TUTORIAL

A Sneak Peek at React Router v6

React

By William Le

Published on March 17, 2020 168.6k

While this tutorial has content that we believe is of great benefit to our community, we have not yet tested or edited it to ensure you have an error-free learning experience. It's on our list, and we're working on it! You can help us out by using the "report an issue" button at the bottom of the tutorial.

At the time of this writing, React Router v6 is still in alpha, but the time is about right to start playing with it and exploring what's to come. This guide will give you a peek at the new features/changes!

As you may know, the lead maintainers forked the React Router project to create a lightweight alternative called Reach Router in early 2018.

During this time, both libraries grew, however it seems that active development for Reach Router will stop, and will be merged into the upcoming React Router v6 🛣️

With the release coming soon, here's a sneak peek of what's coming!

Jump to...

- `<Switch>` is becoming `<Routes>`
- Big changes with `<Route>`
- Nested Routes are simpler
- `useNavigate` instead of `useHistory`
- From 20kb to 8kb

<Switch> is becoming <Routes>

This top-level component is going to be renamed. However, its functionality is mostly remaining the same.

```
// v5
import {
  BrowserRouter,
  Switch,
  Route
} from 'react-router-dom';

function App() {
  return (
    <BrowserRouter>
      <Switch>
        <Route exact path="/"><Home /></Route>
        <Route path="/profile"><Profile /></Route>
      </Switch>
    </BrowserRouter>
  );
}
```

Just drop `<Routes>` in there:

```
// v6
import {
  BrowserRouter,
  Routes,
  Route
} from 'react-router-dom';

function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="profile/*" element={<Profile />} />
      </Routes>
    </BrowserRouter>
  );
}
```

Big Changes with <Route>

In v6 the `<Route>` component is receiving the biggest overhaul. Fortunately, these new changes will actually be making it simpler to use!

The `component/render` prop will be substituted for the `element` prop:

```
import Profile from './Profile';

// v5
<Route path=":userId" component={Profile} />
<Route
  path=":userId"
  render={routeProps => (
    <Profile routeProps={routeProps} animate={true} />
  )}
/>

// v6
<Route path=":userId" element={<Profile />} />
<Route path=":userId" element={<Profile animate={true} />} />
```

If you noticed, in v6 it’s much easier to pass props now. This has negated the use of the `render` prop in v5.

Nested Routes are Simpler

Nested routes in v5 had to be very explicitly defined. This required including a lot of string-matching logic into these components. See <Profile>:

```
// v5
import {
  BrowserRouter,
  Switch,
  Route,
  Link,
  useRouteMatch
} from 'react-router-dom';

function App() {
  return (
    <BrowserRouter>
      <Switch>
        <Route exact path="/" component={Home} />
        <Route path="/profile" component={Profile} />
      </Switch>
    </BrowserRouter>
  );
}

function Profile() {

  let match = useRouteMatch();

  return (
    <div>
      <nav>
        <Link to={`${match.url}/me`}>My Profile</Link>
      </nav>

      <Switch>
        <Route path={`${match.path}/me`}>
          <MyProfile />
        </Route>
        <Route path={`${match.path}/:id`}>
          <OthersProfile />
        </Route>
      </Switch>
    </div>
  );
}
```

In v6, you can remove the string-matching logic. There isn’t any need for `useRouteMatch()` either! The result is pleasantly minimal:

```
// v6
import {
  BrowserRouter,
  Routes,
  Route,
  Link,
  Outlet
} from 'react-router-dom';

// Approach #1
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="profile/*" element={<Profile/>} />
      </Routes>
    </BrowserRouter>
  );
}

function Profile() {
  return (
    <div>
      <nav>
        <Link to="me">My Profile</Link>
      </nav>

      <Routes>
        <Route path="me" element={<MyProfile />} />
        <Route path=":id" element={<OthersProfile />} />
      </Routes>
    </div>
  );
}

// Approach #2
// You can also define all
// <Route> in a single place
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<Home />} />
        <Route path="profile" element={<Profile />}>
          <Route path=":id" element={<MyProfile />} />
          <Route path="me" element={<OthersProfile />} />
        </Route>
      </Routes>
    </BrowserRouter>
  );
}

function Profile() {
  return (
    <div>
      <nav>
        <Link to="me">My Profile</Link>
      </nav>

      <Outlet />
    </div>
  )
}
```

