

React Fundamentals

Module – React Router



Peter Kassenaar –
info@kassenaar.com



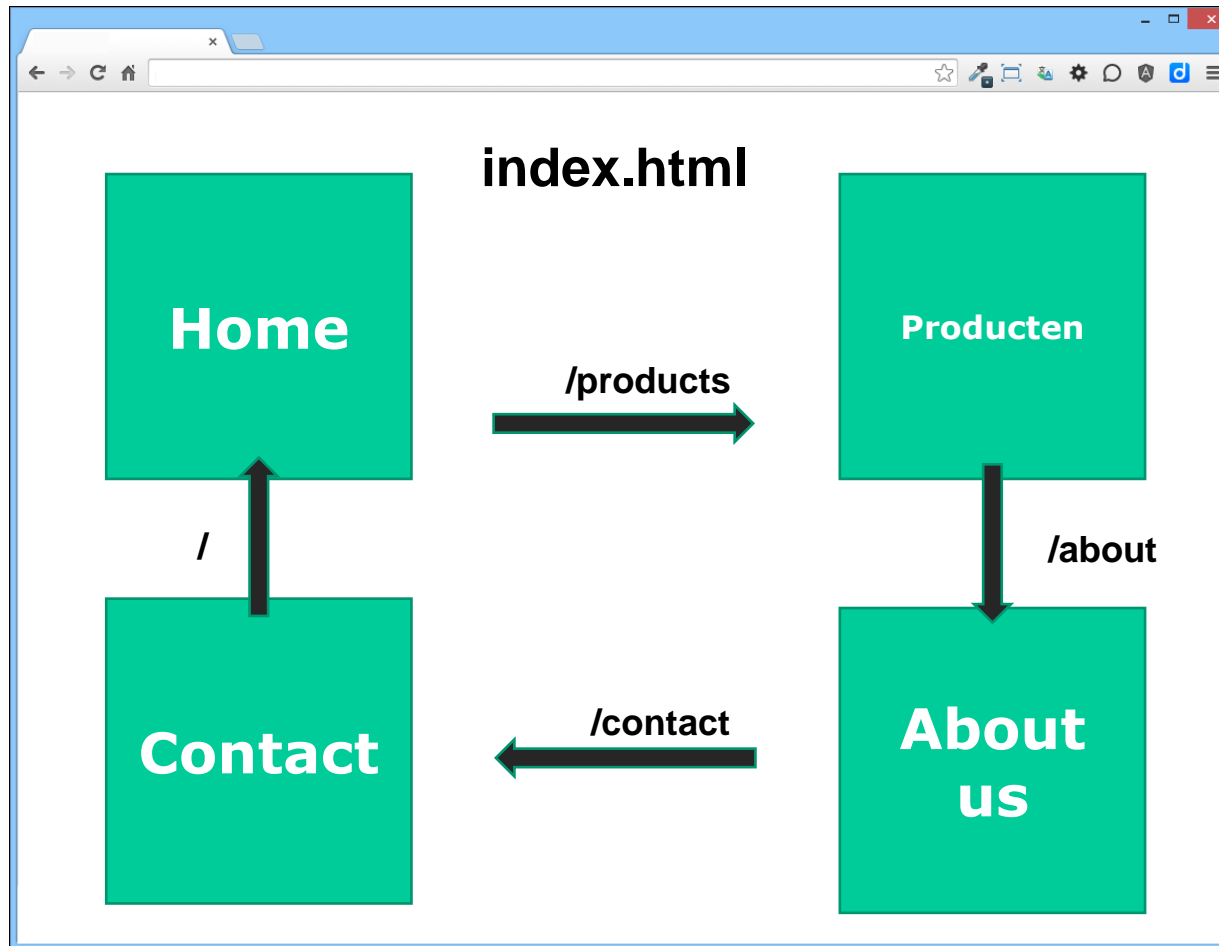
Routing in your application

Reflecting the state of your application in the URL

*"React Router is the **standard routing library** for React. React Router keeps your UI in sync with the URL. It has a simple API with powerful features like lazy loading, dynamic route matching and location transition."*

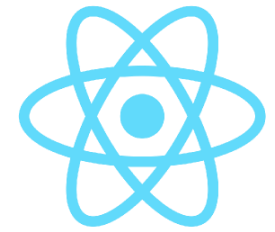
<https://www.freecodecamp.org/news/beginners-guide-to-react-router-4-8959ceb3ad58/>

Routing architecture and goal



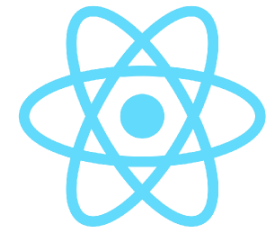
- Make use of SPA principle
- Making deep links possible

Routing capabilities / characteristics

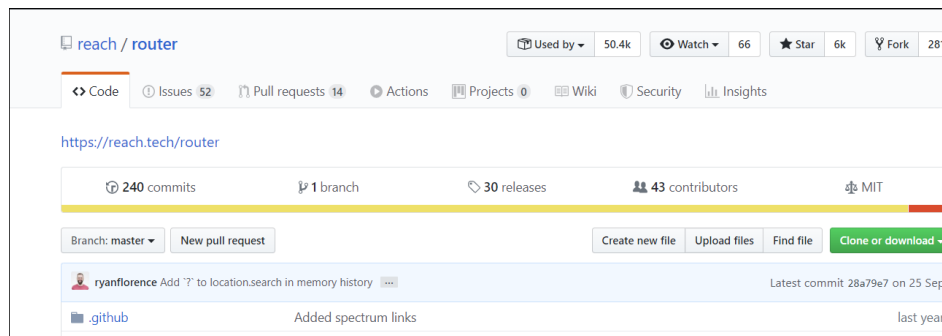


- **Update** URL when changing to routes/components
- **Nested** routes/view mapping
- **Modular**, component based router configuration
- Route **params**, query, wildcards
- View transition **effects**
- Link with automatic active CSS classes to denote **active route**
- HTML5 **history mode**
- Extensive **API**

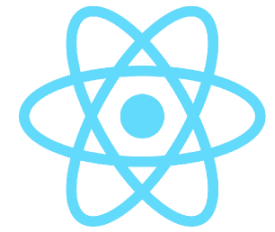
React Router



- Most popular choice: `react-router`
 - Created and maintained by team at reacttraining.com
- Also popular: Reach Router
 - Is now integrated with `react-router`
 - <https://github.com/reach/router>
 - Status – **don't use** for newer projects.

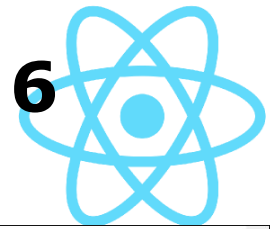








Version numbers






- React Router v3 – Hardly used anymore
- React Router v4 – Still very popular
 - used a lot
- React Router v5 – ‘intermediate’ version
- React Router v6 – Current version
 - use in new projects






Background info – new in React Router 6





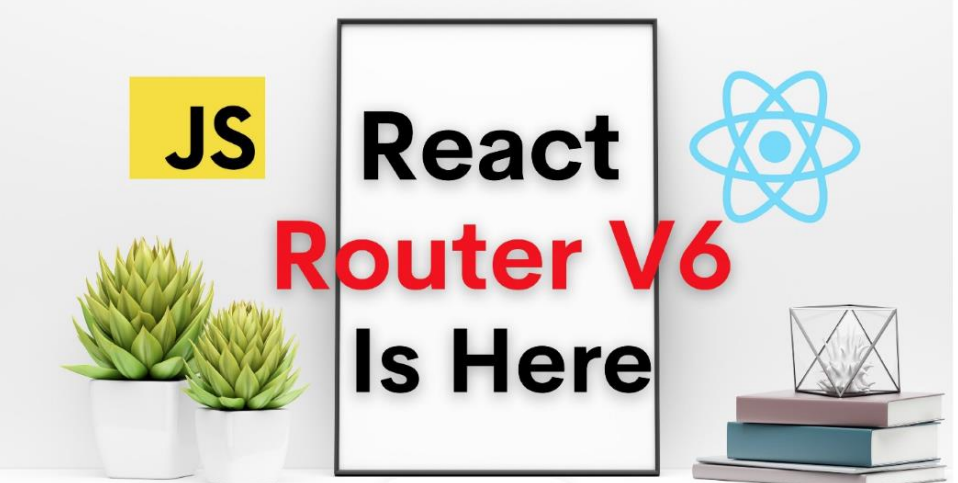
Published in Enlear Academy



**Piumi Liyana Gunawardhana**
Nov 11, 2021 · 5 min read ·  Listen








What's New in React Router 6?


A Quick Overview of React Router v6



 325 |  1

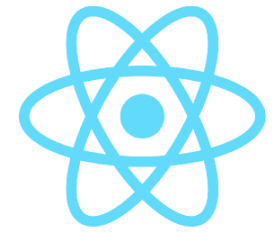
**Piumi Liyana Gunawardhana**
283 Followers
Software Engineer | Technical Writer | BSc. IT(Hons)
University of Moratuwa — Faculty of IT
 

More from Medium
 Sachin chaur... in JavaScript in Plain ...
5 Awesome Libraries To Use In Your Next React Project
 Abdolrahman Farshgar
Learn React Router V6 with a simple example!

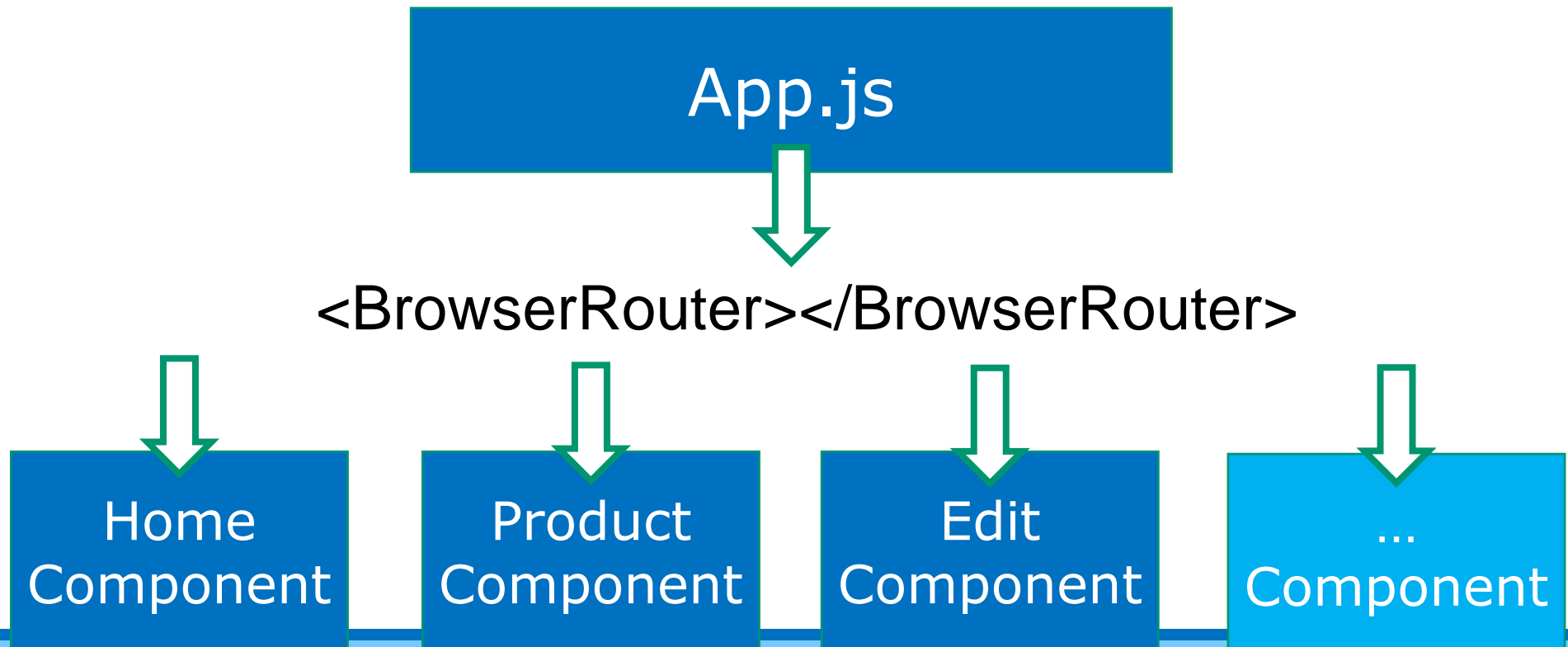


<https://enlear.academy/whats-new-in-react-router-6-e34e451e5285>

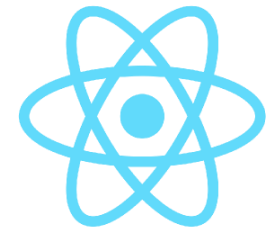
Routing – every route is a Component



- `App.js` gets a main menu (typically in its own component :-)
- Components are injected in `<BrowserRouter></BrowserRouter>`

