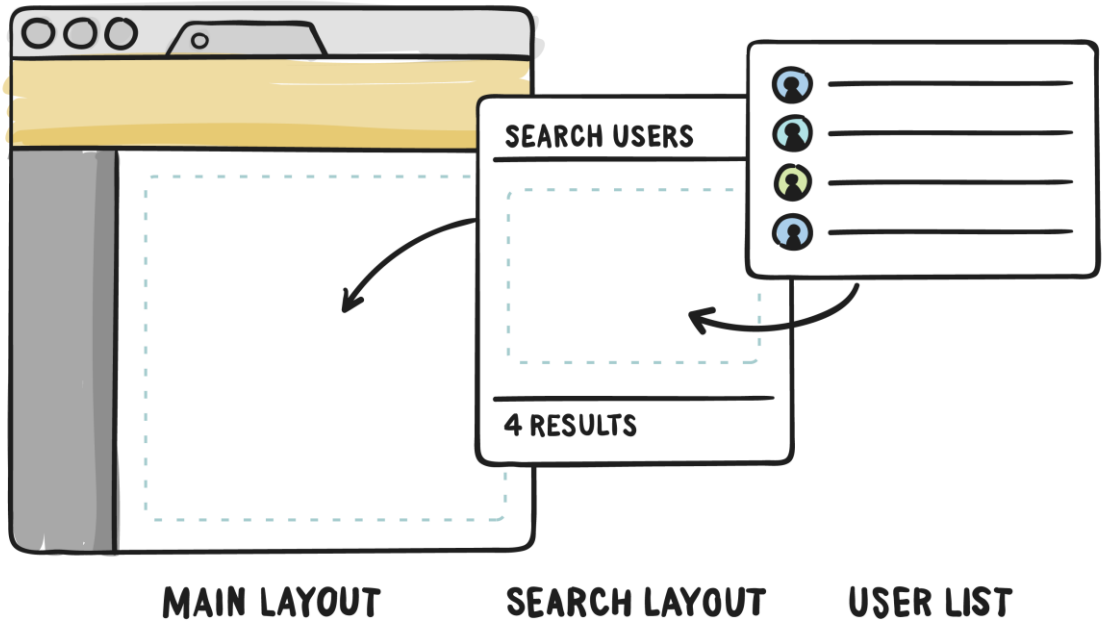
1. As a user you want to go different URLs
2. The URL is passed to router
3. Router takes the URL and returns the right component based on the URL to React
4. React will take that component and render that on the browser



Rendering a Route

02. hand_made

/search/list



The Basic Idea

Router is an object that you require from `react-router` that decides what react component to render when the URL changes

`hashHistory` is what interacts with URL and it passes the URL to react router

```
1.  import * as React from 'react';
2.  import * as ReactDom from 'react-dom';
3.  import { Router, Route, hashHistory } from 'react-router';
```

The Basic Idea

03. first_look

```
1.  ...
2.  import { App } from './App';
3.  import { About } from './About';
4.  import { Posts } from './Posts';
5.
6.  ReactDom.render((
7.      <Router history={hashHistory}>
8.          <Route path="/" component={App}>
9.              <Route path="posts" component={Posts} />
10.             <Route path="about" component={About} />
11.         </Route>
12.     </Router>
13. ), document.getElementById('app'));
```

- You have a component called App
- You have a Router with an indication that some change needs to happen when # something is entered e.g., /#/posts
- The URL where something needs to change is /about (/#/about)
- When / about is used, the About component should be displayed

Path Syntax

A route path is a string pattern that is used to match a URL (or a portion of one).

- `:paramName` – matches a URL segment up to the next /, ?, or #. The matched string is called a param
- `()` – wraps a portion of the URL that is optional
- `*` – matches all characters (non-greedy) up to the next character in the pattern, or to the end of the URL if there is none, and creates a splat param

```
1. <Route path="/hello/:name"> // matches /hello/Michael, /hello/ryan
2. <Route path="/hello(/:name)"> // matches /hello/Michael, /hello/ryan, /hello
3. <Route path="/files/*.*"> // matches /files/hello.jpg, /files/hello.html
```

