Building Web Applications with React

CHAPTER 5: ROUTING

Chapter Objectives

In this chapter, we will:

- Consider the role of routing in an SPA
- Explore the `react-router` module
- Add routes to the application
- Create a parametrized route

Chapter Concepts

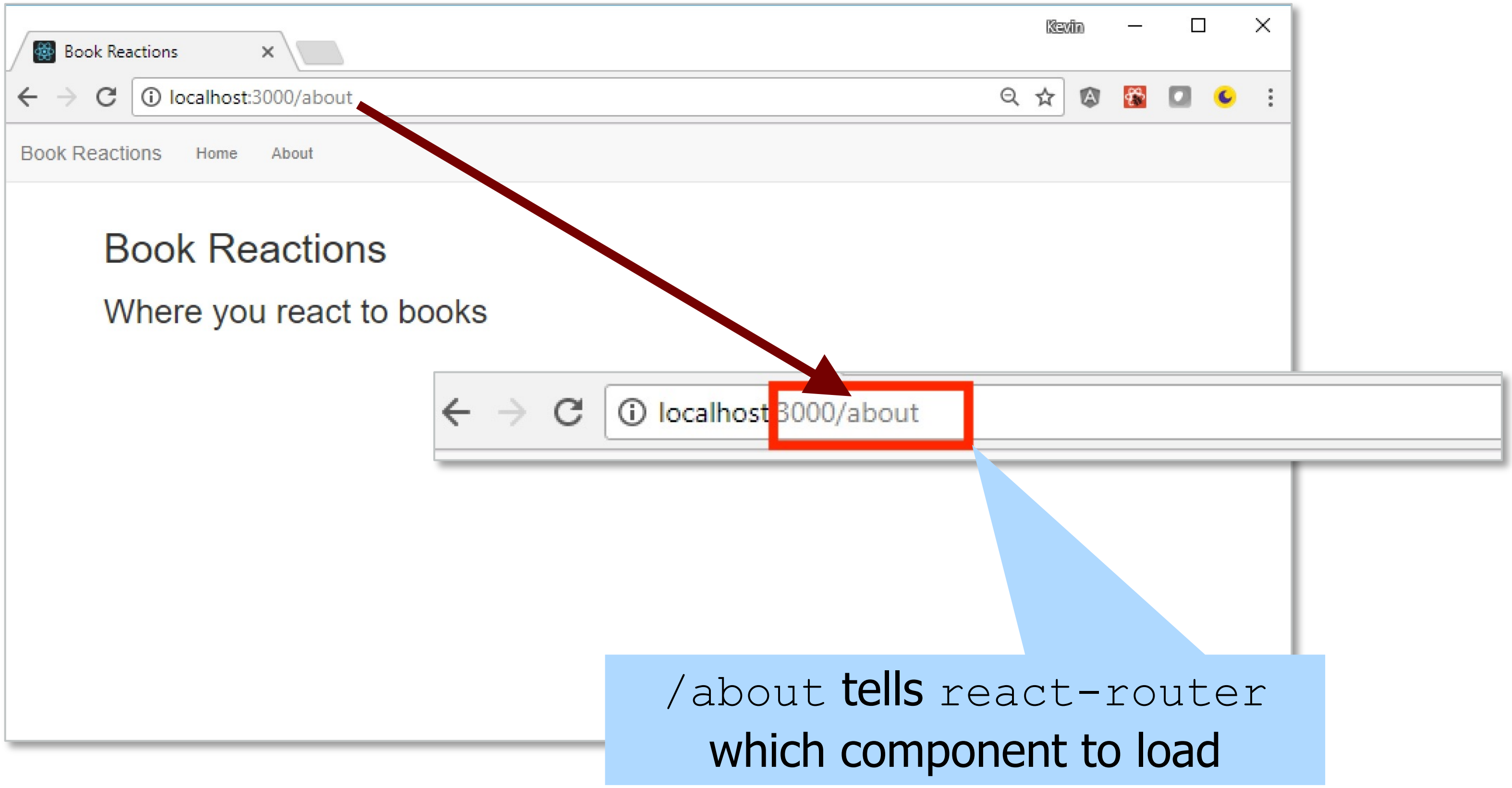
SPA Routes

- Single Page Applications are just that:
  - They only contain one web page
- All additional ‘pages’ are created dynamically using Ajax data
- Possible to have an SPA without routing
  - Limitations
    - Users cannot bookmark any content except default home page
    - Very complex to manage transitions between content
      - Only realistic option for very small applications
- SPA applications of any size require routing