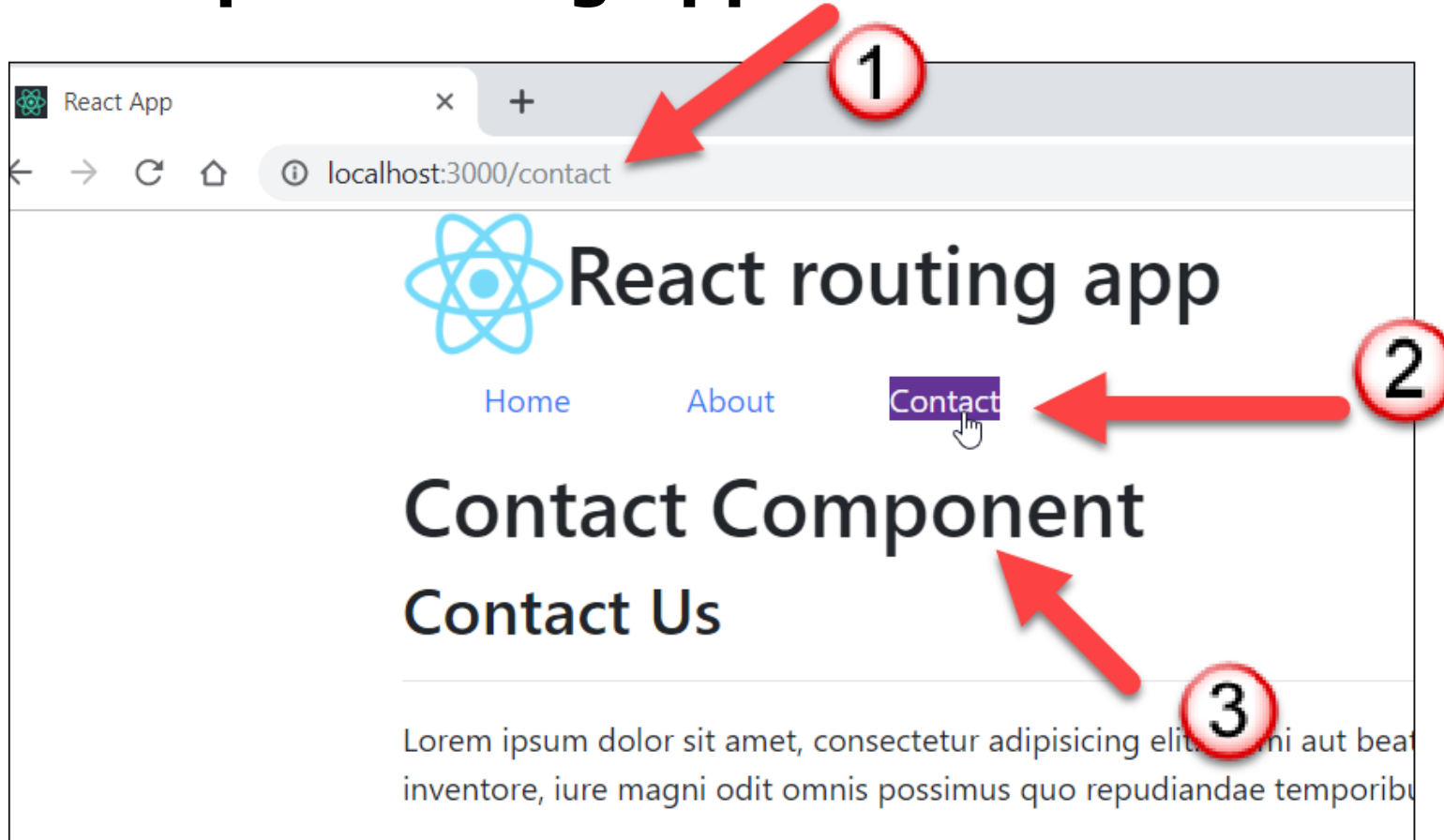
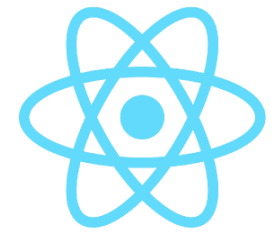


Routing in Components

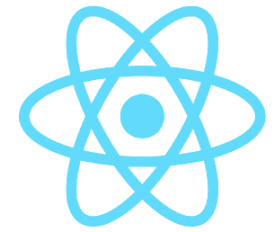


- Components can have **their own** router configs
 - Child routes
 - Nested routes
- They are loaded at runtime, when the component is loaded.
- **No central routing configuration** table like in Angular and Vue

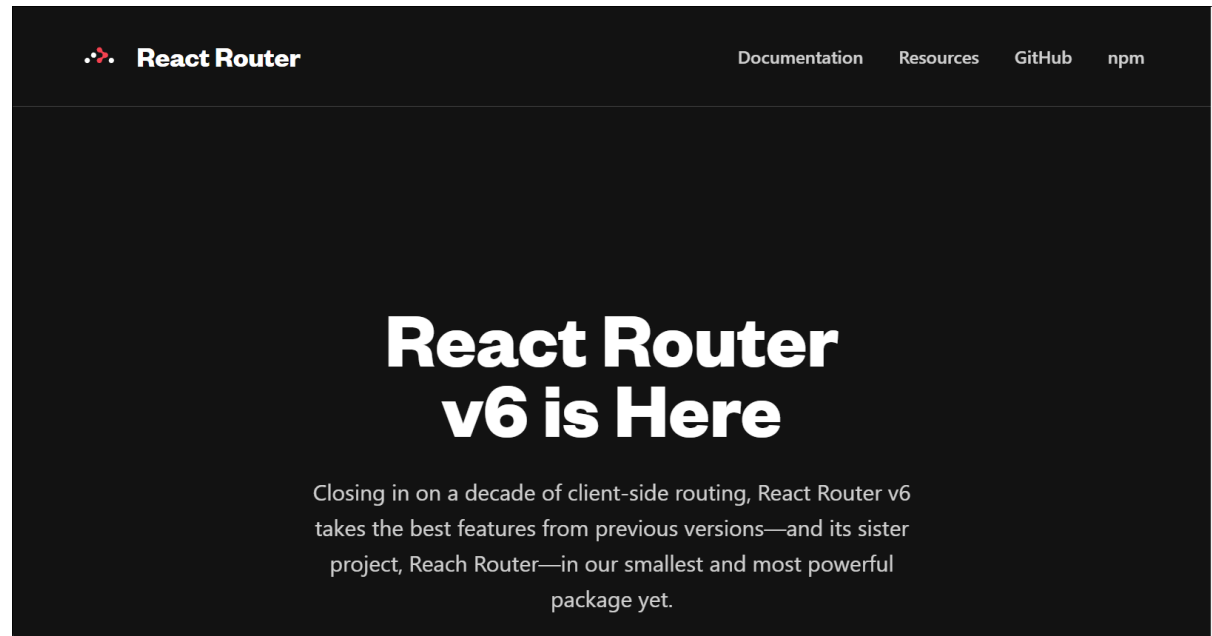
What are we going to build – a simple routing app for starters



Routing Versions

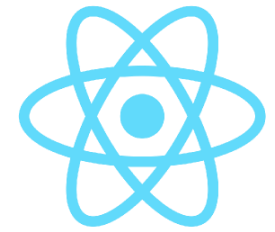


- Previously: React Router 4, React Router 5
 - Mainly used in **existing projects**
 - Some breaking changes with v6.
- Current version: React Router 6



<https://reactrouter.com/>

Steps in creating routing for your app

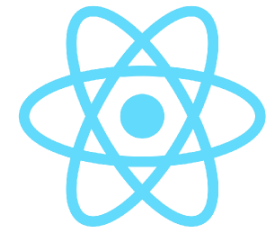


1. Create a new app, or start from existing app
2. Install `react-router`
3. Update `App.js` or `index.js` with `<BrowserRouter>` to use routes
4. Create the main navigation
5. Create components, to be shown inside the routes
6. Create the basic routes

*It is React, so **everything** is a component. Including*

*`<MainNavigation />` **and** `<Routes />`*

2. Install the router



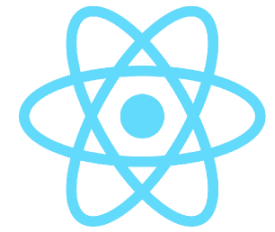
- We're using a new project, created with CRA
- Install the correct version

```
npm install react-router-dom@6
```

```
PS C:\Users\Gebruiker\Desktop\router-tutorial> npm install react-router-dom@6
npm WARN @apideck/better-ajv-errors@0.3.3 requires a peer of ajv@>= 6.12.0 but none is installed. You must install peer dependencies yourself.
npm WARN fork-ts-checker-webpack-plugin@6.5.2 requires a peer of typescript@>= 2.7 but none is installed. You must install peer dependencies yourself.
npm WARN tsutils@3.21.0 requires a peer of typescript@>=2.8.0 || >= 3.2.0-dev || >= 3.3.0 || >= 3.6.0-beta || >= 3.7.0-dev || >= 3.7.0-beta but none is installed. You must install peer dependencies yourself.
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: v
"arch":"x64"})

+ react-router-dom@6.3.0
added 3 packages from 1 contributor and audited 1430 packages in 5.081s
```

3. Use `<BrowserRouter>` element

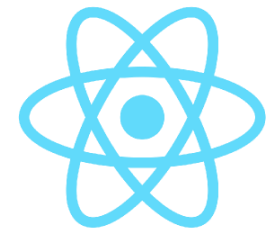


- Update `index.js` (or `App.js`) to use the routes.
- Note the usage of `<BrowserRouter>`
 - This is the *top level component* for every routing app
 - Remember to import from the correct libraries

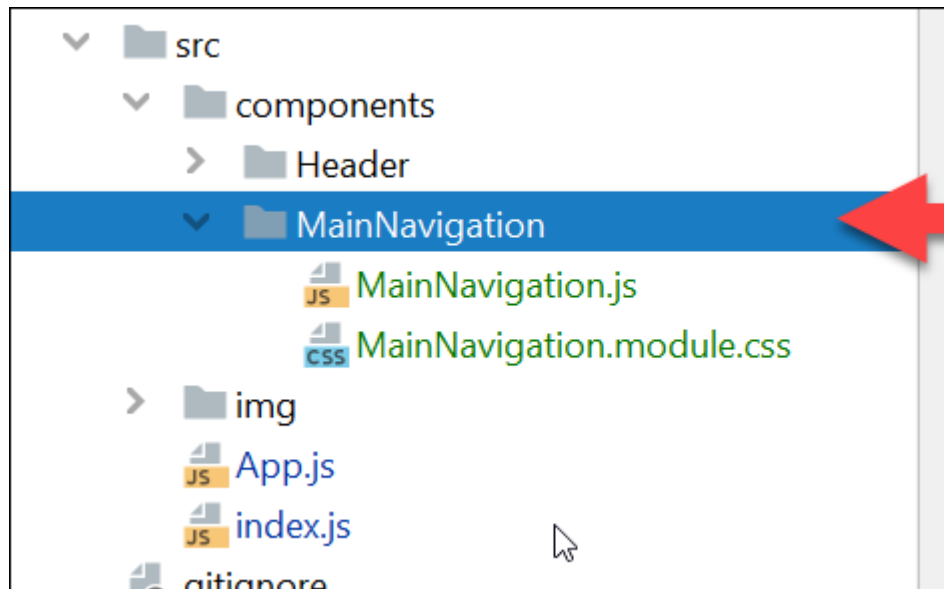
```
// index.js
// routing stuff
import {BrowserRouter} from "react-router-dom";

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <BrowserRouter>
    <App/>
  </BrowserRouter>
);
```

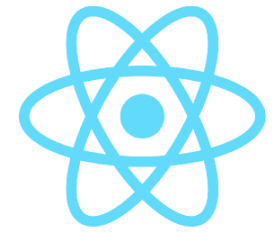
4. Create the main navigation



- Choice: single component holding the main navigation
 - Also valid: having the navigation inside a component for nested/child routes
- Optional: having a CSS module for navigation styles



Using the `<Link />` element



```
import React, {Component} from 'react';
import {Link} from "react-router-dom";

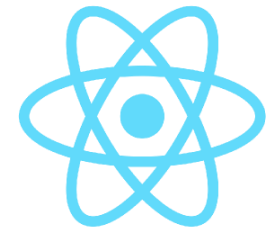
class MainNavigation extends Component {
  render() {
    return (
      <div>
        <ul>
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/about">About</Link>
          </li>
          <li>
            <Link to="/contact">Contact</Link>
          </li>
        </ul>
      </div>
    );
  }
}

export default MainNavigation;
```

