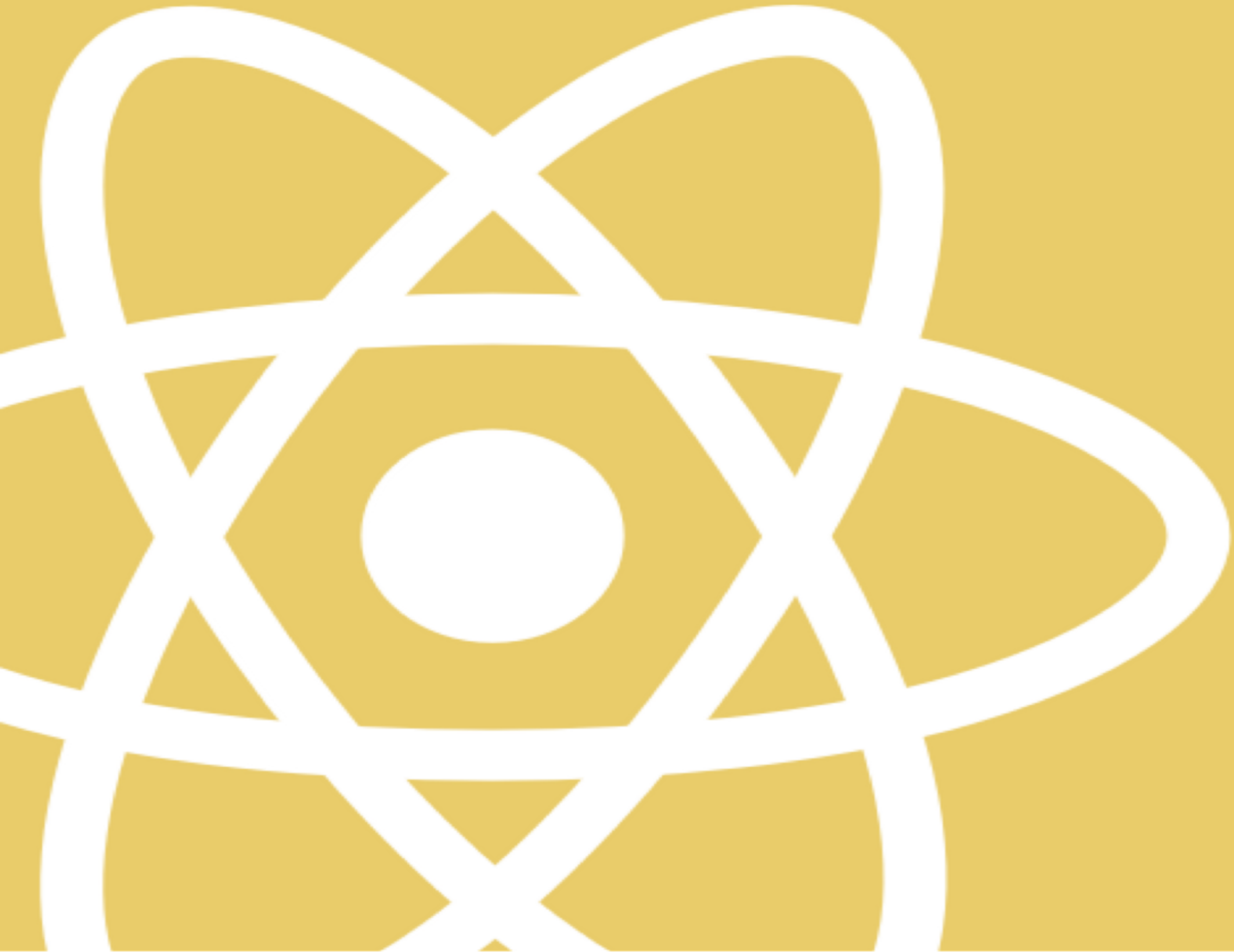
by robin wieruch

the Road to learn React



The Road to learn React

Your journey to master plain yet pragmatic React

Robin Wieruch

This book is for sale at http://leanpub.com/the-road-to-learn-react

This version was published on 2018-10-03

Tweet This Book!

Please help Robin Wieruch by spreading the word about this book on Twitter!