Note: the `<Outlet>` component is used like `{this.props.children}` in React Router v6. This was a very popular feature from Reach Router!

useNavigate Instead of useHistory

Sometimes you'll want to programmatically navigate. For example, after a user submits a form and they need to be redirected to a confirmation page. This is the `useHistory` library in v5, which has been renamed to `useNavigate` in v6:

```
// v5
import { useHistory } from 'react-router-dom';

function MyButton() {
  let history = useHistory();
  function handleClick() {
    history.push('/home');
  };
  return <button onClick={handleClick}>Submit</button>;
};
```

Now `history.push()` will be replaced with `navigate()`:

```
// v6
import { useNavigate } from 'react-router-dom';

function MyButton() {
  let navigate = useNavigate();
  function handleClick() {
    navigate('/home');
  };
  return <button onClick={handleClick}>Submit</button>;
};
```

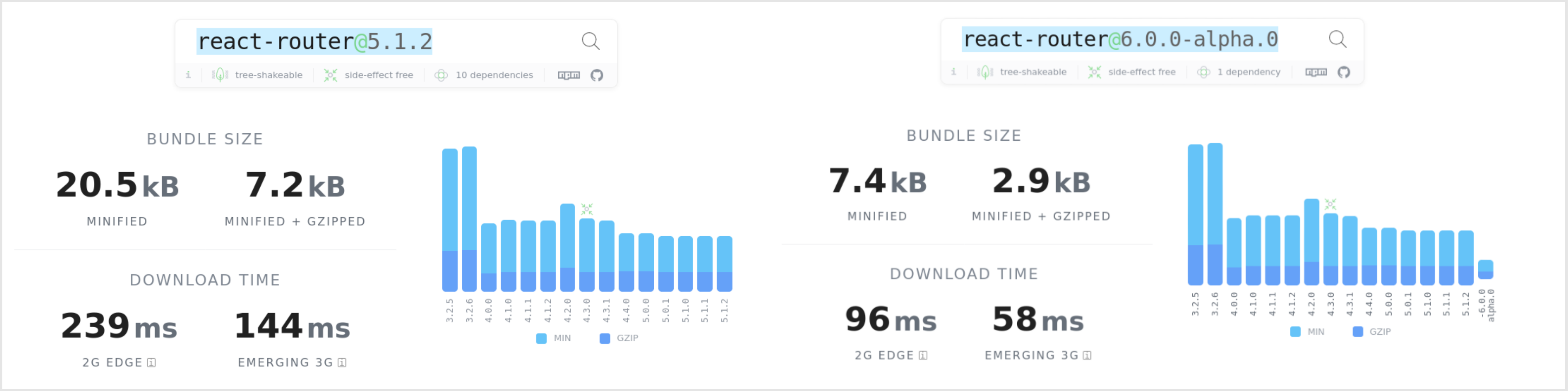
In some cases, you'll want to replace an URL in the browser history instead of pushing a new URL. This has slightly changed with v6:

```
// v5
history.push('/home');
history.replace('/home');

// v6
navigate('/home');
navigate('/home', {replace: true});
```

From 20kb to 8kb

With all of these changes, you'd expect the bundle size to grow, but it's actually reduced by half! The minified bundle of v5 was ~20kb, and v6 is only ~8kb.


react-router@5.1.2

tree-shakeable | side-effect free | 10 dependencies

BUNDLE SIZE

20.5 kB MINIFIED

7.2 kB MINIFIED + GZIPPED

DOWNLOAD TIME

239 ms 2G EDGE

144 ms EMERGING 3G

react-router@6.0.0-alpha.0

tree-shakeable | side-effect free | 1 dependency

BUNDLE SIZE

7.4 kB MINIFIED

2.9 kB MINIFIED + GZIPPED

DOWNLOAD TIME

96 ms 2G EDGE

58 ms EMERGING 3G

Bundle sizes are calculated using the BundlePhobia tool.