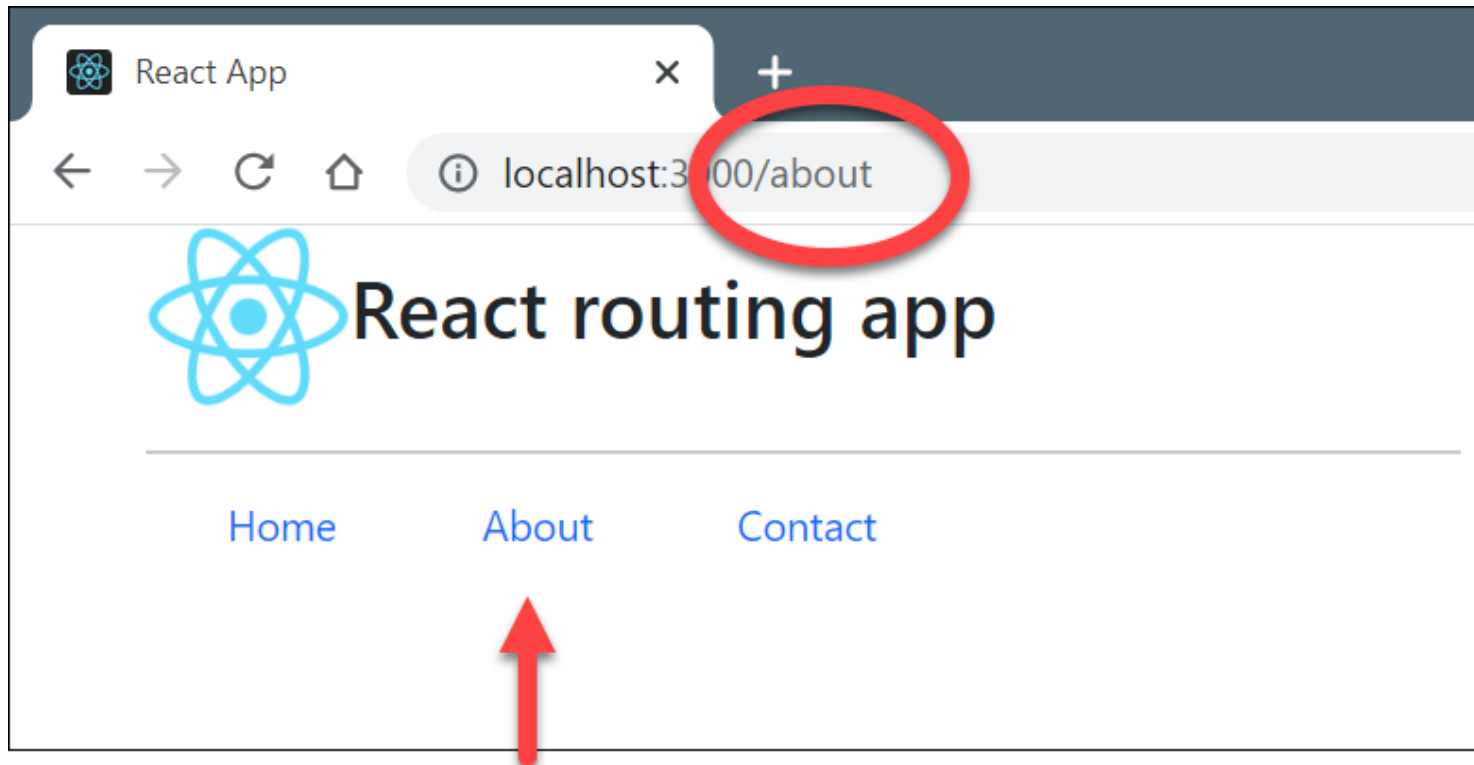


`<Link />` to navigate to routes

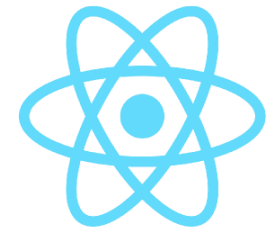
Routing now works!



No components are show yet, but URL is updated



5. Create Components



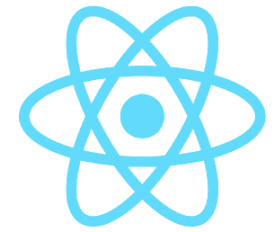
- We are using simple components w/ some text

```
class Home extends Component {  
  render() {  
    return (  
      <div>  
        <h1>Home Component</h1>  
        ...  
      </div>  
    )  
  }  
}
```

```
class About extends Component {  
  render() {  
    return (  
      <div>  
        <h1>About Component</h1>  
        <div>Our Mission</div>  
        ...  
      </div>  
    )  
  }  
}
```

```
class Contact extends Component {  
  render() {  
    return (  
      <div>  
        <h1>Contact Component</h1>  
        <div>Contact us:</div>  
        ...  
      </div>  
    )  
  }  
}
```

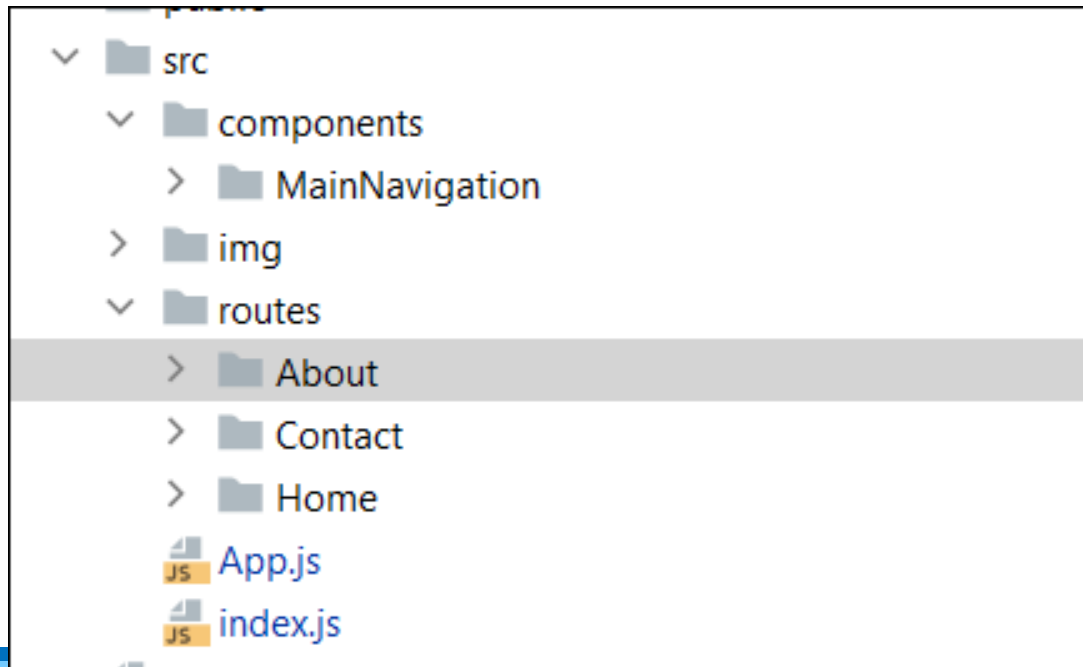
Directory Structure



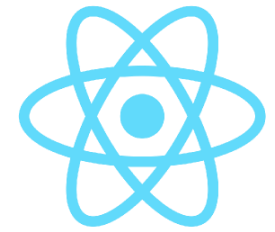
You can place files wherever you want.

We choose a folder called `/routes` here

you will also see `/pages`, `/views`, `/partials`, etc



6. Create basic routes



- Tell React Router **which component** is loaded at **which Route**.
- Use the `<Routes />` and `<Route />` Element from `react-router-dom`
- In React Router 6, Always nest a `<Route />` inside a `<Routes />` element

```
// import routing stuff  
import {Routes, Route} from "react-router-dom";
```

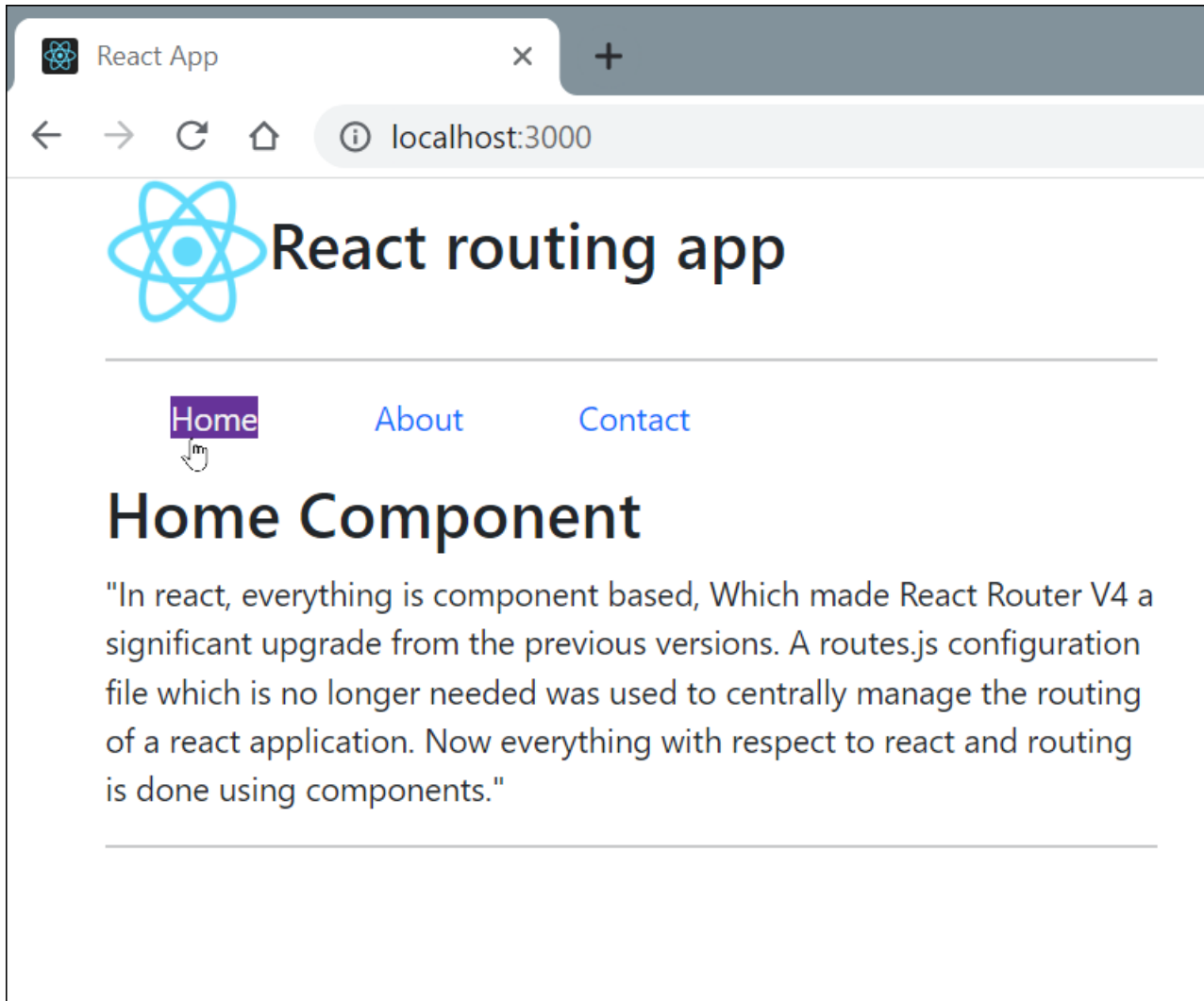
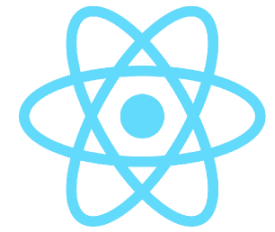
Imports

```
// components to be shown inside the router  
import Home from "../routes/Home/Home";  
import About from "../routes/About/About";  
import Contact from "../routes/Contact/Contact";
```

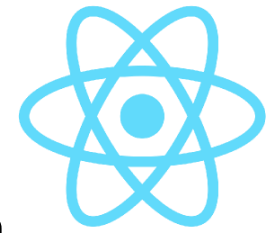
```
function App() {  
  return (  
    <div className="container">  
      (...)  
      <Routes>  
        <Route path="/" element={<Home/>}/>  
        <Route path="/about" element={<About/>}/>  
        <Route path="/contact" element={<Contact/>}/>  
      </Routes>  
    </div>  
  );  
}  
  
export default App;
```

Valid routes and
components to
load

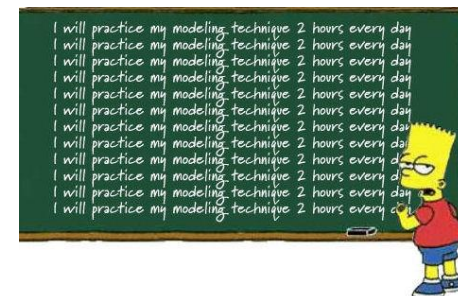
Result



Workshop



- 1. Use your own app OR create a new app from scratch
- Add routing to it. Make sure you use `v6`.
- Divide the components over different routes
- Maybe you are passing `props`. You can do that as usual
- 2. OR: Work from the example project
- Add a new component (`<Products />`) to it
- Add a `<Route>` and a `<Link>` to that new component, make sure it is accessible from the `<MainNavigation>`
- Example `../600-routing-basics`

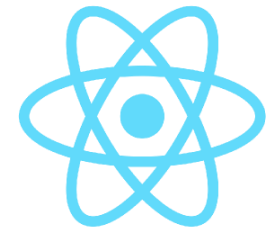




Styling the active link

Emphasizing the current route in the UI, based on a URL match


Special CSS class names



- When using `<NavLink>` instead of `<Link>`, we can assign a special class `activeRoute` to the element
- You can write CSS for that
 - Global CSS or a CSS module
 - For instance a class `.activeRoute` (but you can name it any way you like)