SPA Route Advantages

- Routing in an SPA allows:
  - Creation of links that simulate behavior of traditional web apps
    - Bookmarked link goes to desired location within application
      - Known as ‘deep linking’
  - Simple transitions between different parts of the application
  - Automation of login redirects
  - More
- SPA routes are logical, not physical
  - Use RESTful hierarchies to specify resources

Routing in SPAs Illustrated



Book Reactions

Where you react to books

`/about` tells `react-router` which component to load

Chapter Concepts

Routing in React

- Facebook does not provide a routing implementation
- Third-party module `react-router` widely used
- Provides React components to define routes
  - `<BrowserRouter></BrowserRouter>`
  - `<Route />`
- Also provides the `<Link />` component
  - Used to generate HTML links to routes

`react-router`

- React Router is defined in three packages:
  - `react-router`: the core package
  - `react-router-dom`: browser-specific components
  - `react-router-native`: components for native apps
    - Android, IoS
- No need to import `react-router` to components
  - `react-router-dom` re-exports all `react-router` exports

Routing with `react-router-dom`

- Steps to routing with `react-router-dom`

1. Import `BrowseRouter` , and `Route`

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

   – `{ }` syntax reflects multiple module exports in `react-router-dom`

2. Set `BrowserRouter` as the root component in the App.js

```
<BrowserRouter>
  <div className="container-fluid">
    <Navigation />
    <div className="container">
      <Route exact path="/" component={BookList} />
      <Route path="/about" component={About} />
   </div>
  </div>
</BrowserRouter>
```

