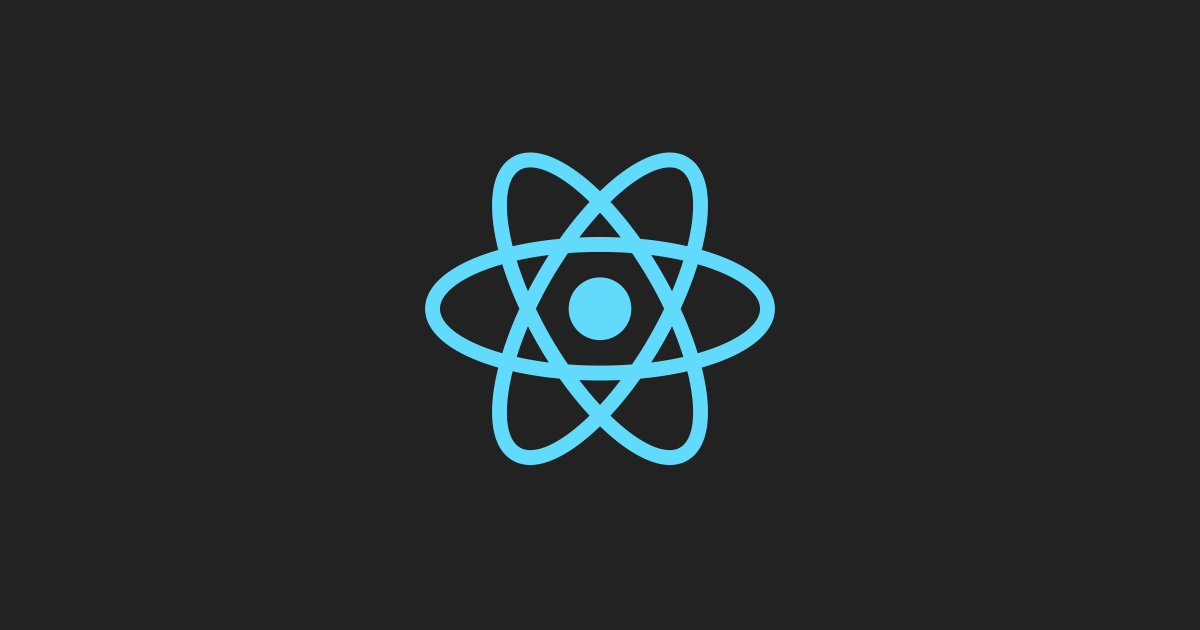
React Documentation



Contents

[Installation & Generating/Starting Project 3](#_Toc114221349)

[React Router 4](#_Toc114221350)

[Api calls & async 5](#_Toc114221351)

# Installation & Generating/Starting Project

In order to use React, you’re going to need NPM and Nodejs, so download & install those on your system before you proceed. Generating a react project is done through the terminal, generating a project can be done with the following command:



That will create a react project, using Javascript. There’s also a way to generate it with Typescript, instead of Javascipt, using the following command:



If you receive a react project from someone, it will likely not include the “node\_modules” folder. To be able to run the project, you’re going to have to generate the folder, using the following command:



Then finally, to run your project all you have to type into the terminal is:



The terminal will (assuming it was able to start the project,) show you a URL through which you can open your app, click on URL or simply copy + paste it into your browser.

# React Router

React Router is a package that can be downloaded into your react project through the following command:



You can use it to navigate through your react app. After installing you have to do some simple setting-up, but after you’re good to go.

Start with “index.js”. This is the file in which we’re going setup the routing itself, define routes, etc. First, start with importing everything needed to be able to use React Router:



After that you can start defining your routes within the “root.render()” method. You begin by opening and closing the BrowserRouter-tag, which is a router for use in web browsers. Inside that open and close the Routes-tag, a container for a tree of elements.

For each route you want to define, create a Route-tag. The most important properties of Route that we are interested in are path and element. The path is the….. path on which you want to setup the route, when you go to that specific route In your app, you want to display a specific component-screen, that is specific in the element-property. So you need to create a screen/page, after which you can import it in “index.js” and name it as the element to be rendered when routing to a path.



When you want to navigate to a different screen at the end of a method for example, there’s a couple of ways to navigate programmatically and one with button-click.

First the button-click method. To do that you need the Link-tag. Usually you’ll have a piece of text or a div you want to make clickable, just wrap that element in the Link-tag and give it the to-property. The to-property takes the path you want to navigate to. If you want you can give it a state-property, if takes some data in case you want to send data to the other page.



You can use  to access the data.

Component navigation method. All you need is to import the Navigate-tag and pass it a to-property.



Hook method. A hook is a special function, you can use “navigate()” and pass it a URL to navigate to.



# Api calls & async

To send and get data to/from the database, you need to make some API calls. That can be done with the built-in fetch() API or Axios, an npm package. In this project I used Axios, before you can use it, you have to install it:



After that, import Axios in the component you’re going to use it in:



In my methods I used async and await. Putting “async” before a function makes the function return a promise. Using “await” makes JavaScript wait until the promise returns a result.

Short explanation of what a promise is by MDN web docs:

*The****Promise****object represents the eventual completion (or failure) of an asynchronous operation and its resulting value.*

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise>

To get some data from my API, I create a function with “async” before the “function” keyword. I then use the .get() method from Axios to specify which URL I want to send request to, in this case a GET-request. But before that statement, be sure to use the “await” keyword. After the response is stored in, in this case, “const response”, you can do whatever you want with it. Below I store it in a state.



You can also send a POST-request, in case you need to save some data into your database for example. The code for that is mostly the same, although you obviously don’t want to use .get() for that but .post(). And since you want to *send* some data as well, you need to include a body with that data. That data will then be used by your API to do whatever you want with it.