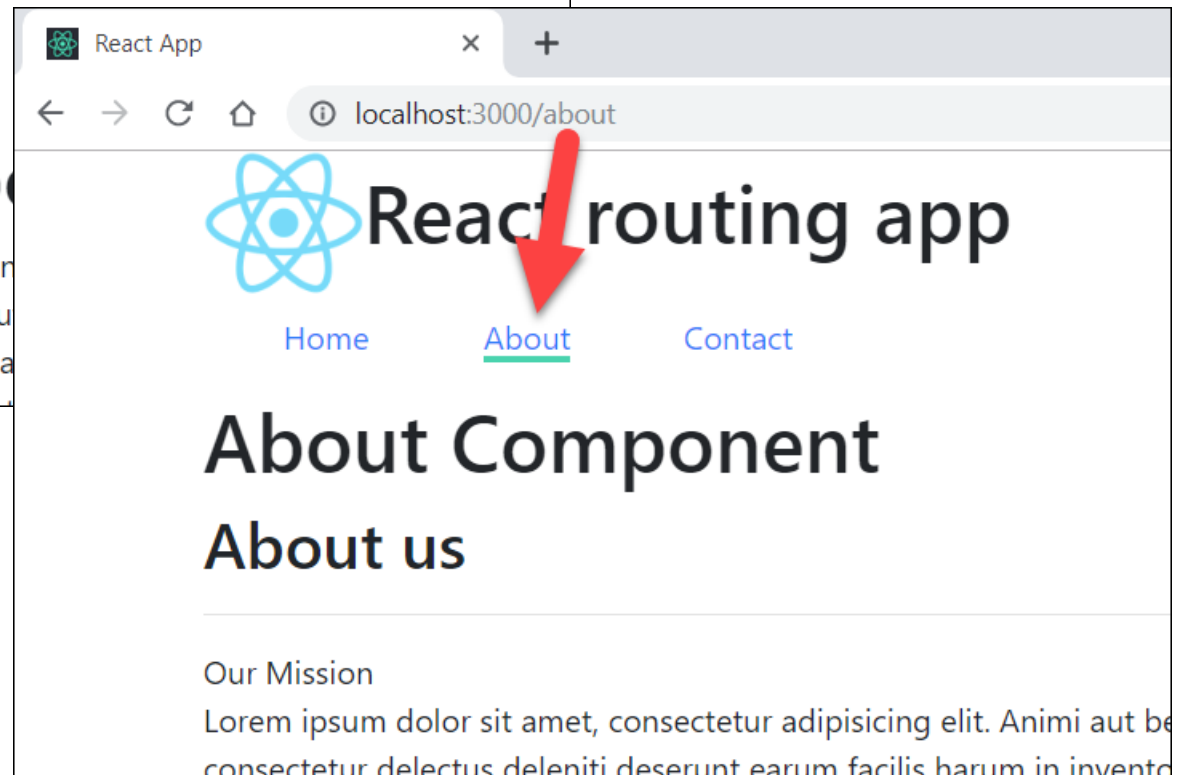
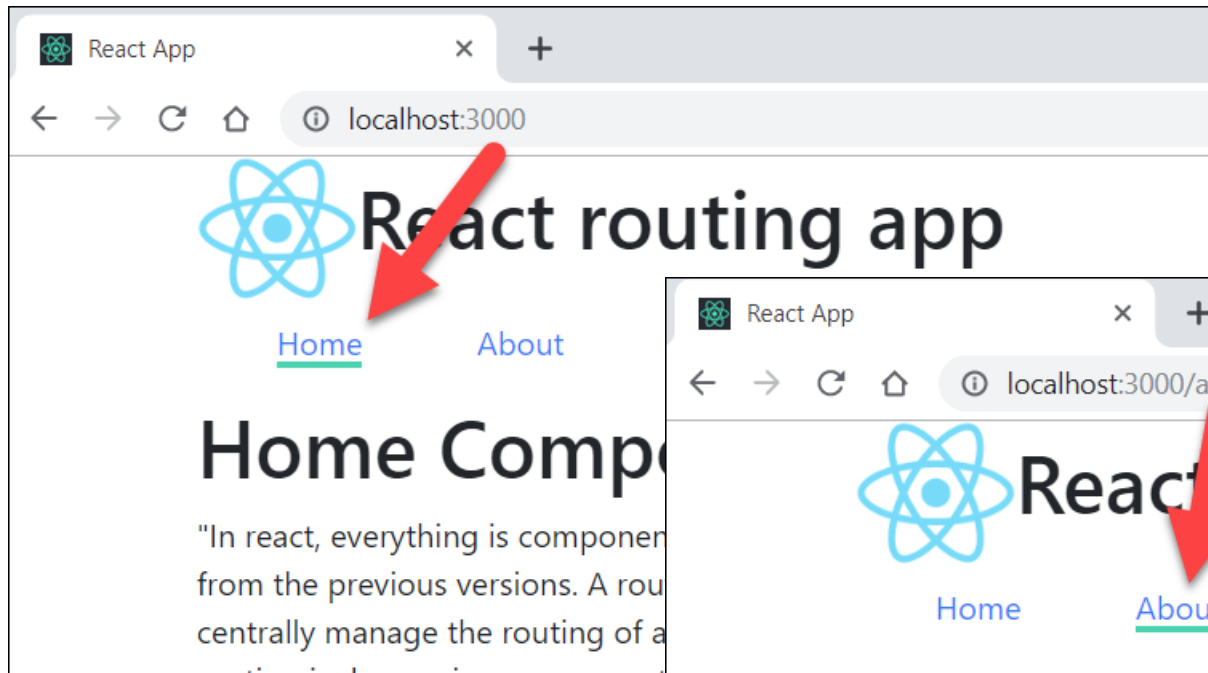
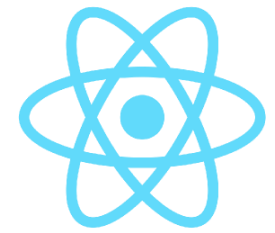
```
.activeRoute {  
  border-bottom: 3px solid #09d3ac;  
}  
.inactiveRoute {  
  border-bottom: 3px solid transparent;  
}
```

Update Main navigation, use <NavLink>

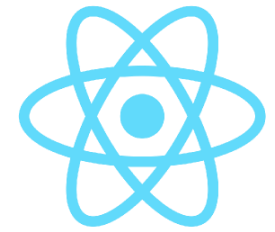


```
<nav>
  { /*Start navigation*/ }
  <ul>
    <li>
      <NavLink to="/"
        className={({ isActive }) =>
          (isActive ? styles.activeRoute : styles.inactiveRoute)}
        >Home</NavLink>
    </li>
    (...)
  </ul>
  { /*End navigation*/ }
</nav>
```

Result



Alternative approach



- Create a custom `<Link>` component, use `useMatch()` and `useResolvedPath()` hooks
- React Router/Stackblitz example

Custom Link Example

This example demonstrates how to make a custom `<Link>` component to render something different when the link is "active" using the `useMatch()` and `useResolvedPath()` hooks.

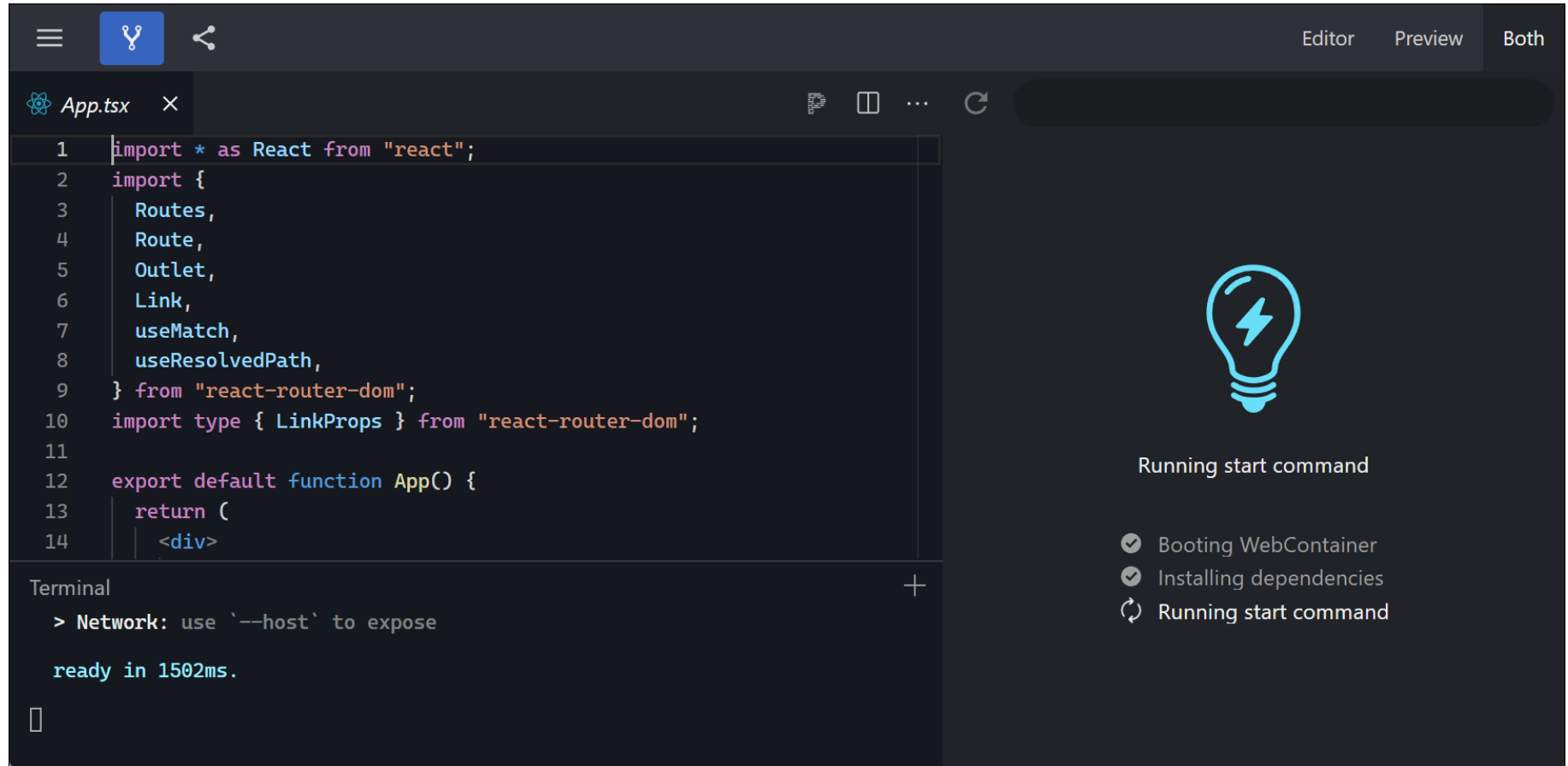
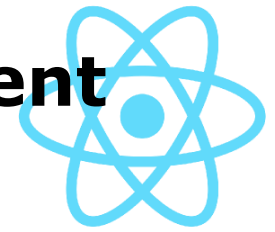
Preview

Open this example on [StackBlitz](#):



Open in StackBlitz

Example, creating `<CustomLink />` element



The screenshot shows a code editor interface with a dark theme. The top bar includes a menu icon, a logo, and a share icon. The right side has tabs for 'Editor', 'Preview', and 'Both'. The main editor area displays the `App.tsx` file with the following code:

```
1 import * as React from "react";
2 import {
3   Routes,
4   Route,
5   Outlet,
6   Link,
7   useMatch,
8   useResolvedPath,
9 } from "react-router-dom";
10 import type { LinkProps } from "react-router-dom";
11
12 export default function App() {
13   return (
14     <div>
```

Below the code editor is a terminal window showing the output of a command:

```
> Network: use '--host' to expose
ready in 1502ms.
```

On the right side of the interface, there is a lightbulb icon and the text "Running start command". Below this, there is a list of tasks with checkboxes:

- ✓ Booting WebContainer
- ✓ Installing dependencies
- 🔄 Running start command

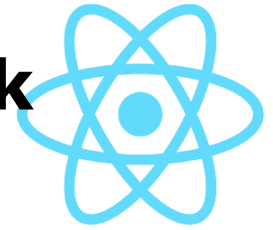
<https://stackblitz.com/github/remix-run/react-router/tree/main/examples/custom-link?file=src%2FApp.tsx>



Navigating from code

Activating another Route from JavaScript

React Router v6 has a `useNavigate` hook



- Navigate to other routes from code (NOT from `<Link>` or `<NavLink />`)
- Use function components
- `import useNavigate()` hook
- Call hook from a `<button>`, or `<a href>`.

