Outline

Some final words and what you can learn after taking this course!

Outline

So now we've reached the end of The Road to Learn React. I hope you enjoyed it, and I hope it helped you to gain some traction in React. If you liked the course, tell your friends about it.

So, where can you go from here? I recommend you extend the application on your own, so eventually, you can start creating your very own React projects. Consider doing this before you dive into another course or tutorial. Do it for one week, take it to production by deploying it, and reach out to me me or others to showcase it. I am always interested in seeing what my readers built and helping them along.

If you are looking for more extensions for your application, I recommend several learning paths after you've mastered the basics:

- **Connecting to a Database and/or Authentication:** In a growing React application, you may want to persist data eventually. The data should be stored in a database so it can survive after a browser session, and so it can be shared across different users using your application. The simplest way to introduce a database is Firebase. In this tutorial, you will find a step-by-step guide on how to use Firebase authentication in React. Beyond that, you can use Firebase's real-time database to store user entities.

- **State Management:** You have used React `this.setState()` and `this.state` to manage and access local component state. That's a good start. In a larger application, however, you will experience the limits of React's local component state. It is imperative you learn to use third-party state management libraries like Redux or MobX. On the Road to React platform, you will find the course "Taming the State in React" that

platform, you will find the course "Taming the State in React" that teaches about advanced local state management using Redux and MobX.

The course comes with an ebook as well, but I recommend you dive into the source code and screencasts too.

- **Tooling with Webpack and Babel:** We used *create-react-app* to set up the application we created for this course. At some point, you may want to learn the tooling around it, which enables you to setup your own project without *create-react-app*. I recommend a minimal setup with Webpack and Babel, after which you can apply additional tooling on your own. For instance, you could use ESLint to follow a unified code style.

- **Code Organization:** Recall if you will the chapter about code organization. You can apply these changes now if you haven't already. It will help organize your components in structured files and folders (modules), and it will help you understand the principles of code splitting, reusability, maintainability, and module API design. Your applications will eventually grow and need to be structured into modules, so it's better to start now.

- **Testing:** The course only scratched the surface of testing. If you are not familiar with the topic, you should dive deeper into unit testing and integration testing, especially with React applications. I would recommend sticking to Enzyme and Jest for implementations, to refine your approaches with unit and snapshot tests.

- **React Component Syntax:** The best practices for implementing React components evolve over time. You will find many ways to write your React components, especially React class components, in other learning resources. A GitHub repository called react-alternative-class-component-syntax is a great way to learn alternative ways to write React class components. By using class field declarations, you can write them even more concisely in the future.

- **UI Components:** Many beginners make the mistake of introducing UI component libraries too early in their projects. It is more practical to learn how to implement and use a dropdown, checkbox, or dialog in React with standard HTML elements. Most of these components will manage their own local state. A checkbox has to know whether it is

checked or unchecked, so you should implement them as controlled components. After we covered the foundational implementations,

introducing a UI component library with checkboxes and dialogs as React components should be easier.

- **Routing:** You can implement routing for your application with react-router. So far, there is only one page in your application. React Router helps manage multiple pages across multiple URLs. When you introduce routing to your application, you don't make any requests to your web server to request the next page. The router will do everything for you on the client-side. So get your hands dirty and introduce routing in your application.

- **Type Checking:** Earlier, we used React PropTypes to define component interfaces. It is a good practice to prevent bugs, but the PropTypes are only checked on runtime. You can go further by introducing static type checking at compile time. TypeScript and Flow are popular type systems for React.

- **React Native:** React Native brings your application to mobile devices, such as iOS and Android applications. Once you've mastered React, the learning curve for React Native shouldn't be that steep, as they share the same principles. The only difference with mobile is the layout components.

- **Other Projects:** There are plenty of tutorials out there that use only React to build exciting applications, which provide good practice to apply what you've learned in this course before you move on to intermediate projects. Here are a few of my own:

    - A paginated and infinite scrolling list
    - Showcasing tweets on a Twitter wall
    - Connecting your React application to Stripe for charging money.

I invite you to visit my website to find more interesting topics about web development and software engineering. You can also subscribe to my Newsletter to get updates about articles and courses. Furthermore, the Road to React course platform offers more advanced lessons about the React ecosystem.

Finally, I hope to find more Patrons who are willing to support my content, as there are many students who cannot afford to pay for educational content. That's why I put lots of my content out there for free. Supporting me on Patreon allows me to educate others for free.

Thank you for taking the Road to learn React.

Regards,

Robin Wieruch