Clean URLs

04. browser_history

```
1.  import * as React from 'react';
2.  import * as ReactDom from 'react-dom';
3.  import { Router, Route, browserHistory } from 'react-router';
4.  import { App } from './App';
5.  import { About } from './About';
6.  import { Posts } from './Posts';
7.
8.  ReactDom.render((
9.      <Router history={browserHistory}>
10.         <Route path="/" component={App}>
11.             <Route path="posts" component={Posts} />
12.             <Route path="about" component={About} />
13.         </Route>
14.     </Router>
15. ), document.getElementById('app'));
```

- Import `browserHistory` instead of `hashHistory`
- Change `history` prop value in `Router`
- You have to configure server to send `index.html` on every URL

404 page

05. not_found

```
1.  import * as React from 'react';
2.  …
3.  import { NotFound } from './NotFound';
4.
5.  ReactDom.render((
6.      <Router history={browserHistory}>
7.          <Route path="/" component={App}>
8.              <Route path="posts" component={Posts} />
9.              <Route path="about" component={About} />
10.             <Route path="*" component={NotFound} />
11.         </Route>
12. </Router>
13. ), document.getElementById('app'));
```

- React Router tries to match route from top to bottom
- It stops searching when the route is matched
- To match all use `*`
- Order matters

Navigating with Link

06. link

```
1.  import * as React from 'react';
2.  import { Link } from 'react-router';
3.
4.  export class App extends React.Component {
5.      render() {
6.          return (
7.              <div>
8.                  <h1>App</h1>
9.                  <ul>
10.                     <li><Link to="/about">About</Link></li>
11.                     <li><Link to={{ pathname: '/posts' }}>Posts</Link></li>
12.                 </ul>
13.
14.                 {this.props.children}
15.             </div>
16.         );
17.     }
18. }
19.
```

- Use `<Link to="/path" />` instead of `<a />` in react components to link to other parts of the application
- Link, is almost identical to the `<a />` tag you're used to except that it's aware of the Router it was rendered in.

Styling active routes

07. link_styling

A nice feature of the `<Link />` component is its ability to know when it's active.

We can set CSS styling of active routes using `activeClassName` prop

```
<Link to="/about" activeClassName="active_link">About</Link>
```

Or can assign a class with help of `activeStyle`

```
<Link to="/about" activeStyle={{ color: 'green' }}>About</Link>
```

Adding a 'Home' link

08. home_link

Actually, parent routes are active when child routes are active. Unfortunately, `/` is the parent of everything.

For this link, we want it to only be active when the index route is active.

```
<Link to="/" activeStyle={{ color: 'cyan' }} onlyActiveOnIndex={true}>Home</Link>
```

Or you can use `IndexLink`

```
1. …
2. import { Link, IndexLink } from 'react-router';
3. …
4. <IndexLink to="/" activeStyle={{ color: 'cyan' }}>Home</IndexLink >
5. …
```


Routes configuration

Nested Routes

09. nested_routes

Paths may have hierarchy. Each nested component become children of its parent.

```
1.  <Router history={browserHistory}>
2.      <Route path="/" component={App}>
3.          <Route path="posts" component={Posts}>
4.              <Route path="article" component={Article} />
5.          </Route>
6.          <Route path="about" component={About} />
7.          <Route path="*" component={NotFound} />
8.      </Route>
9.  </Router>
```

Don’t forget to render children in parent component. Now `this.props.children` can render `Article` component.

```
1.   <div>
2.       <h2>Posts</h2>
3.       {this.props.children}
4.   </div>
```

Index Route

To illustrate the use case for `IndexRoute`, imagine the following route config without it:

```
1.  <Router>
2.    <Route path="/" component={App}>
3.      <Route path="accounts" component={Accounts}/>
4.      <Route path="statements" component={Statements}/>
5.    </Route>
6.  </Router>
```

When the user visits /, the App component is rendered, but none of the children are, so `this.props.children` inside of App will be undefined.

```
1.  <Router>
2.    <Route path="/" component={App}>
3.      <IndexRoute component={Home}/>
4.      <Route path="accounts" component={Accounts}/>
5.      <Route path="statements" component={Statements}/>
6.    </Route>
7.  </Router>
```