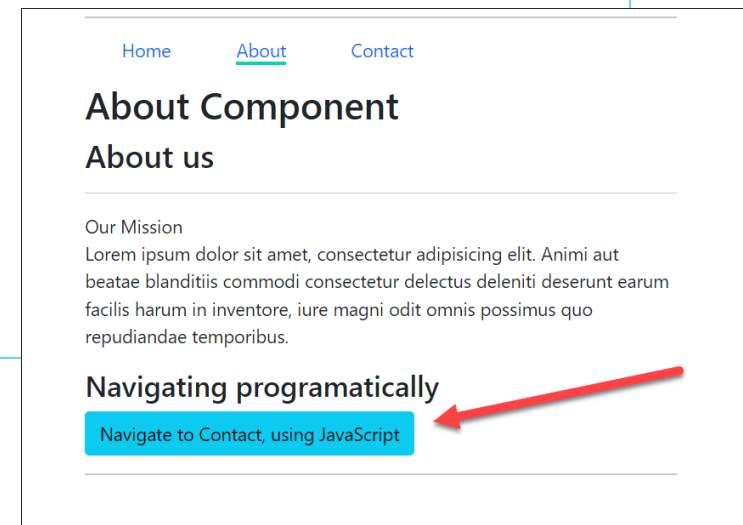

```

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

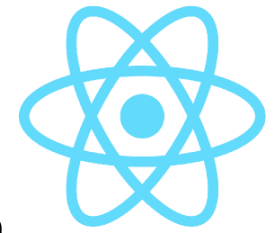
const About = () => {
  const navigate = useNavigate();

  return (
    <div>
      (...)
      <h3>Navigating programatically</h3>
      <button
        className="btn btn-info"
        onClick={() => navigate('/contact')}>
        Navigate to Contact, using JavaScript
      </button>
    </div>
  )
}
export default About

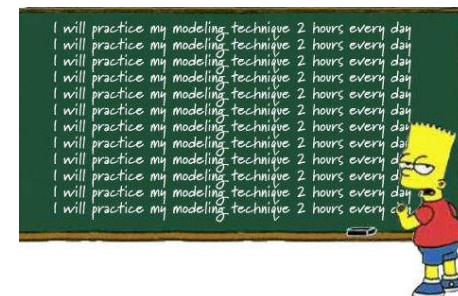
```



Workshop



- 1. Use your own app OR create a new app from scratch
- Create an 'active link' class and use it with `<NavLink />`
- Create a button and navigate from code using JavaScript
- 2. OR: Work from the example project
- Give the Active route class a different styling and make sure it works. Also inspect the rendered HTML
- Create a button on the `Contact` component to navigate to the Home page.
- Example `../610-active-link-and-code`

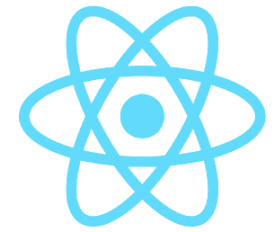




Using Route Parameters

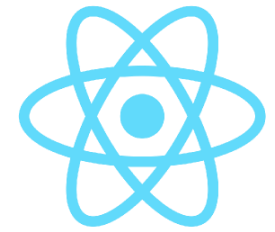
Passing detail parameters to the next view

Route Parameters



- Creating Master/Detail Views
- Select a `country|product|employee|etc` on the first page and pass the selection to the next page
- Steps:
 1. Update `<Link>` or `<NavLink>` to **send parameters**
 2. (**Optional**: create a `404/FileNotFound/NoMatch` component)
 3. Create a `<Link />` with **dynamic parameters**.
 4. Create/Update a `DetailComponent` to **receive parameters**
 5. Fetch Details and show them in the UI

0. Prepare Home component



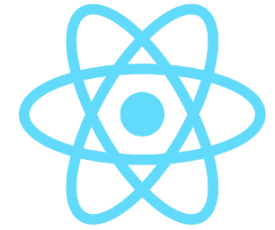
- Showing a list of countries
- Import `countryData.js` and prepare state as usual

```
import countryData from "../data/CountryData";

class Home extends Component {
  state = {
    countries: countryData.countries
  };
  ...
};
```

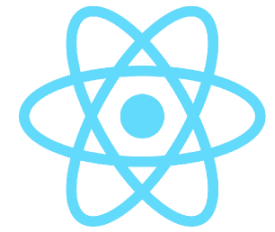
```
<ul className="list-group">
  {this.state.countries.map(country =>
    <li className="list-group-item"
      key={country.id}>
      {country.name}
    </li>
  )}
</ul>
```

1. Sending the parameters

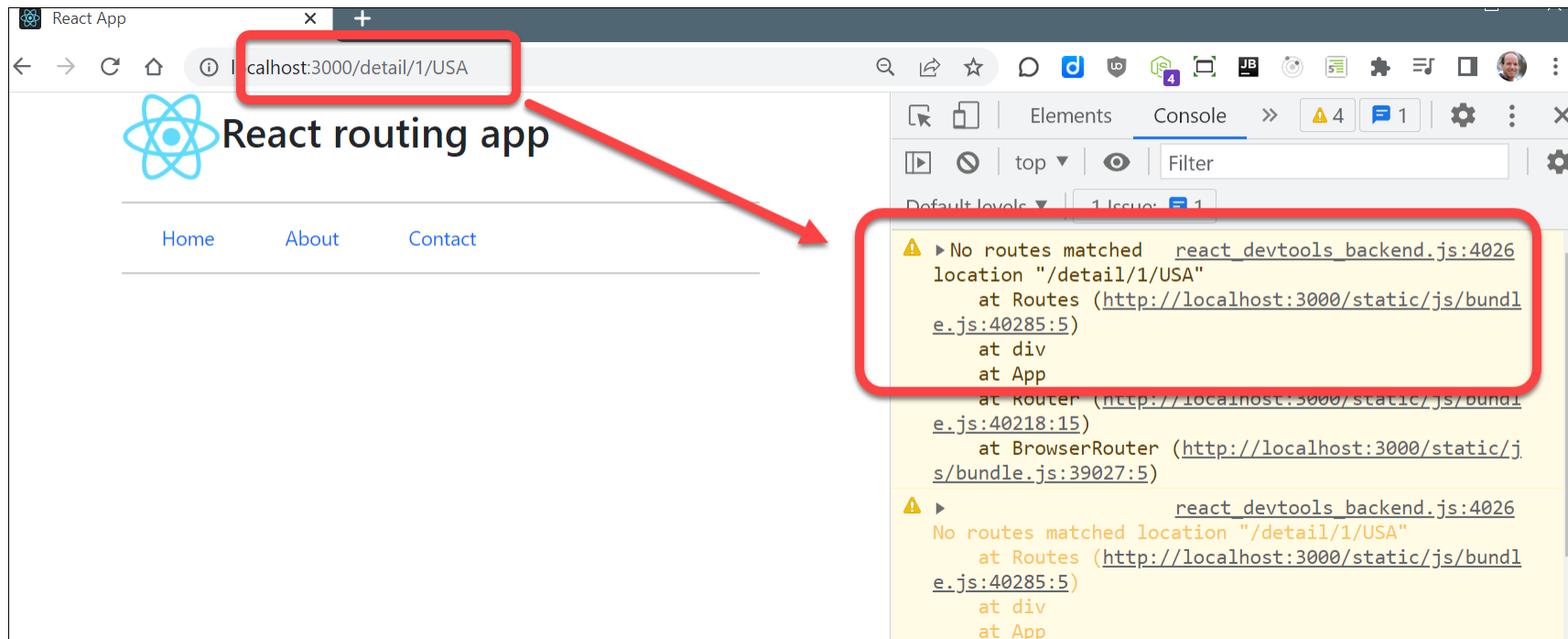


```
<ul className="list-group">
  {this.state.countries.map(country =>
    <li className="list-group-item"
      key={country.id}>
      <Link to={`/detail/${country.id}/${country.name}`}>
        {country.name}
      </Link>
    </li>
  )}
</ul>
```

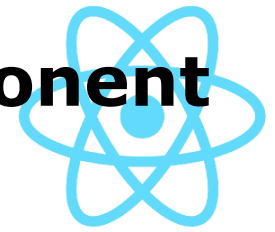
That didn't work out well...



- Link is constructed, Params are send, but no matching route!



2. Optional – create FileNotFound component



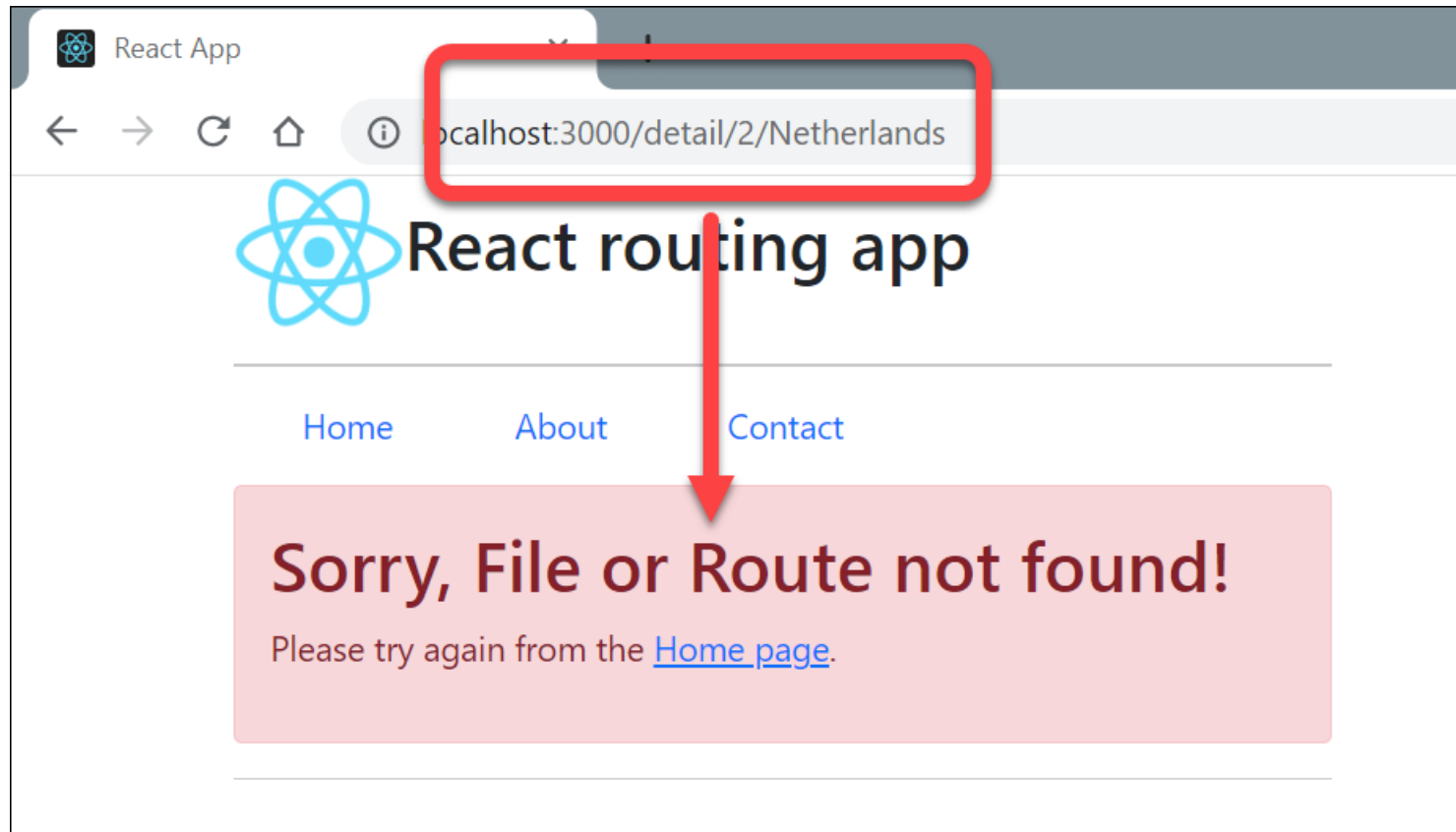
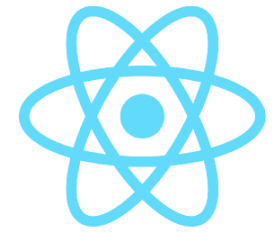
- Add/Create a `<FileNotFound />` component
- Add a `NoMatch`-route:

```
// FileNotFound.js
import {Link} from "react-router-dom";

const FileNotFound = ()=>{
  return(
    <div className="alert alert-danger">
      <h1>Sorry, File or Route not found!</h1>
      <p>
        Please try again from the <Link to="/">Home page</Link>
      </p>
    </div>
  )
}
export default FileNotFound
```

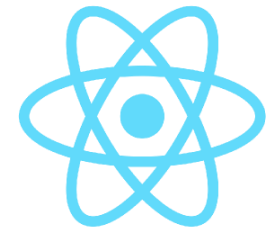
```
<Routes>
  ...
  <Route path='*' element={<FileNotFound />}/>
</Routes>
```


Result



(P.S. – always add the `FileNotFound` route last!)