The suggested tweet for this book is:

I am going to learn #ReactJs with The Road to learn React by @rwieruch Join me on my journey https://roadtoreact.com

The suggested hashtag for this book is #ReactJs.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#ReactJs

Contents

Foreword

The Road to learn React teaches the fundamentals of React. You will build a real-world application in plain React without complicated tooling. Everything from project setup to deployment on a server will be explained for you. The book comes with additional referenced reading material and exercises with each chapter. After reading the book, you will be able to build your own applications in React. The material is kept up to date by myself and the community.

In the Road to learn React, I offer a foundation before you dive into the broader React ecosystem. The concepts will have less tooling and less external state management, but a lot of information about React. It explains general concepts, patterns, and best practices in a real world React application.

Essentially, you will learn to build your own React application from scratch, with features like pagination, client-side caching, and interactions like searching and sorting. Additionally, you will transition from JavaScript ES5 to JavaScript ES6. I hope this book captures my enthusiasm for React and JavaScript, and that it helps you get started with it.

About the Author

I am a German software and web engineer dedicated to learning and teaching programming in JavaScript. After obtaining my Master's Degree in computer science, I continued learning on my own. I gained experience from the startup world, where I used JavaScript intensively during both my professional life and spare time, which eventually led to a desire to teach others about these topics.

For a few years, I worked closely with an exceptional team of engineers at a company called Small Improvements, developing large scale applications. The company offered a SaaS product that enables customers to give feedback to businesses. This application was developed using JavaScript on its frontend, and Java as its backend. The first iteration of Small Improvements' frontend was written in Java with the Wicket Framework and jQuery. When the first generation of SPAs became popular, the company migrated to Angular 1.x for its frontend application. After using Angular for over two years, it became clear that Angular wasn't the best solution to work with state intense applications, so they made the jump to React and Redux. This enabled it to operate on a large scale successfully.

During my time in the company, I regularly wrote articles about web development on my website. I received great feedback from people learning from my articles which allowed me to improve his writing and teaching style. Article after article, I grew my ability to teach others. I felt that my first articles were packed with too much information, quite overwhelming for students, but I improved by focusing on one subject at a time.

Currently, I am a self-employed software engineer and educator. I find it a fulfilling pastime to see students thrive by giving them clear objectives and short feedback loops. You can find more information about me and ways to support and work with me on my website[1].

[1]https://www.robinwieruch.de/about

Testimonials

There are many testimonials[2], ratings[3] and reviews[4] about the book that you can read to ascertain its quality. I am proud of it, and I never expected such overwhelming feedback. I would love to find your rating/review. It helps me to spread the word about the book and make improvements for future projects. The following shows a short excerpt of these voices:

Muhammad Kashif[5]: "The Road to Learn React is a unique book that I recommend to any student or professional interested in learning react basics to advanced level. It is packed with insightful tips and techniques that are hard to find elsewhere, and remarkably thorough in its use of examples and references to sample problems, i have 17 years of experience in web and desktop app development, and before reading this book i was having trouble in learning react, but this book works like magic."

Andre Vargas[6]: "The Road to Learn React by Robin Wieruch is such an awesome book! Most of what I learned about React and even ES6 was through it!"

Nicholas Hunt-Walker, Instructor of Python at a Seattle Coding School[7]: "This is one of the most well-written & informative coding books I've ever worked through. A solid React & ES6 introduction."

Austin Green[8]: "Thanks, really loved the book. Perfect blend to learn React, ES6, and higher level programming concepts."

Nicole Ferguson[9]: "I'm doing Robin's Road to Learn React course this weekend & I almost feel guilty for having so much fun."

Karan[10]: "Just finished your Road to React. Best book for a beginner in the world of React and JS. Elegant exposure to ES. Kudos! :)"

Eric Priou[11]: "The Road to learn React by Robin Wieruch is a must read. Clean and concise for React and JavaScript."

[2]https://roadtoreact.com/
[3]https://www.goodreads.com/book/show/37503118-the-road-to-learn-react
[4]https://www.amazon.com/dp/B077HJFCQX
[5]https://twitter.com/appsdevpk/status/848625244956901376
[6]https://twitter.com/andrevar66/status/853789166987038720
[7]https://twitter.com/nhuntwalker/status/845730837823840256
[8]https://twitter.com/AustinGreen/status/845321540627521536
[9]https://twitter.com/nicoleffe/status/833488391148822528
[10]https://twitter.com/kvss1992/status/889197346344493056
[11]https://twitter.com/erixtekila/status/840875459730657283

Education for Children

The book should enable everyone to learn React. However, not everyone has access to the required resources, because not everyone is educated in the English language. I want to use this project to support other projects that teach children English in the developing world.