Index Route

10. index_route

```
1.  import { Router, Route, IndexRoute, browserHistory } from 'react-router';
2.  import { Welcome } from './Welcome';
3.  import { List } from './List';
4.  …
5.  <Route path="/" component={App}>
6.      <IndexRoute component={Welcome} />
7.      <Route path="posts" component={Posts}>
8.          <IndexRoute component={List} />
9.          <Route path="article" component={Article} />
10.     </Route>
11.     <Route path="about" component={About} />
12. </Route>
```

`IndexRoute` helps you display default component if there are no child components that match parent URL

URL Params

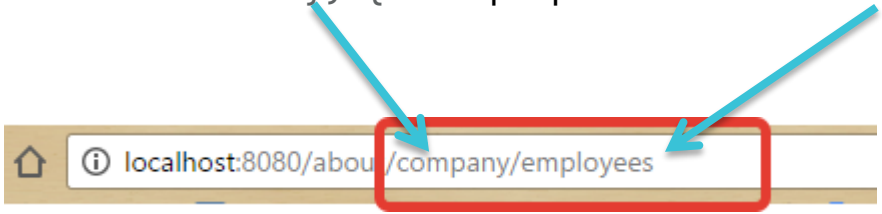
You can set params inside route path like that

```
<Route path="about/:tab/:section" component={About} />
```

Here `:tab` and `section` are placeholders for the values in URL. For example in URL like that `/about/places/offices` the last two parts are params: tab and section respectively

In `About` component you have access to these params via props using `routeParams`

```
<p>data: {this.props.routeParams.tab}, {this.props.routeParams.section}</p>
```



Query

Query goes after question mark in URL

It looks like key and value pair divided by `=` sign

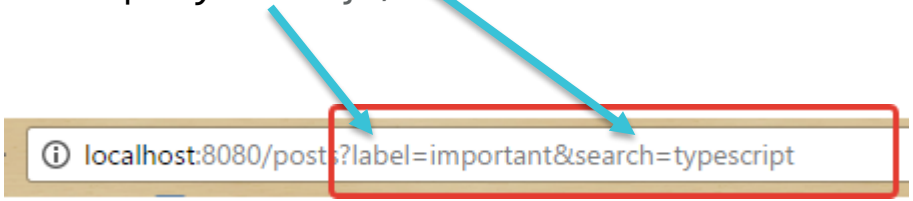
Several values are separated by `&` sign

There are two options how to pass query

```
1.  <Link to="/posts?search=typescript&label=important">Posts with filter</Link>
2.  <Link to={{ pathname: "/posts", query: { "search": "typescript", label: 'important' } }} >Posts with filter</Link>
```

You have access to query via `location` prop

```
<div>{this.props.location.query.search}</div>
<div>{this.props.location.query.label}</div>
```



Passing props to route

13. props_to_routes

You can pass additional props to route

```
<Route path="*" component={NotFound} code={404} />
```

They will be available inside component via `this.props.route[additional prop name]`

```
<p>Error {this.props.route.code}</p>
```


Advanced

Router API

14. router_api

- Router is available in `this.props.router` in route page component, and in another components it is placed in `this.context.router` (don't forget to set `contextTypes`)
- You can check if the route is active via `this.props.router.isActive(route, onlyActiveOnIndex)`
- Router has the same methods as `history`
- Inside `<Router />` you can set wrapper for all page components via `createElement`
- Also you can set `onUpdate` callback, it will be invoked every time when the route is changed

You can prevent a transition from happening or prompt the user before leaving a route with a leave hook.

```
this.props.router.setRouteLeaveHook(this.props.route, nextLocation => 'You will never leave us!')
```

Route API

15. route_api

- A `<Route />` is used to declaratively map routes to your application's component hierarchy.
- Props
  - `path` - the path used in the URL. If left undefined, the router will try to match the child routes.
  - `component` - a single component to be rendered when the route matches the URL. It can be rendered by the parent route component with `this.props.children`.
  - `components` - routes can define one or more named components as an object of `[name]: component` pairs. They can be rendered by the parent route component with `this.props[name]`.
- Events:
  - `onEnter` – invoked when user just entered the route
  - `onChange` – invoked whet the route is the same but query or params are changed
  - `onLeave` - invoked when user leaves the route