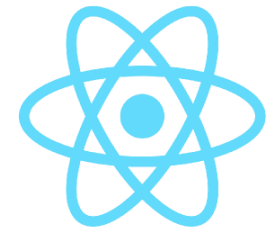
3. Create a route to detail page



- Add a `<Route />` to show a new Detail Component
- Note the colon `/:id` and `/:name`. These are the **dynamic parameters**

```
...  
<Route path="/detail/:id/:name" component={CountryDetail}/>
```

4. Create detail component



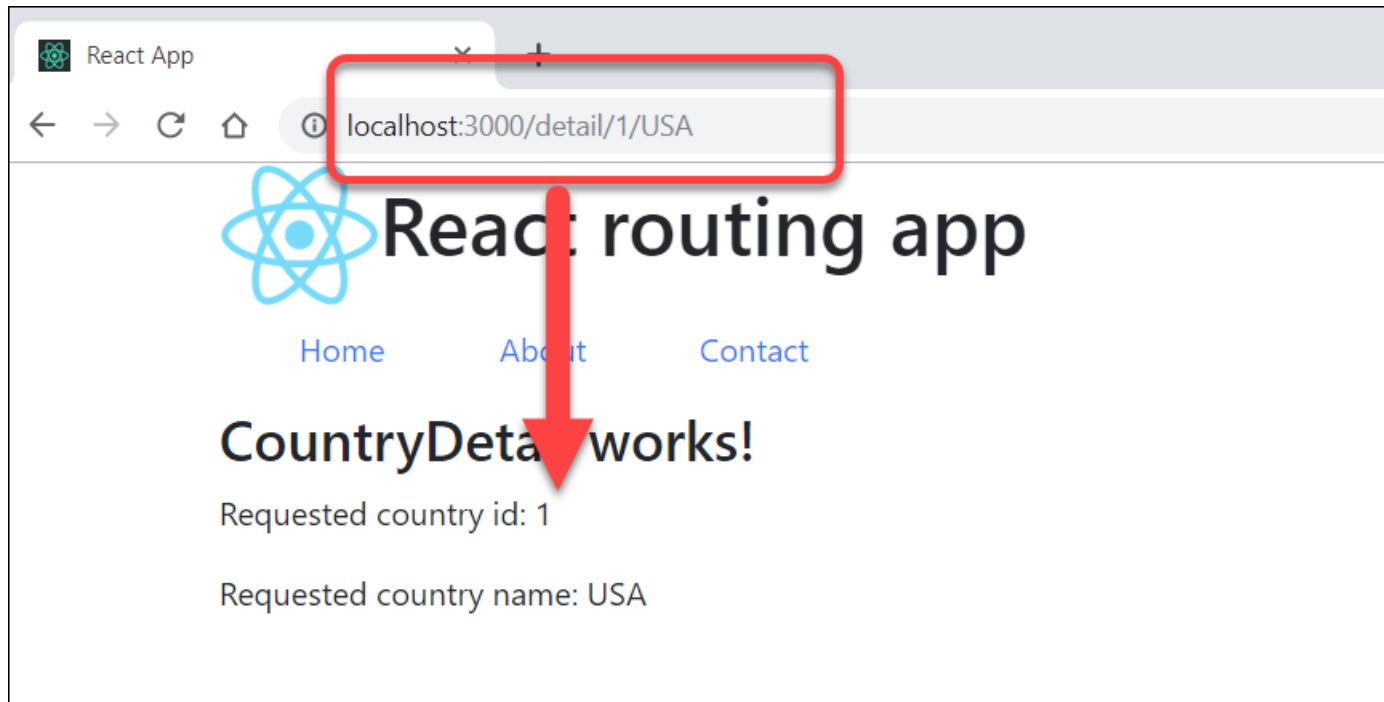
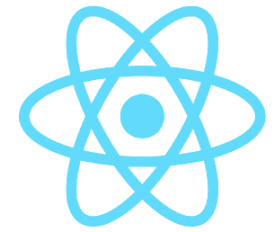
```
import countryData from "../../data/CountryData";
import {useParams} from "react-router-dom";

const CountryDetail = () => {
  let params = useParams();
  const countries = countryData.countries;
  const id = params.id;
  const getCountry = id => countries.find(c => c.id === +id)
  const country = getCountry(id);

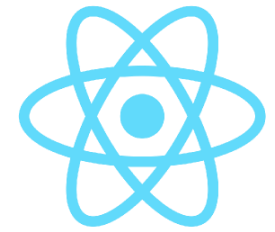
  return (
    <div>
      <h3>Details for country</h3>
      <ul className="list-group">
        <li className="list-group-item">
          id: {country.id}
        </li>
        (...)
      </ul>
    </div>
  );
}
export default CountryDetail;
```

Retrieve
dynamic
parameters
from
useParams()
hook

4. Show Results in the UI



Show Country data

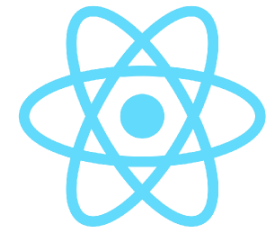


- Grab the data (for now: **local**, to get the correct country)
 - You know how to fetch data from an API from the previous module
- Look for the data based on the `id`

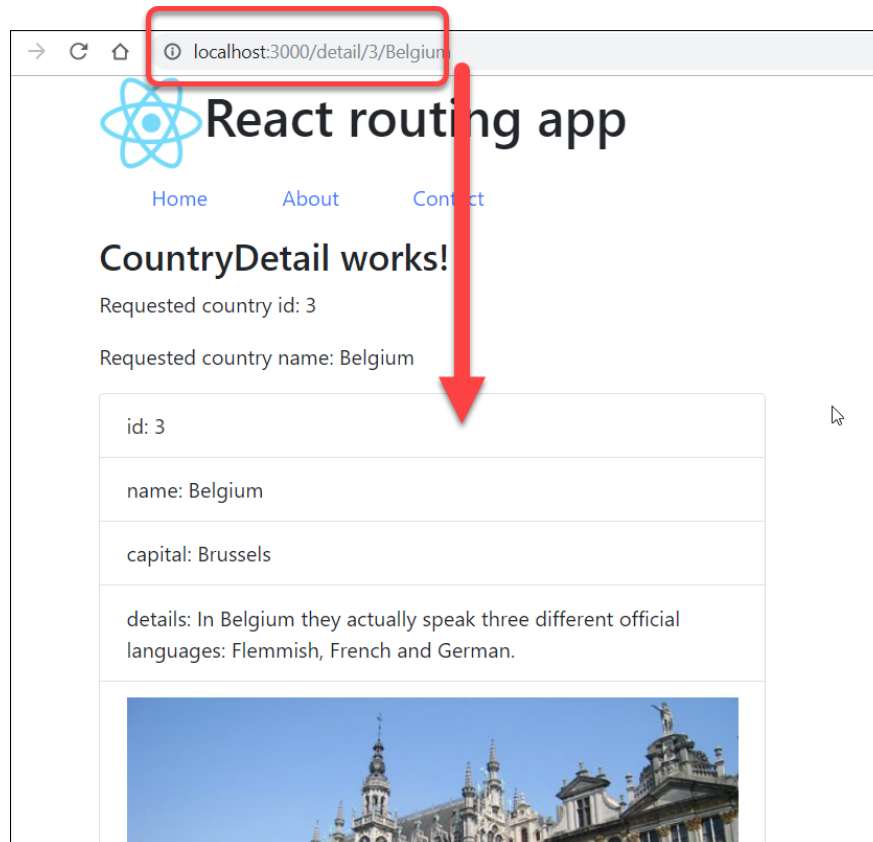
```
const getCountry = id => countries.find(c => c.id === +id)
const country = getCountry(id);
```



Show Country Details in UI

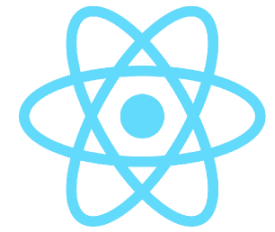


- Create DetailComponent as usual

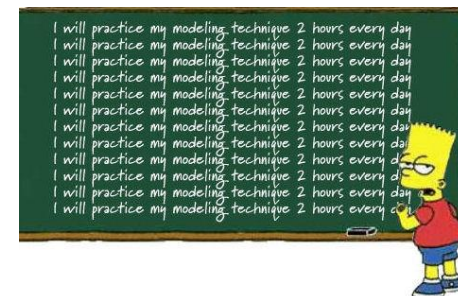


[../examples/620-routing-parameters](#)

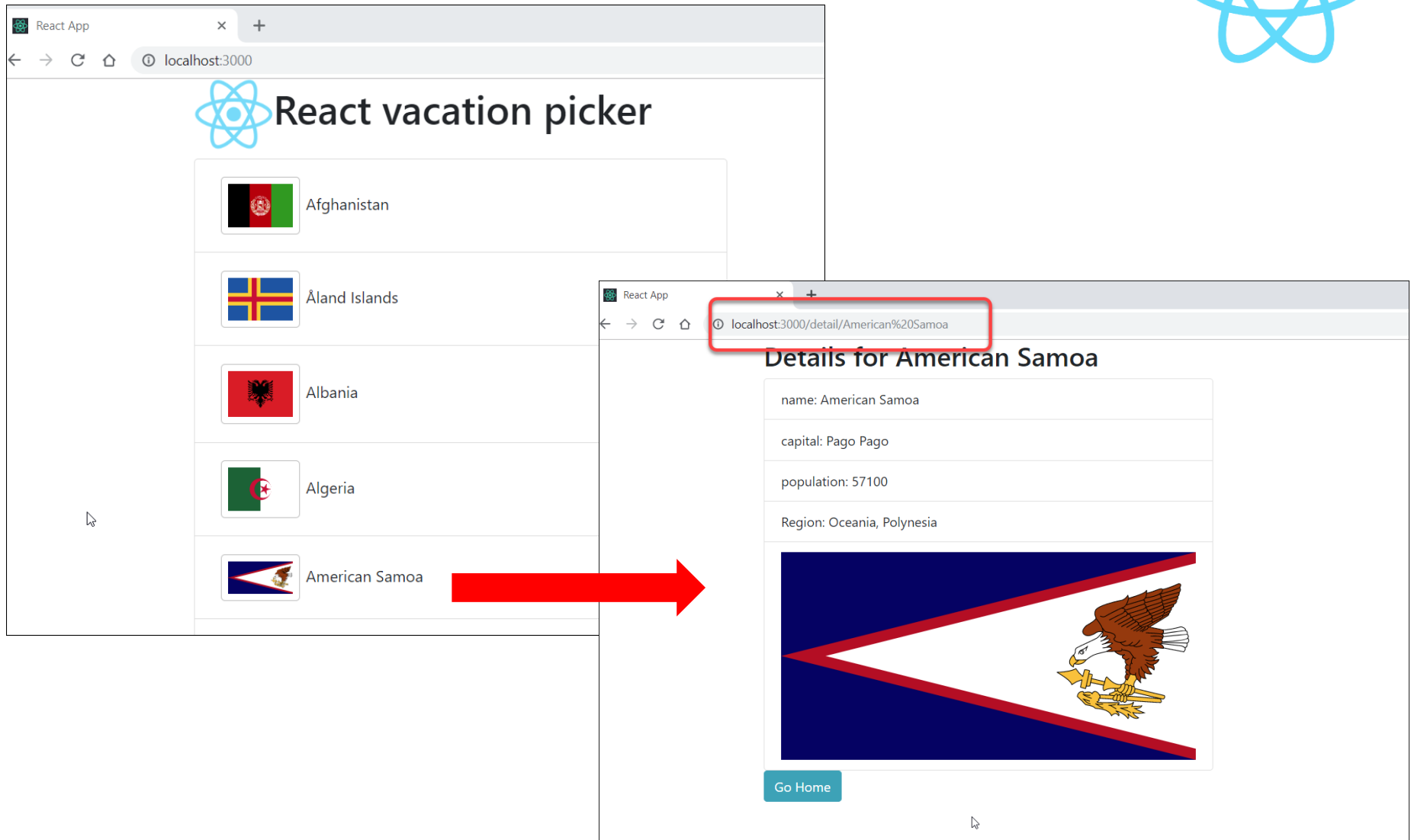
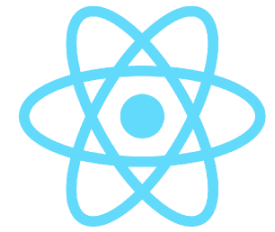
Workshop



- Study the example project
- See how routing parameters work.
- Try some different parameters, or adding/removing a parameter – **CREATE THIS IN YOUR OWN PROJECT**
- Example `../620-routing-parameters`
- **OR:** Work from example `../500-api-call`
 - Add routing to the project
 - Clicking on a specific country from the API navigates to a detail component, which shows and fetches country details
 - (solution: `../workshops/50-routing-api`)



Sample result from the workshop



The image displays two screenshots of a web application titled "React vacation picker".

The left screenshot shows the main interface with a list of countries, each represented by a flag and its name:

- Afghanistan
- Åland Islands
- Albania
- Algeria
- American Samoa

A red arrow points from the "American Samoa" entry to the right screenshot.

