



Javascript

ADVANCED CONCEPTS

Introduction

With all the concepts we have learned in this section, you are more than ready to create your own React applications. However, there are a lot of other, more advanced concepts out there that we haven't discussed, but need to inform you of.

This lesson will be used to introduce you to a few advanced concepts and give you some tips on your path as a React developer. This section isn't meant to teach these concepts, but to give you a high-level overview to enable you to research in the event you need them or wish to learn more.

This lesson is meant as a checklist. Once you are secure with the basics, come back to this lesson and work through the advanced concepts. This doesn't have to be now. If you first want to move ahead in the curriculum and apply your knowledge or feel like you do not need a particular concept right now, just skip it and come back another time when you are ready.

As a friendly reminder, Google and StackOverflow are your friends! We should be familiar with these tools at our disposal at this point.

Good Luck!

Guide to Advanced React

1. PropTypes

One common discussion about JavaScript is whether it would be better if you could declare types for variables and properties. Many programmers agree that the pattern of declaring types, which you have in many other programming languages is preferable because it allows you to catch errors, such as passing a string to a variable that should be a number. Therefore, React provides the possibility to declare types using PropTypes. And if that's not enough for you, you could also use [TypeScript](#) with React.

2. Styled Components

If you start writing larger applications you will, with certainty, encounter the problem that you might want to style buttons or anything else across your application similarly. One way you could solve this problem is through code duplication (writing even the most basic CSS for each button). However, this wouldn't be a very good (or



code duplication (writing even the most basic CSS for each button). However, this wouldn't be a very good (or clean) code. The styled-components package provides a cleaner way to do this. It allows you to give some default stylings to HTML elements, meaning you define a button with some basic styling and reuse this button throughout your application. This way there is no code duplication and it allows your application to be more scalable.

3. Redux

You might have already heard about Redux. Redux is the most popular state management system out there. It is not a part of React, but the two can be very easily combined. Together, they make up an extremely powerful duo. The purpose is to store your application's state in a single place, commonly called a "store". You then dispatch actions to the store, where a reducer will handle the state changes. The primary benefit of using a state management library is to prevent having to pass props through multiple levels of the component tree. A state management library is often only recommended for larger applications.

4. Higher-order Components

Higher-order components are components that consume another component and return a third component.

5. withRouter (history, match, children object)

`withRouter` is one example of a higher-order component and is worth looking into it. By wrapping `withRouter` around one of your components, the component will get access to your history, match, and children objects, which provide some additional syntactic sugar. For example, you could use the history object to push a user from one route to another.

6. Hooks (We only discussed a few, but there are a lot more out there)

There are a lot of hooks out there, with the number only increasing by the day (since you can write your own!). As you have seen in previous lessons, it seems like the React team wants us to use more functional components with hooks in the future, so it is definitely worth getting to know both the hooks that are built-in to React, as well as how to create your own.

[View Course](#)

[Login to track progress](#)

[Next Lesson](#)



 [Improve this lesson on GitHub](#)

Have a question?

Chat with our friendly Odin community in our Discord chatrooms!

[Open Discord](#)

Are you interested in accelerating your web development learning experience?

[Get started](#)



5-6 months



Job Guarantee



1-on-1 Mentorship



[THE ODIN PROJECT](#)



[About](#)
[FAQ](#)
[Blog](#)

[Success Stories](#)
[Contribute](#)
[Terms of Use](#)