- April to 18. April, 2017, Giving Back, By Learning React[12]

[12]https://www.robinwieruch.de/giving-back-by-learning-react/

FAQ

How to get updates? I have two channels where I share updates about my content. You can subscribe to updates by email[13] or follow me on Twitter[14]. Regardless of the channel, my objective is to only share quality content. Once you receive notification the book has changed, you can download a new version of it.

Does it use the recent React version? The book always receives an update when the React version is updated. Programming books are usually outdated soon after their release, but since this book is self-published, I can update it as needed.

Does it cover Redux? It doesn't, so I have written a second book. The Road to learn React should give you a solid foundation before you dive into advanced topics. The implementation of the sample application in the book will show that you don't need Redux to build an application in React. After you have read the book, you should be able to implement a solid application without Redux. Then you can read my second book, Taming the State in React, to learn Redux.

Does it use JavaScript ES6? Yes. Don't worry, though, you will be fine if you are familiar with JavaScript ES5. All JavaScript ES6 features, that I describe in The Journey to Learn React, will transition from ES5 to ES6. The book does not only teach React, but also all useful JavaScript ES6 features.

How to get access to the source code projects and screencasts series? If you bought one of the extended packages that grant access to the source code projects, screencast series or any other add-on, you should find these on your course dashboard[15]. If you bought the course other than the official Road to React[16] course platform, create an account on the platform, and then go to the Admin page and contact me with one of the email templates. After that I can enroll you in the course. If you haven't bought one of the extended packages, you can reach out any time to upgrade your content to access the source code projects and screencast series.

Can I get a copy of the book if I bought it on Amazon? If you have bought the book on Amazon, you may have seen that the book is available on my website too. Since I use Amazon as one way to monetize my often free content, I honestly thank you for your support and invite you to sign up on Road to React[17]. There you can write me an email (Admin page) about your purchase, so that I can unlock the whole course package for you. In addition, you can always download the latest ebook version of the book on the platform.

How can I get help while reading the book? The book has a Slack Group[18] for people who are reading along. You can join the channel to get help, or to help others, as helping others may help

[13]https://www.getrevue.co/profile/rwieruch
[14]https://twitter.com/rwieruch
[15]https://roadtoreact.com/my-courses
[16]https://roadtoreact.com
[17]https://roadtoreact.com
[18]https://slack-the-road-to-learn-react.wieruch.com/

you internalize your own understanding. If there is no one available to help you, you can always reach out to me.

Is there any troubleshoot area? If you run into problems, please join the Slack Group. Also, check the open issues on GitHub[19] to see if any solutions are listed for specific issue. If your problem wasn't mentioned, open a new issue where you can explain your problem, provide a screenshot, and offer more details (e.g. book page, node version).

Can I help to improve the content? Yes, I love to hear feedback. Simply open an issue on GitHub[20]. These can be technical improvements, or clarification on the discussed topics. I am not a native speaker, so any feedback is appreciated. You can open pull requests on the GitHub page as well.

Is there a money back guarantee? Yes, there is 100% money back guarantee for two months if you don't think it's a good fit. Please reach out to me to get a refund.

How to support the project? If you believe in the content I create, you can support me[21]. It also helps if you spread the word about this book, or you can sign on as my Patron on Patreon[22].

What's your motivation behind the book? I want to teach about this topic consistently. I often find materials online that don't receive update, or only applies to a small part of a topic. Sometimes people struggle to find consistent and up-to-date resources to learn from. I want to provide this consistent and up-to-date learning experience. Also, I hope I can support the less fortunate with my projects by giving them the content for free or by having other impacts[23]. Recently I'v found myself fulfilled when teaching others about programming, as it's a meaningful activity I prefer over any 9 to 5 job at any company. I hope to continue this path in the future.

Is there a call to action? Yes. I want you to take a moment to think about a person who would be a good match to learn React. The person could have shown the interest already, could be in the middle of learning React or might not yet be aware about wanting to learn React. Reach out to that person and share the book. It would mean a lot to me. The book is intended to be given to others.