The right screenshot shows the "Details for American Samoa" page. The URL in the browser is `localhost:3000/detail/American%20Samoa`. The details displayed are:

- name: American Samoa
- capital: Pago Pago
- population: 57100
- Region: Oceania, Polynesia

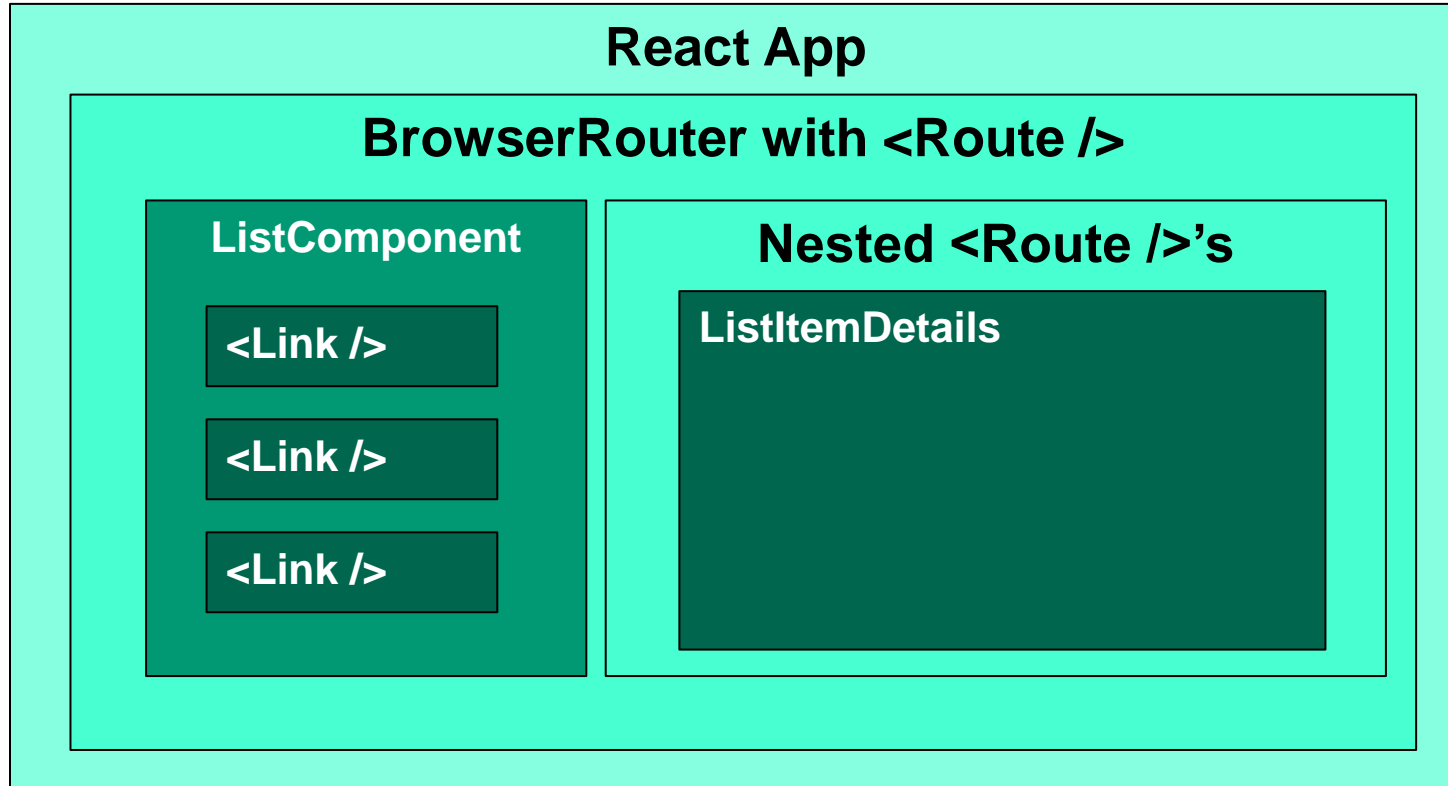
Below the text details is a large image of the flag of American Samoa, which features a blue field with a white triangle and a red border, and a brown eagle with a shield on its chest.

A "Go Home" button is visible at the bottom left of the details page.



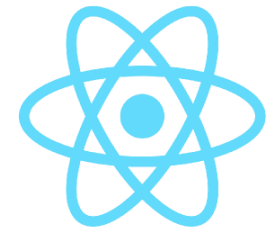
Nested Routes

What are nested router views?



Show router links inside another routed component, but still update the URL

Adding a nested router

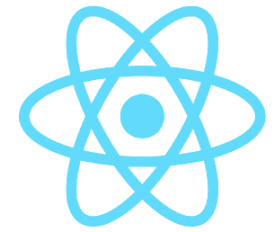


- The `<Route />` is just a component, so you can place it inside other components
 - And, if necessary – conditionally.
 - In example app: `<Home />`
- React does *not have* an array of 'child' routes in the main routing table, like other frameworks

- ```
children: [
 { <child-router-path-1> }, { <child-router-path-2> }
]
```

**Invalid in  
React!**

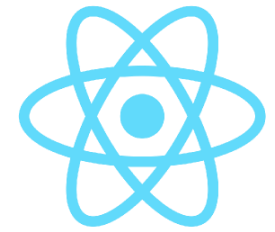
# 1. Create nested routes



```
<Routes>
 <Route path="/" element={<Home/>}>
 <Route path=':id/:name' element={<CountryDetail/>}/>
 </Route>
 ...
</Routes>
```

**<Route /> in Home component.  
If the URL matches, the  
component renders. Otherwise  
the route returns `null`**

## 2. Add `<Outlet />` to the component



- Import, and use `<Outlet />` in the component that contains the nested route

- In our case: `<Home />`

Add the `<Outlet />` tag. It shows the nested routes

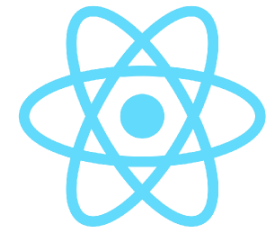
```
import countryData from "../data/CountryData";
import {Link, Outlet} from "react-router-dom";

const Home = () => {
 const countries = countryData.countries;

 return (
 <div className="row">
 <div className="col">
 <h1>Home Component</h1>
 <h2>List of countries</h2>
 ...
 </div>
 <div className="col">
 <Outlet />
 </div>
 </div>
)
}

export default Home
```

### 3. Clearing the route



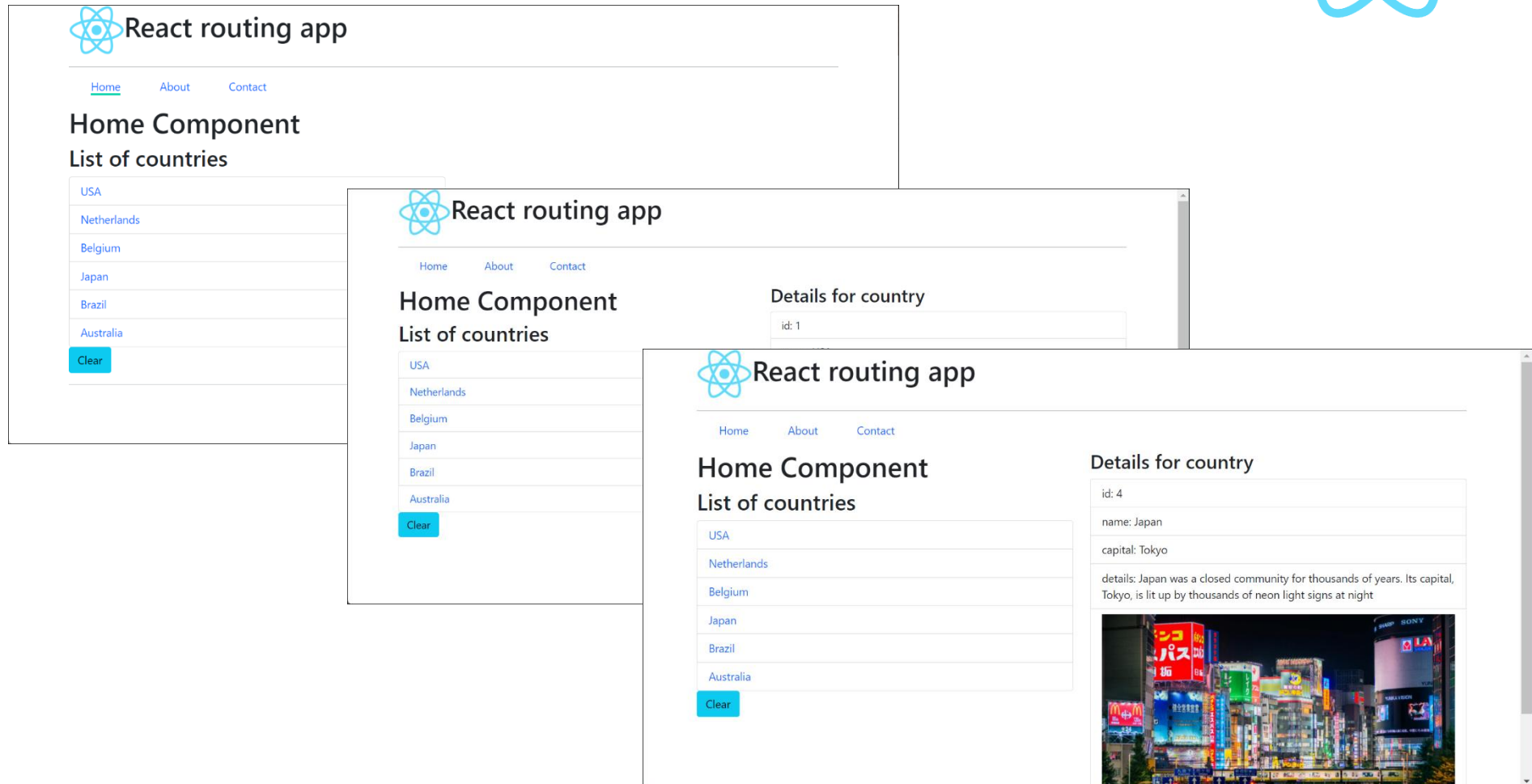
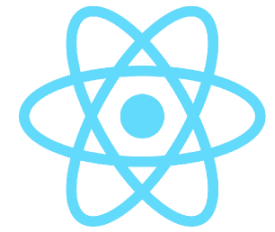
- Various ways. We simply navigate back to `<Home />`
- Import and assing `useNavigate()` hook, as before

```
import {Link, Outlet, useNavigate} from "react-router-dom";

const Home = () => {
 const countries = countryData.countries;
 const navigate = useNavigate();
 ...
}
```

```
<button onClick={()=>navigate('/') }
 className="btn btn-info">Clear</button>
```

# Result

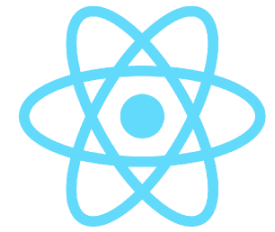


The image displays three overlapping screenshots of a web application titled "React routing app".

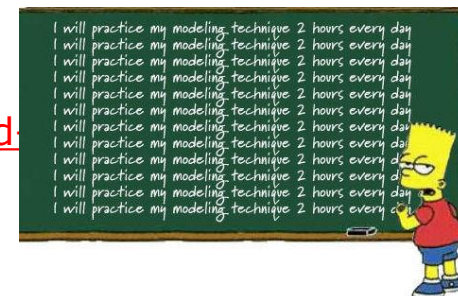
- Top-left screenshot:** Shows the "Home Component" with a navigation bar (Home, About, Contact) and a "List of countries" containing: USA, Netherlands, Belgium, Japan, Brazil, Australia. A "Clear" button is at the bottom.
- Middle screenshot:** Shows the "Home Component" with the same "List of countries". To its right, a "Details for country" section shows "id: 1".
- Bottom-right screenshot:** Shows the "Home Component" with the same "List of countries". To its right, the "Details for country" section shows "id: 4", "name: Japan", "capital: Tokyo", and a paragraph: "details: Japan was a closed community for thousands of years. Its capital, Tokyo, is lit up by thousands of neon light signs at night". Below the text is a photograph of a brightly lit city street at night with many neon signs.

App works as expected

# Workshop



- Study the example project.
- Do this for your own app with a list of data
  - Clicking on a listitem, shows the details in a nested route
- Example `../630-nested-routes`
- OR: Work from your previous workshop
  - Clicking on a specific country from the API navigates to a detail component, which shows and fetches country details *inside a nested route*
- Study the React Router Example:
  - <https://reactrouter.com/docs/en/v6/getting-started/tutorial#nested>
  - <https://stackblitz.com/edit/github-agqlf5?file=src%2FApp.jsx>



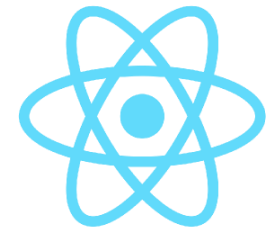




# Miscellaneous routing

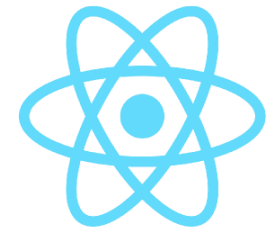
Using `<Switch />`, passing props and more

# Various routing tips & tricks



- Optional parameters
- Using `render`

# Search Params



- You want some optional data to be added to the URL
- Use `searchParams` for that, in combination with `navigate()`
- For instance – we added a `population` field to our data
  - But not all records have a `population` field
  - We only want to pass it, if it is known!

```
countries: [
 {
 id: 1,
 name: 'USA',
 ...,
 population: 331000000
 },
 {
 id: 2,
 name: 'Netherlands',
 ...no population available
 },
]
```

# Sending searchParams:

```
 countryDetails(country)}>
 {country.name}

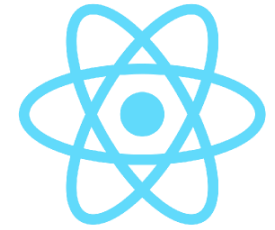
```



```
const navigate = useNavigate();

const countryDetails = (country) => {
 navigate({
 pathname: `${country.id}/${country.name}`,
 search: country.population ? `?population=${country.population}` : null
 })
}
```

# Receiving searchParams:

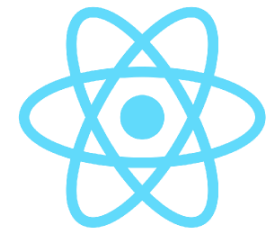


```
const [searchParams] = useSearchParams()
const population = searchParams.get('population');
```



```
{
 population &&
 <li className="list-group-item">
 Population: {population}

}
```



## Home Component

### List of countries

[USA](#)  
[Netherlands](#)  
[Belgium](#)  
[Japan](#)  
[Brazil](#)  
[Australia](#)

Clear

## Details for country


id: 2

name: Netherlands

capital: Amsterdam

Population: 17000000

details: The capital of the Netherlands, Amsterdam, is over 1000 years old.



[Home](#) [About](#) [Contact](#)

## Home Component

### List of countries

[USA](#)  
[Netherlands](#)  
[Belgium](#)  
[Japan](#)  
[Brazil](#)  
[Australia](#)

Clear


## Details for country

id: 3

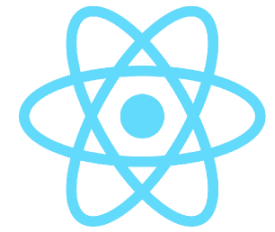
name: Belgium


capital: Brussels

details: In Belgium they actually speak three different official languages: Flemish, French and German.