Lazy routes

16. lazy_routes

```
getChildRoutes(partialNextState, callback) {
  require.ensure([], function (require) {
    callback(null, [
      require('./routes/Announcements'),
      require('./routes/Assignments'),
      require('./routes/Grades'),
    ])
  })
}

getIndexRoute(partialNextState, callback) {
  require.ensure([], function (require) {
    callback(null, {
      component: require('./components/Index'),
    })
  })
}

getComponents(nextState, callback) {
  require.ensure([], function (require) {
    callback(null, require('./components/Course'))
  })
}
```

- It is possible to load components asynchronously by demand
- Routes may define `getChildRoutes`, `getIndexRoute`, and `getComponents` methods. These are asynchronous and only called when needed.
- Coupled with a smart code splitting tool like webpack, a once tiresome architecture is now simple and declarative.

Redirect

A `<Redirect />` sets up a redirect to another route in your application to maintain old URLs.

Props

- `from` - the path you want to redirect from, including dynamic segments.
- `to` - the path you want to redirect to.
- `query` - by default, the query parameters will just pass through but you can specify them if you need to.

```
1. <Route component={App}>
2.   <Route path="about/:userId" component={UserProfile} />
3.   {/* /profile/123 -> /about/123 */}
4.   <Redirect from="profile/:userId" to="about/:userId" />
5. </Route>
```

Index Redirects

17. redirect

Suppose your basic route configuration looks like:

```
1.  <Route path="/" component={App}>
2.    <Route path="welcome" component={Welcome} />
3.    <Route path="about" component={About} />
4.  </Route>
```

Suppose you want to redirect `/` to `/welcome`. To do this, you need to set up an index route that does the redirect. To do this, use the `<IndexRedirect />` component:

```
1.  <Route path="/" component={App}>
2.    <IndexRedirect to="/welcome" />
3.    <Route path="welcome" component={Welcome} />
4.    <Route path="about" component={About} />
5.  </Route>
```

Plain route

18. plain_route

- You can pass configuration via plain JavaScript object
- A plain JavaScript object route definition. `<Router />` turns JSX `<Route />` into these objects, but you can use them directly if you prefer.
  - childRoutes - An array of child routes, same as children in JSX route configs.
  - indexRoute - the index route. This is the same as specifying an <IndexRoute> child when using JSX route configs.

```
const routes = {
    path: '/',
    component: App,
    indexRoute: { component: Welcome },
    childRoutes: [
        {
            path: 'posts',
            component: Posts,
            indexRoute: { component: List },
            childRoutes: [
                { path: 'article/:name', component: Article },
                { path: 'story/:name', onEnter: (nextState, replace) => replace(`/posts/article/${nextState.params.name}`) },
            ]
        },
        {
            path: '*',
            component: NotFound
        }
    ]
};
```

Page transition animation

19. transition_animation

You can animate transition from one route no another using i.e. `react-addons-css-transition-group`

In the component that renders nested routes you wrap with animation container and clone children with different key

```
<ReactCSSTransitionGroup
    transitionName="page"
    transitionEnterTimeout={1000}
    transitionLeaveTimeout={1000}
    >
    {React.cloneElement(this.props.children, {
        key: location.pathname
    })}
</ReactCSSTransitionGroup>
```

Navigate to a route programmatically

20. navigate_programmatically

For navigation you can use

- History module itself
- `browserHistory` or similar from `react-router`
- `this.context.router.push(path)` or `this.context.router.push(path)` don't forget to set `contextTypes`

To create path with query you can use `this.context.router.createPath({ pathname, query })`

```
1.  ..
2.  import { browserHistory as history } from 'react-router';
3.  ..
4.  <button onClick={() => history.push('/')}>Home</button>
5.  <button onClick={() => this.context.router.push('/about/company/employees')}>About</button>
6.  <button onClick={() => history.replace('/posts')}>Posts</button>
7.  <button onClick={() => history.push('/posts?search=typescript&label=important')}>Posts with filter</button>
```

