Outline

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

We've reached the end of the road to React, and I hope you enjoyed reading it, and that it helped you gain some traction in React. If you liked the course, share it with your friends who are interested in learning more about React. Also, a review on Amazon or Goodreads would help me provide better content in the future based on your feedback.

From here, I recommend you extend the application to create your own React projects before engaging another course, course or tutorial. Try it for a week, take it to production by deploying it, and reach out to me or others to showcase it. I am always interested in seeing what my readers built, and learning how I can help them along.

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

- **Connecting to a Database and/or Authentication:** Growing React applications will eventually require persistent data. The data should be stored in a database so that keeps it intact after browser sessions, to be shared with different users. Firebase is one of the simplest ways to introduce a database without writing a backend application. In my course titled "The Road to Firebase", you will find a step-by-step guide on how to use Firebase authentication and database in React.
- **Connecting to a Backend:** React handles frontend applications, and we've only requested data from a third-party backend's API thus far. You can also introduce an API with a backend application that connects to a database and manages authentication/authorization. In "The Road to GraphQL", I teach you how to use GraphQL for client-server communication. You'll learn how to connect your backend to a database, how to manage user sessions, and how to talk from a frontend to your backend application via a GraphQL API.

backend application via a GraphQL API.

- **State Management:** You have used React to manage local component state exclusively in this learning experience. It's a good start for most applications, but there are also external state management solutions for React. I explore the most popular one in my course "The Road to Redux".
- **Tooling with Webpack and Babel:** We used *create-react-app* to set up the application in this course. At some point you may want to learn the tooling around it, to create projects without *create-react-app*. I recommend a minimal setup with Webpack, after which you can apply additional tooling.
- **Code Organization:** Recall the chapter about code organization and apply these changes, if you haven't already. It will help organize your components into structured files and folders, and it will help you understand the principles of code splitting, reusability, maintainability, and module API design. Your application will grow and need structured modules eventually; so it's better to start now.
- **Testing:** We only scratched the surface of testing. If you are unfamiliar with testing web applications, dive deeper into unit testing and integration testing, especially with React applications. Cypress is a useful tool to explore for end-to-end testing in React.
- **Type Checking:** Earlier we used TypeScript in React, which is good practice to prevent bugs and improve developer experience. Dive deeper into this topic to make your JavaScript applications more robust. Maybe you end up using TypeScript instead of JavaScript all along.
- **UI Components:** Many beginners introduce UI component libraries like Bootstrap too early in their projects. It is more practical to 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 you cover the basic implementations of these crucial UI component, introducing a UI component library should be easier.
- **Routing:** You can implement routing for your application with react-

router. There is only one page in the application we've created, but that will grow. React Router helps manage multiple pages across multiple URLs. When you introduce routing to your application, no requests are made to the web server for the next page. The router handles this client-side.

- **React Native:** React Native brings your application to mobile devices like iOS and Android. 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 devices are the layout components, the build tools, and the APIs of your mobile device.

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 or Twitter page to get updates about articles, courses, and courses.

Thank you for reading the Road to React.

Regards,

Robin Wieruch 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