3. Add Routes mapping paths to components

The Route Component

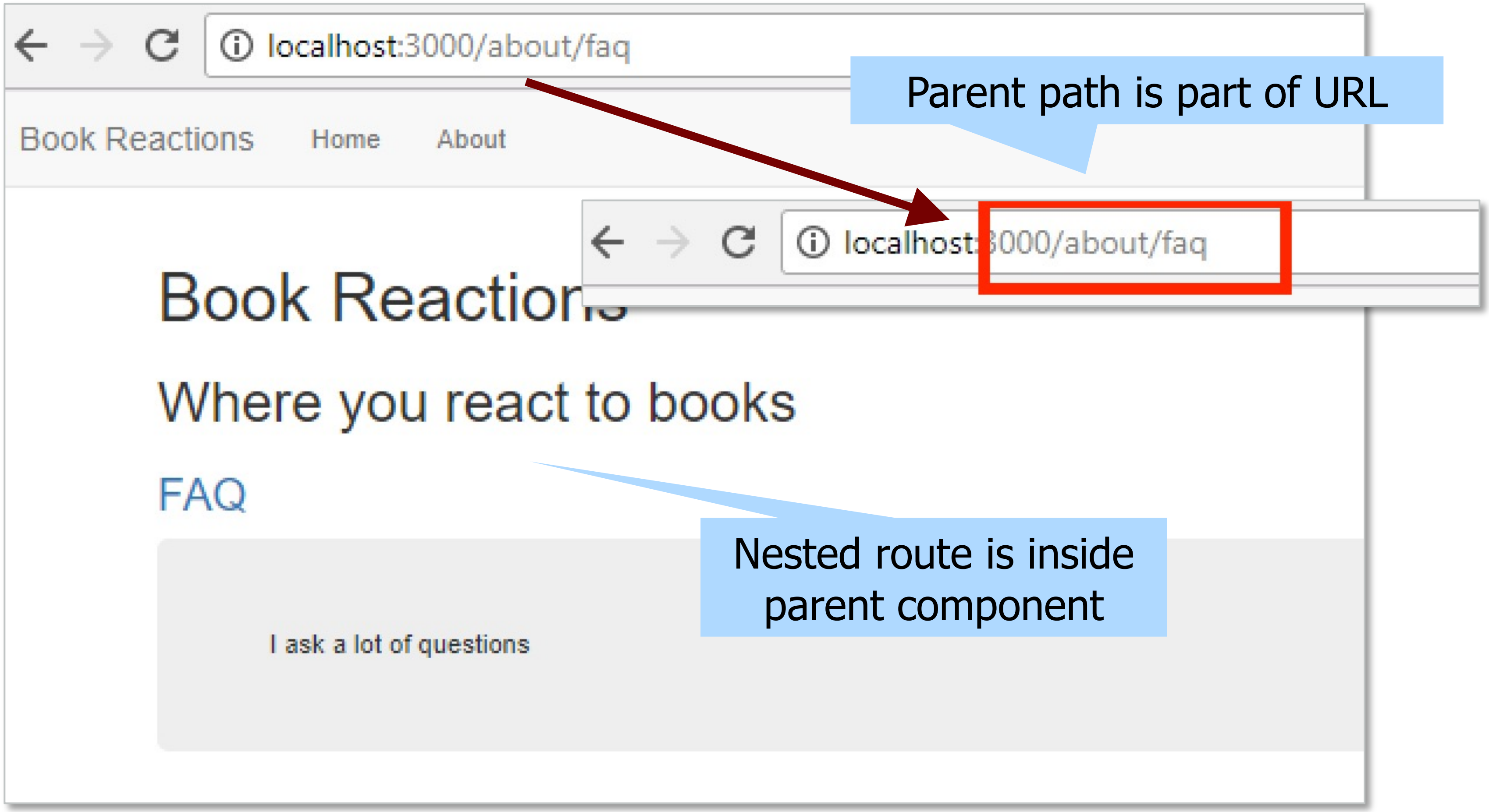
- Defines an individual route
  - Matching a URL pattern to a React component
- Can be placed anywhere inside the component tree
  - Not just in the root component
  - Much more flexible than configuration-based routing

The `match` prop makes it easy to build a URL that includes the parent path

```
const About = ({match}) => (
  <div className="row">
    <h1>Book Reactions</h1>
    <h2>Where you react to books</h2>

    <h3><Link to={match.url + "/faq"}>FAQ</Link></h3>



    <Route path={match.url + "/faq"} component={Faq} />
</div>)
```

This is a *nested* route

Nested Routes


localhost:3000/about/faq

Book Reactions

Where you react to books

FAQ

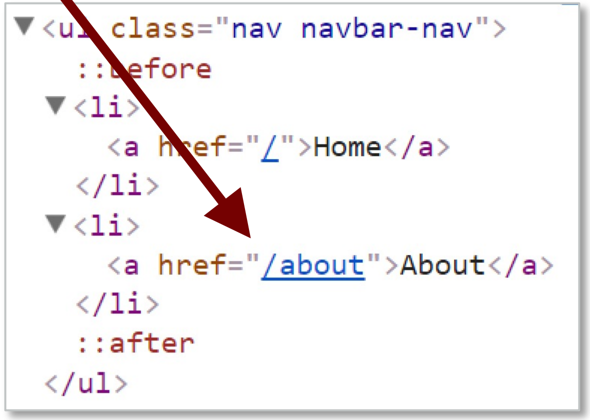
I ask a lot of questions

Book Reactions
Home
About
localhost:3000/about/faq
Parent path is part of URL
Nested route is inside parent component