[19]https://github.com/rwieruch/the-road-to-learn-react/issues
[20]https://github.com/rwieruch/the-road-to-learn-react
[21]https://www.robinwieruch.de/about/
[22]https://www.patreon.com/rwieruch
[23]https://www.robinwieruch.de/giving-back-by-learning-react/

Change Log

10. January 2017:

- v2 Pull Request[24]
- even more beginner friendly
- 37% more content
- 30% improved content
- 13 improved and new chapters
- 140 pages of learning material
- + interactive course of the book on educative.io[25]

08. March 2017:

- v3 Pull Request[26]
- 20% more content
- 25% improved content
- 9 new chapters
- 170 pages of learning material

15. April 2017:

- upgrade to React 15.5

5. July 2017:

- upgrade to node 8.1.3
- upgrade to npm 5.0.4
- upgrade to create-react-app 1.3.3

17. October 2017:

- upgrade to node 8.3.0
- upgrade to npm 5.5.1
- upgrade to create-react-app 1.4.1

[24] https://github.com/rwieruch/the-road-to-learn-react/pull/18
[25] https://www.educative.io/collection/5740745361195008/5676830073815040
[26] https://github.com/rwieruch/the-road-to-learn-react/pull/34

- upgrade to React 16
- v4 Pull Request[27]
- 15% more content
- 15% improved content
- 3 new chapters (Bindings, Event Handlers, Error Handling)
- 190+ pages of learning material
- +9 Source Code Projects[28]

17. February 2018:

- upgrade to node 8.9.4
- upgrade to npm 5.6.0
- upgrade to create-react-app 1.5.1
- v5 Pull Request[29]
- more learning paths
- extra reading material
- 1 new chapter (Axios instead of Fetch)

31. August 2018:

- professional proofreading and editing by Emmanuel Stalling
- 16 Source Code Projects[30]
- v6 Pull Request[31]

3. October 2018:

- upgrade to node 10.11.0
- upgrade to npm 6.4.1
- upgrade to create-react-app 2.0.2

[27]https://github.com/rwieruch/the-road-to-learn-react/pull/72
[28]https://roadtoreact.com
[29]https://github.com/the-road-to-learn-react/the-road-to-learn-react/pull/105
[30]https://roadtoreact.com
[31]https://github.com/the-road-to-learn-react/the-road-to-learn-react/pull/172

Challenge

Personally I write a lot about my learnings. That's how I got where I am right now. You are teaching a topic at its best when you have just learned about it yourself. Since teaching helped me a lot in my career, I want you to experience the same effects of it. But first you have to do the cause: teaching yourself. My challenge for the book is the following: teach others what you are learning while reading the book. A couple of breakpoints on how you could achieve it:

- Write a blog post about a specific topic from the book. It's not about copying and pasting the material but rather about teaching the topic your own way. Find your own words to explain something, grab a problem and solve it, and dive even more into the topic by understanding every detail about it. Then teach it to others in this one article. You will see how it fills your knowledge gaps, because you have to dig deeper into the topic, and how it opens doors for your career in the long term.
- If you are active on social media, grab a couple of things you have learned while reading the book and share them with your friends. For instance, you can tweet a hot tip on Twitter about your last learning from the book which may be interesting for others too. Just take a screenshot of the passage of the book or even better: write about it in your own words. That's how you can get into teaching without investing much time.
- If you feel confident recording your learning adventure, share your way through the book on Facebook Live, YouTube Live or Twitch. It helps you to stay concentrated and work your way through the book. Even though you don't have many people following your live session, you can always use the video to put it on YouTube afterward. Besides it is a great way to verbalize your problems and how you are going to solve them.

I would love to see people doing especially the last breakpoint: record yourself while reading this book, implementing the application(s), and conducting the exercises, and put the final version on YouTube. If parts in between of the recording are taking longer, just cut the video or use a timelapse effect for them. If you get stuck and need to fix a bug, don't leave it out but rather include these passages in the video, because they are so valuable for your audience which may run into the same issues. I believe it is important to have these parts in your video. A couple of tipps for the video:

- Do it in your native language or in English if you feel comfortable with it.
- Verbalize your thoughts, the things you are doing or the problems you are running into. Having a visual video is only one part of the challenge, but the other part is you narrating through the implementation. It doesn't have to be perfect. Instead it should feel natural and not polished as all the other video courses online where nobody runs into problems.