# More info on search params




 **Get 67% off the React Master Bundle!**  
See the bundle then add to cart and your discount is applied.


0 : 23 : 16 : 23  
DAYS HOURS MINS SECS


 Ultimate Courses™

Courses  About Reviews eBooks Blog Contact Login




## Getting Query Strings (Search Params) in React Router







 by Todd Motto · React · Nov 12, 2021 · 5 mins read



### Learn React the right way!

Everything you need to become a React expert


 by Todd Motto  
Google Developer Expert

-  React Basics 84 lessons
-  React Styling 9 lessons
-  React Router v6 37 lessons
-  React State Management 22 lessons
-  React Masterclass 0 lessons
-  React Universal 0 lessons

See Courses →

Reading Query Strings, otherwise known as Search Params, from the URL is a common practice in web application development and you're going to learn how to read them with React Router. React Router v6 provides a `useSearchParams()` hook that we can use to read those query string search params that we need from the URL.

✦ Written for React Router v6, check out my brand new React Router v6 course to fully master it.

 A newer version of this site just became available. Please refresh this page to activate it.

<https://ultimatecourses.com/blog/query-strings-search-params-react-router>

<https://ultimatecourses.com/blog/navigate-to-url-query-strings-search-params-react-router>

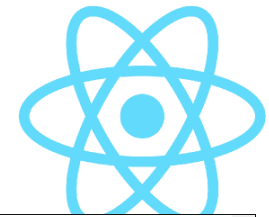



# More info

More info on React Router and techniques



# Official documentation



 **React Router**

DocumentationResourcesGitHubnpm

**DOCS HOME**

**GETTING STARTED**

Installation

Quick Start

Tutorial

FAQs

Main Concepts

**ROUTERS**

BrowserRouter

HashRouter

HistoryRouter

MemoryRouter

NativeRouter

Router

StaticRouter

**COMPONENTS**

Link

Link (React Native)

NavLink

## Welcome to React Router

### New to React Router?

We suggest you start with the tutorial. It's got everything you need to know to get up and running in React Router quickly.

### Familiar with React Router?

We introduced several new features and exciting changes in version 6. Learn all about them in this quick overview of the features that make v6 special.

### Wanna dive deep?

Level up your understanding of React Router with the concepts, vocabulary, and design principles of React Router and routing in general. A must read for would-be contributors.

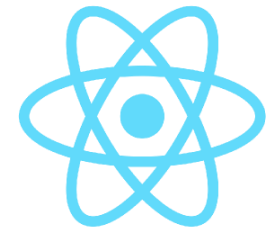
### Frequently Asked Questions

Running into a problem? Chances are, you're not the first! Explore some of the questions that people commonly have about React Router v6.

### Previous Versions

- [React Router v4/5 docs](#)
- [React Router v3 docs](#)

<https://reactrouter.com/docs/en/v6>



freeCodeCamp (🔥)

Forum Donate


Learn to code – [free 3,000-hour curriculum](#)

DECEMBER 14, 2021 / #REACT

## React Router Version 6 Tutorial – How to Set Up Router and Route to Other Components



Ihechikara Vincent Abba

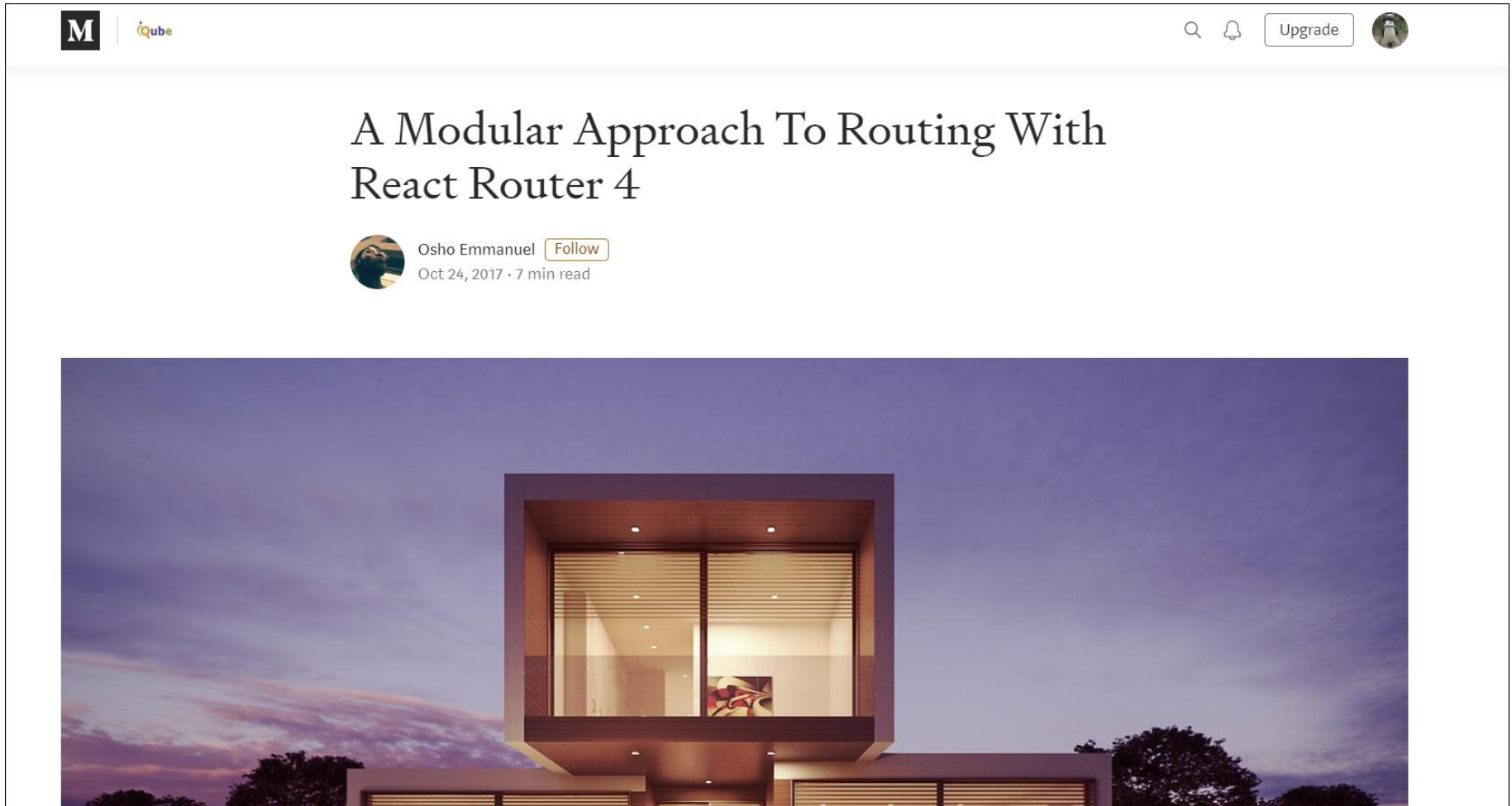
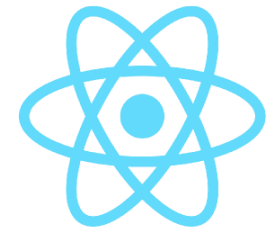


## React Router Version 6



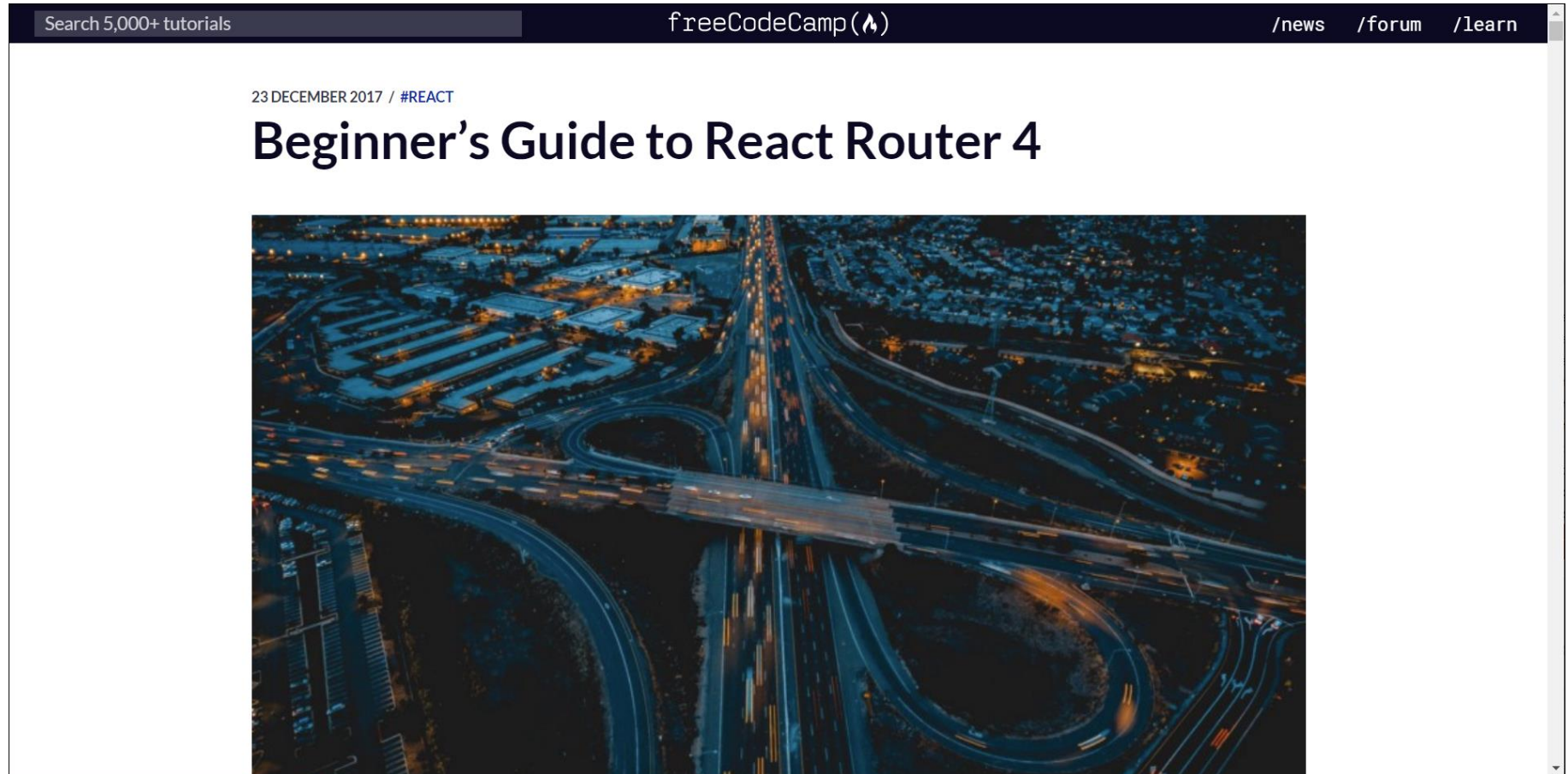
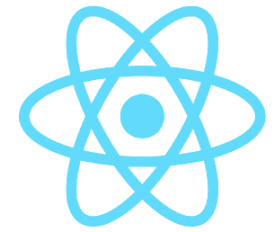
<https://www.freecodecamp.org/news/how-to-use-react-router-version-6/>

# Blogs and more (but mind the version number)...



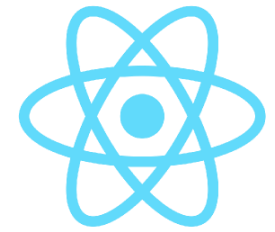
<https://medium.com/iqube-bits/a-modular-approach-to-routing-with-react-router-4-d4a3db9f56ae>

# Beginner's Guide to React Router



<https://www.freecodecamp.org/news/beginners-guide-to-react-router-4-8959ceb3ad58/>

# Course React Router



TYLERMCGINNIS.COM

For Business **Courses** Blog Newsletter Reviews Login

## React Router

4.8 ★★★★★ 419 Reviews

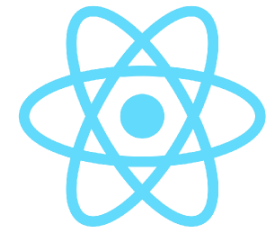
Updated 15 months ago

This course is **guaranteed** to be up to date.

For good reason, React Router is the most popular 3rd party library in the React ecosystem. If you're using React, odds are you're also using React Router. React Router v4 introduced a new dynamic, component based approach to routing. If you're used to static routing (like in Express, Angular, or Ember), this new paradigm might be difficult to grasp. The goal of this course is to tackle every scenario you might encounter when building an app with React Router so that when the time comes, you're ready. With over six hours of video, 18 different routing scenarios, and an entire real world project you'll build, we can confidently say this course is the most comprehensive and effective way to

<https://tylermcginnis.com/courses/react-router/>

# Checkpoint



- You know how to **install** the router and how it works
- You are familiar with `<BrowserRouter>`, `<Route>`, `<Link>` and `<NavLink>`
- You can **highlight** the current route
- You can **navigate from code**
- You know how to handle route **parameters**
- You can create **nested routes** and components