The `Link` Component

- Used to create HTML `<a>` elements
  - Pointing to routes defined in root component
  - The `to` prop maps to `Route` *`path`*

```
const Navigation = () => (
          <ul className="nav navbar-nav">
             <li><Link to="/">Home</Link></li>
             <li><Link to="/about">About</Link></li>
         </ul>
);
```



```
▼<ul class="nav navbar-nav">
   ::before
  ▼<li>
     <a href="/">Home</a>
   </li>
  ▼<li>
     <a href="/about">About</a>
   </li>
   ::after
 </ul>
```

Exercise 5.1: Adding Routes to the Application

HANDS-ON EXERCISE

- In this exercise, you will add routing to your Single Page Application
- Please refer to the Exercise Manual

Debrief: Wrapper Components

- React components can wrap other components

```
<BrowserRouter>
  <div className="container-fluid">
  <Navigation />
  <Route exact path="/" component={BookList} />
  <Route path="/about" component={About} />
  </div>
</BrowserRouter>
```

- Wrapper components can apply functionality to arbitrary children
  – Many third-party libraries are written this way
- Problem: wrapper component can't know what the children will be
  – How to add them inside the parent JSX?
- Solution: `props.children`
  – Allows parent component access to its children
  – We will do this later in the class

Programmatic Routing

- Not all navigation is user-directed
  - Sometimes, the application may redirect the user programmatically
    - For example, before and after logging in
- Use the `Redirect` component to pass user to new location

```
const PrivateRoute = ({ component: Component, ...rest }) => (
   <Route {...rest} render={props => (
     user.isAuthorized() ? (
         <Component {...props}/>
       ) : (
          <Redirect to={{
               pathname: '/forbidden',
               state: { from: props.location } }}/>
               )
     )}/>
);
```

If user is authenticated, pass through to component; otherwise, inject `Redirect`

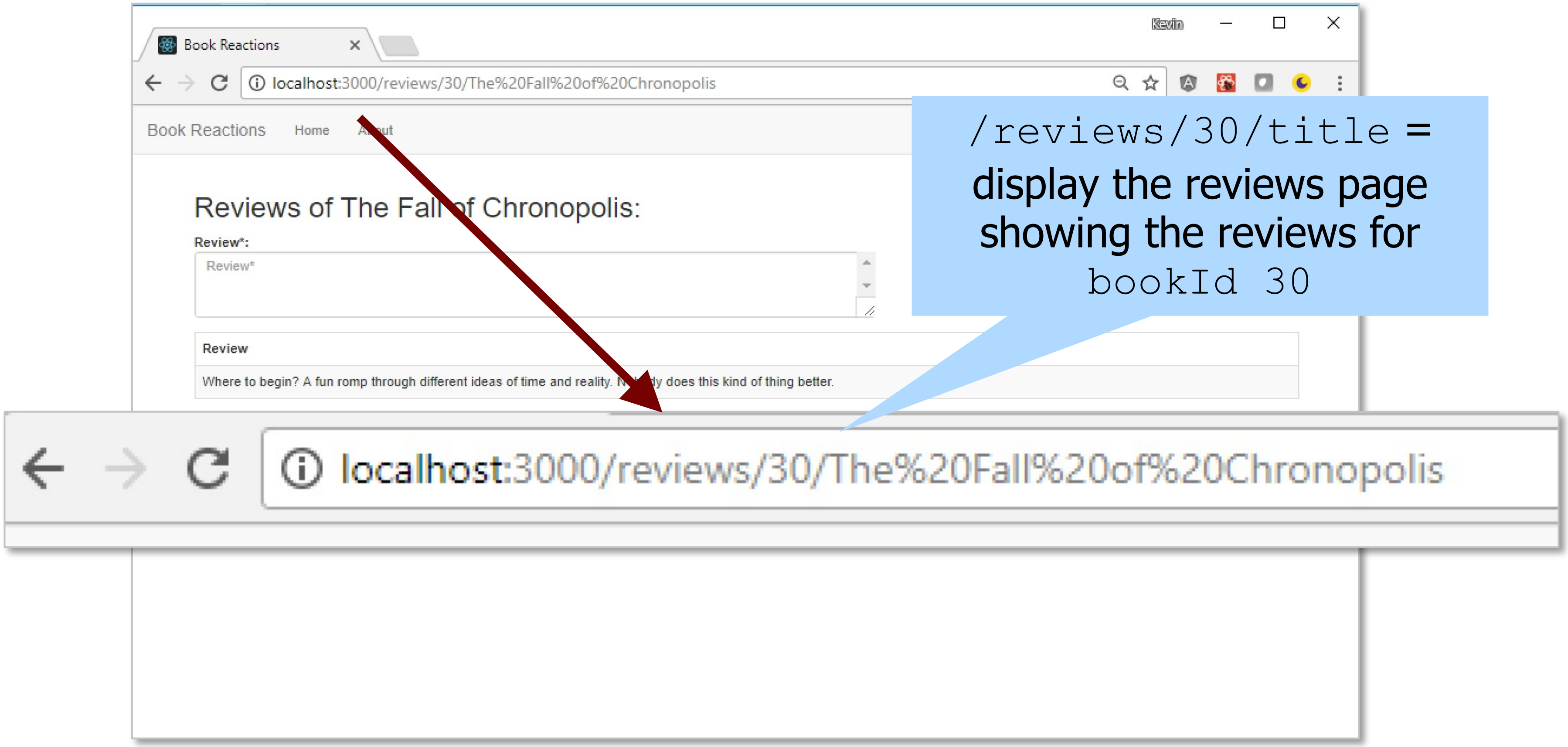
Use `PrivateRoute` inside `BrowserRouter`

```
<PrivateRoute path="/faq" component={Faq} />
```