- If you run into bugs, embrace the trouble. Try to fix the problem yourself and search online for help, don't give up, and speak about the problem and how you attempt to solve it. This helps others to follow your thought process. As I said before, it has no value to follow polished video courses online where the instructor never runs into problems. It's the most valuable part to see someone else fixing a bug in the source code.
- Some words about the technical side of the recording: Check your audio before you record a longer video. It should have the correct volume and the quality should be alright too. Regarding your editor/IDE/terminal, make sure to increase the font size. Maybe it is possible to place the code and the browser side by side. If not, make them fullscreen and switch between (e.g. MacOS CMD + Tab).
- Edit the video yourself before you put it on YouTube. It doesn't have to be a high quality, but you should try to keep it concise for your audience (e.g. leaving out the reading passages and rather summarize the steps in your own words).

In the end, you can reach out to me for promoting anything you have released. For instance, if the video turns out well, I would love to include it in this book as officially supplementary material. Just reach out to me once you finished it. After all, I hope you accept this challenge to enhance your learning experience while reading the book which also may help others. I wish you all the best for it.

Introduction to React

This chapter is an introduction to React, a JavaScript library for rendering interfaces in single-page and mobile applications, where I explain why developers should consider adding the React library to their toolbelts. We will dive into the React ecosystem, creating your first React application from scratch with no configuration. Along the way, we will introduce **JSX**, the syntax for React, and **ReactDOM**, so you have an understanding of React's practical uses in modern web applications.

Hi, my name is React.

Single page applications (SPA[32]) have become increasingly popular in recent years, as frameworks like Angular, Ember, and Backbone allow JavaScript developers to build modern web applications using techniques beyond vanilla JavaScript and jQuery. The three mentioned are among the first SPAs, each coming into its own between 2010 and 2011, but there are many more options for single-page development. The first generation of SPA frameworks arrived at the enterprise level, so their frameworks are more rigid. React, on the other hand, remains an innovative library that has been adopted by many technological leaders like Airbnb, Netflix, and Facebook[33].

React was released by Facebook's web development team in 2013 as a view library, which makes it the 'V' in the MVC[34] (model view controller). As a view, it allows you to render components as viewable elements in a browser, while its ecosystem lets us build single page applications. While the first generation of frameworks tried to solve many things at once, React is only used to build your view layer; specifically, it is a library wherein the view is a hierarchy of composable components. If you haven't heard about MVC before, don't bother about it, because it's just there to put React historically into context for people who come from other programming languages.

In React, the focus remains on the view layer until more aspects are introduced to the application. These are the building blocks for an SPA, which are essential to build a mature application. They come with two advantages:

- You can learn the building blocks one at a time without having to understand them altogether. In contrast, an SPA framework gives you every building block from the start. This book focuses on React as the first building block. More building blocks will eventually follow.
- All building blocks are interchangeable, which makes the ecosystem around React highly innovative. Multiple solutions can compete with each other, and you can choose the most appealing solution for any given challenge.

React is one of the best choices for building modern web applications. Again, it only delivers the view layer, but the surrounding ecosystem makes up an entirely flexible and interchangeable framework. React has a slim API, a robust and evolving ecosystem, and a great community.

Exercises

If you'd like to know more about why I chose React, or to find a more in-depth look at the topics mentioned above, these articles grant a deeper perspective:

- Why I moved from Angular to React[35]

[32]https://en.wikipedia.org/wiki/Single-page_application
[33]https://github.com/facebook/react/wiki/Sites-Using-React
[34]https://en.wikipedia.org/wiki/Modelâ€"viewâ€"controller
[35]https://www.robinwieruch.de/reasons-why-i-moved-from-angular-to-react/

- React's flexible ecosystem[36]
- How to learn a framework[37]

[36]https://www.robinwieruch.de/essential-react-libraries-framework/
[37]https://www.robinwieruch.de/how-to-learn-framework/

Requirements

To follow this book, you should be familiar with the basics of web development, i.e how to use HTML, CSS, and JavaScript. It also makes sense to understand how APIs[38] work, as they will be covered thoroughly. Also, I encourage you to join the official Slack Group[39] to be a part of a growing React community where you can learn from and help others.

Editor and Terminal

For the lessons, you will need a text editor or an IDE and terminal (command line tool). I have provided a setup guide[40] if you need additional help. Optionally, we recommend you keep your projects in GitHub while conducting the exercises in this book. There is a short guide[41] on how to use these tools. Github has excellent version control, so you can see what changes were made if you make a mistake or just want a more direct way to follow along.

Node and NPM

Finally, you will need an installation of node and npm[42]. Both are used to manage libraries you will need along the way. In this book, you will install external node packages via npm (node package manager). These node packages can be libraries or whole frameworks.