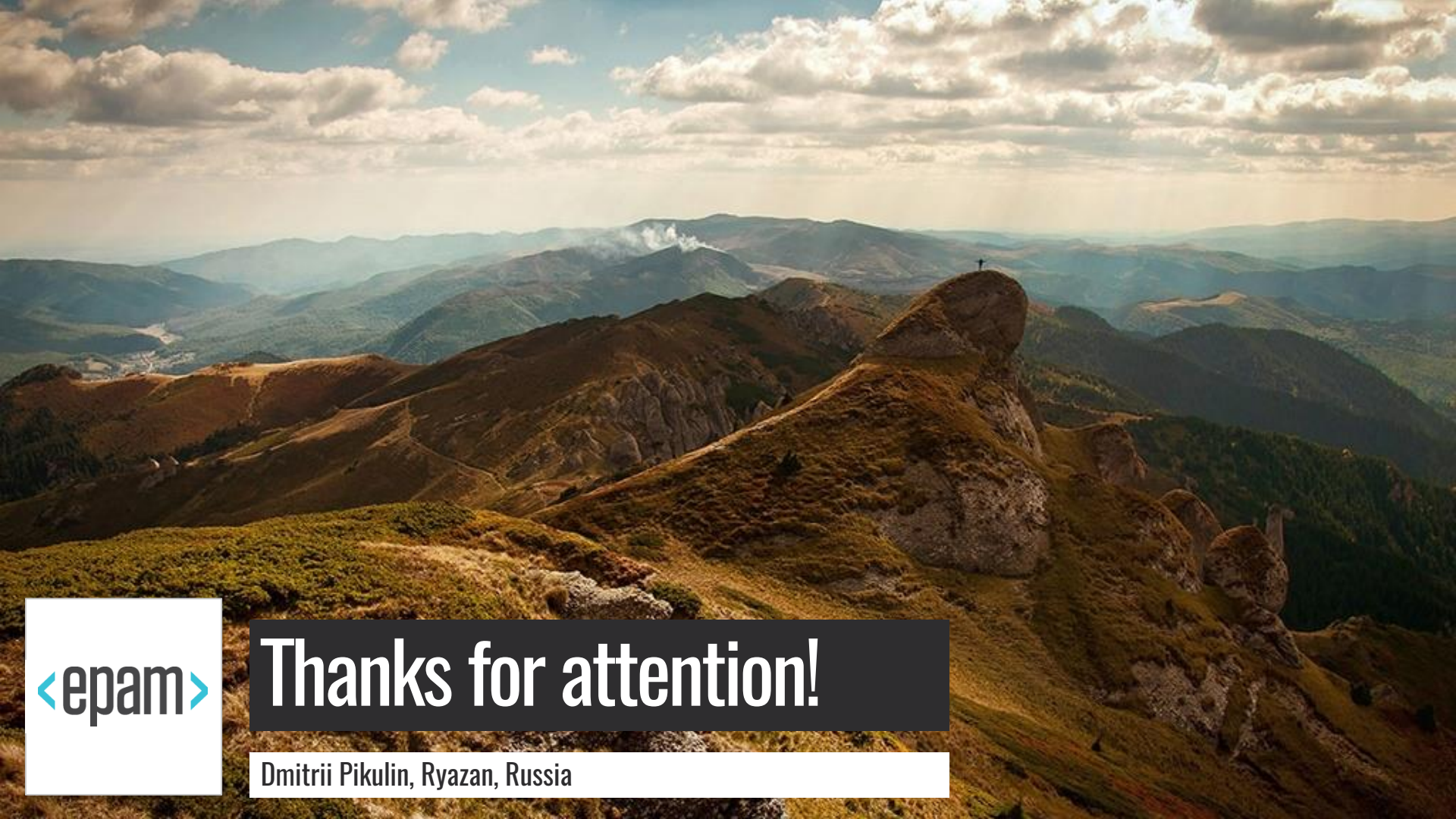
Useful links

ENG

- https://github.com/ReactTraining/react-router
- https://css-tricks.com/learning-react-router/
- http://redux.js.org/docs/advanced/UsageWithReactRouter.html
- https://medium.com/@dabit3/beginner-s-guide-to-react-router-53094349669#.oxlxc97te

RU

- https://habrahabr.ru/post/282369/
- http://prgssr.ru/development/pogruzhenie-v-react-router.html
- https://maxfarseer.gitbooks.io/react-router-course-ru/content/


‹epam›
Thanks for attention!
Dmitrii Pikulin, Ryazan, Russia