Conclusion

I'm pretty excited about the release of React Router v6. Hopefully this article has given you an idea of what to expect when it releases (which should be soon)! You can read more about React Router v6 in the latest release notes 📝

For a complete list of new features, see the official React Router v6 migration guide

Was this helpful? Yes No

Report an issue

About the authors

William Le

is a Community author on DigitalOcean.

Still looking for an answer?

Ask a question

Search for more help

RELATED

Join the DigitalOcean Community

Join 1M+ other developers and:

- Get help and share knowledge in Q&A
- Subscribe to topics of interest
- Get courses & tools that help you grow as a developer or small business owner

Join Now

Getting Started With React

Tutorial

How To Code in React.js eBook

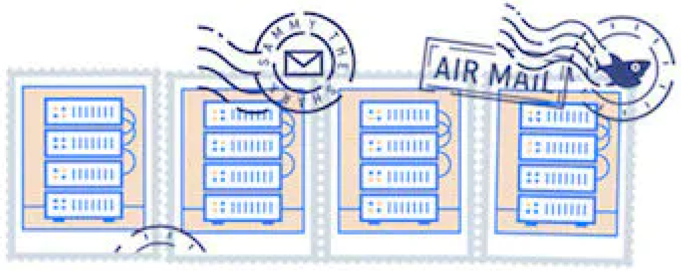
Tutorial

Comments

0 Comments

Leave a comment...

Sign In to Comment



GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.

HOLLIE'S HUB FOR GOOD

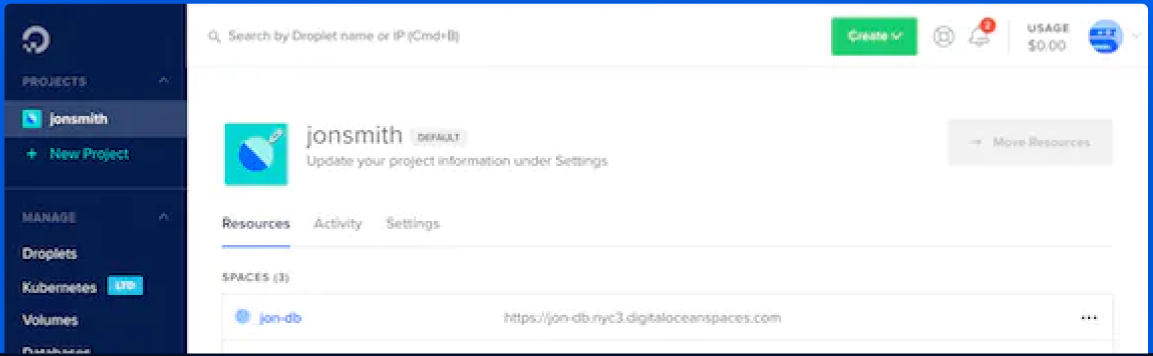
Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

BECOME A CONTRIBUTOR

You get paid; we donate to tech nonprofits.

Featured on Community Kubernetes Course Learn Python 3 Machine Learning in Python Getting started with Go Intro to Kubernetes

DigitalOcean Products Virtual Machines Managed Databases Managed Kubernetes Block Storage Object Storage Marketplace VPC Load Balancers



Update your project information under Settings

https://jon-db.nyc3.digitaloceanspaces.com

Company

- About
- Leadership
- Blog
- Careers
- Partners
- Referral Program
- Press
- Legal
- Security & Trust Center

Products

- Pricing
- Products Overview
- Droplets
- Kubernetes
- Managed Databases
- Spaces
- Marketplace
- Load Balancers
- Block Storage
- API Documentation
- Documentation
- Release Notes

Community

- Tutorials
- Q&A
- Tools and Integrations
- Tags
- Write for DigitalOcean
- Presentation Grants
- Hatch Startup Program
- Shop Swag
- Research Program
- Open Source
- Code of Conduct