You can verify your versions of node and npm on the command line. If you don't get any output in the terminal, you need to install node and npm first. These are my versions at the time of writing this book:

Command Line

```
node --version
*v10.11.0
npm --version
*v6.4.1
```

The additional content of this section is a crash course in node and npm. It is not exhaustive, but it will cover all of the necessary tools. If you are familiar with both of them, you can skip this section.

The **node package manager** (npm) installs external node packages from the command line. These packages can be a set of utility functions, libraries, or whole frameworks, and they are the dependencies of your application. You can either install these packages to your global node package folder, or to your local project folder.

Global node packages are accessible from everywhere in the terminal, and only need to be installed to the global directory once. Install a global package by typing the following into a terminal:

[38]https://www.robinwieruch.de/what-is-an-api-javascript/
[39]https://slack-the-road-to-learn-react.wieruch.com/
[40]https://www.robinwieruch.de/developer-setup/
[41]https://www.robinwieruch.de/git-essential-commands/
[42]https://nodejs.org/en/

Command Line

```
npm install -g <package>
```

The `-g` flag tells npm to install the package globally. Local packages are used in your application by default. For our purposes, we will install React to the local directory terminal by typing:

Command Line

```
npm install react
```

The installed package will automatically appear in a folder called *node_modules/* and will be listed in the *package.json* file next to your other dependencies.

To initialize the *node_modules/* folder and the *package.json* file for your project, use the following npm command. Then, you can install new local packages via npm:

Command Line

```
npm init -y
```

The `-y` flag initializes all the defaults in your *package.json*. After initializing your npm project, you are ready to install new packages via `npm install <package>`.

The *package.json* file allows you to share your project with other developers without sharing all the node packages. It will contain references to all node packages used in your project, called **dependencies**. Other users can copy a project without the dependencies using the references in *package.json*, where the references make it easy to install all packages using `npm install`. A `npm install` script will take all the dependencies listed in the *package.json* file and install them in the *node_modules/* folder.

Finally, there's one more command to cover about npm:

Command Line

```
npm install --save-dev <package>
```

The `--save-dev` flag indicates that the node package is only used in the development environment, meaning it won't be used in when the application is deployed to a the server or used in production. It is useful for testing an application using a node package, but want to exclude it from your production environment.

Some of you may want to use other package managers to work with node packages in your applications. **Yarn** is a dependency manager that works similar to **npm**. It has its own list of commands, but you still have access to the same npm registry. Yarn was created to solve issues npm couldn't, but both tools have evolved to the point where either will suffice today.

Exercises:

- Set up a throw away npm project using the terminal:
  - Create a new folder with `mkdir <folder_name>`
  - Navigate into the folder with `cd <folder_name>`
  - Execute `npm init -y` or `npm init`
  - Install a local package like React with `npm install react`
  - Check the *package.json* file and the *node_modules/* folder
  - Attempt to uninstall and reinstall the *react* node package
- Read about npm[43]
- Read about yarn[44] package manager

[43]https://docs.npmjs.com/
[44]https://yarnpkg.com/en/docs/

Installation

There are many approaches to getting started with a React application. The first we'll explore is a CDN, short for content delivery network[45]. Don't worry too much about CDNs now, because you will not use them in this book, but it makes sense to explain them briefly. Many companies use CDNs to host files publicly for their consumers. Some of these files are libraries like React, since the bundled React library is just a *react.js* JavaScript file.

To get started with React by using a CDN, find the `<script>` tag in your web page HTML that points to a CDN url. You will need two libraries: *react* and *react-dom*.

Code Playground

```
<script
  crossorigin
  src="https://unpkg.com/react@16/umd/react.development.js"
></script>

<script
  crossorigin
  src="https://unpkg.com/react-dom@16/umd/react-dom.development.js"
></script>
```

You can also get React into your application by initializing it as node project. With a *package.json* file, you can install *react* and *react-dom* from the command line. However, the folder must be initialized as a npm project using `npm init -y` with a *package.json* file. You can install multiple node packages with npm:

Command Line

```
npm install react react-dom
```

This approach is often used to add React to an existing application managed with npm.

You may also have to deal with Babel[46] to make your application aware of JSX (the React syntax) and JavaScript ES6. Babel transpiles your code–that is, it converts it to vanilla JavaScript–so most modern browsers can interpret JavaScript ES6 and JSX. Because of this difficult setup, Facebook introduced *create-react-app* as a zero-configuration React solution. The next section will show you how to setup your application using this bootstrapping tool.