Chapter Concepts

Route Parameters

- Some routes need parameters in order to be useful
  - Can't ask for book reviews without knowing which book



Book Reactions

localhost:3000/reviews/30/The%20Fall%20of%20Chronopolis

Reviews of The Fall of Chronopolis:

Review*:

Review
Where to begin? A fun romp through different ideas of time and reality. Nobody does this kind of thing better.

`/reviews/30/title` = display the reviews page showing the reviews for `bookId 30`

Defining and Retrieving Parameters

- Parameters are defined on the individual `<Route />`
  - Each parameter has its own path segment, prefixed with a colon
    - The colon is not included in the actual URL

```
<Route path="reviews/:bookId/:bookName" />
```

Each `/:name` is a parameter

- Can be accessed inside component via `props.match.params`

```
    componentWillMount() {
        this.props.receiveReviews(this.props.match.params.bookId);
    }
```

Destructuring Parameters

- Functional components optionally receive props as an argument
  - Standard to destructure to retrieve named properties
  - Makes code inside the function simpler and more readable

```
function ReviewList({ match: { params: { title, bookId } } }) {
```

`title` and `bookId` will be available inside the function

- Class components also frequently use destructuring

```
const { title, author, cover } = this.props;
return (<tr>
        <td>{author}</td>
        <td><img src={cover} alt={title} /></td>
       </tr>);
```

Passing Parameters

- Parameters are passed to the routing module as part of the URL
  - Can be done programmatically with `push()`
  - Or via the `<Link />` component using ES6 template strings
- ES6 template strings are delimited by `` ` ' `` back-ticks
  - Allow interpolation of programmatic content inside string

Back-ticks for template strings

Colon is NOT part of URL

```
<Link to={`/reviews/${this.props.bookId}/${this.props.title}'}>
         {this.props.title}
</Link>
```

`${variable}` inserted into string

Exercise 5.2: Passing and Receiving Route Parameters

- In this exercise, you will set and retrieve route parameters
- Please refer to the Exercise Manual

Chapter Concepts

Routing in SPAs

Introducing `react-router`

Parameterized Routes

Chapter Summary

Chapter Summary

In this chapter, we have:

- Considered the role of routing in an SPA
- Explored the `react-router` module
- Added routes to the application
- Created a parametrized route