Exercises:

- Read about React installations[47]

[45]https://en.wikipedia.org/wiki/Content_delivery_network
[46]http://babeljs.io/
[47]https://reactjs.org/docs/getting-started.html

Zero-Configuration Setup

In the Road to learn React, we will be using create-react-app[48] to bootstrap your application. It's an opinionated yet zero-configuration starter kit for React introduced by Facebook in 2016, recommended for beginners by 96% of React users[49]. In *create-react-app* the tooling and configuration evolve in the background, while the focus is on the application implementation.

To get started, install the package to your global node packages, which keeps it available on the command line to bootstrap new React applications:

Command Line

```
npm install -g create-react-app
```

You can check the version of *create-react-app* to verify a successful installation on your command line:

Command Line

```
create-react-app --version
*v2.0.2
```

Now you are ready to bootstrap your first React application. The example will be referred to as *hackernews*, but you may choose any name you like. First, navigate into the folder:

Command Line

```
create-react-app hackernews
cd hackernews
```

Now you can open the application in your editor. The following folder structure, or a variation of it depending on the *create-react-app* version, should be presented to you:

Folder Structure

```
hackernews/
  README.md
  node_modules/
  package.json
  .gitignore
  public/
    favicon.ico
    index.html
    manifest.json
```

[48]https://github.com/facebookincubator/create-react-app
[49]https://twitter.com/dan_abramov/status/806985854099062785

```
src/
  App.css
  App.js
  App.test.js
  index.css
  index.js
  logo.svg
  serviceWorker.js
```

This is a breakdown of the folders and files:

- **README.md**: The .md extension indicates the file is a markdown file. Markdown is used as a lightweight markup language with plain text formatting syntax. Many source code projects come with a *README.md* file to give you initial instructions about the project. When pushing your project to a platform such as GitHub, the *README.md* file usually displays information about the content contained in the repository. Because you used *create-react-app*, your *README.md* should be the same as the official create-react-app GitHub repository[50].
- **node_modules/**: This folder contains all node packages that have been installed via npm. Since you used *create-react-app*, there should already be a couple of node modules installed for you. You will rarely touch this folder, because node packages are generally installed and uninstalled with npm from the command line.
- **package.json**: This file shows you a list of node package dependencies and other project configurations.
- **.gitignore**: This file displays all files and folders that shouldn't be added to your git repository when using git; such files and folders should only be located in your local project. The *node_-modules/* folder is one example. It is enough to share the *package.json* file with others, so they can install dependencies on their end with `npm install` without your dependency folder.
- **public/**: This folder holds development files, such as *public/index.html.* The index is displayed on localhost:3000 when developing your app. The boilerplate takes care of relating this index with all the scripts in *src/*.
- **build/** This folder is created when you build the project for production, as it holds all of the production files. When building your project for production, all the source code in the *src/* and *public/* folders are bundled and placed in the build folder.
- **manifest.json** and **serviceWorker.js**: These files won't be used for this project, so you can ignore them for now.

In the beginning, everything you need is located in the *src/* folder. The main focus lies on the *src/App.js* file which is used to implement React components. It will be used to implement your application, but later you might want to split up your components into multiple files, where each file maintains one or more components on its own.

[50]https://github.com/facebookincubator/create-react-app

Additionally, you will find a *src/App.test.js* file for your tests, and a *src/index.js* as an entry point to the React world. You will get to know both files intimately in a later chapter. There is also a *src/index.css* and a *src/App.css* file to style your general application and components, which comes with the default style when you open them. You will modify them later as well.

The *create-react-app* application is a npm project you can use to install and uninstall node packages. It comes with the following npm scripts for your command line:

Command Line

```
# Runs the application in http://localhost:3000
npm start

# Runs the tests
npm test

# Builds the application for production
npm run build
```

The scripts are defined in your *package.json*, and your basic React application is bootstrapped. The following exercises will finally allow you to run your bootstrapped application in a browser.

Exercises:

- `npm start` your application and visit the application in your browser (Exit the command by pressing Control + C)
- Run the `npm test` script
- Run the `npm run build` script and verify that a *build/* folder was added to your project (you can remove it afterward. Note that the build folder can be used later on to deploy your application[51])
- Familiarize yourself with the folder structure
- Check the content of the files
- Read about npm scripts and create-react-app[52]

[51]https://www.robinwieruch.de/deploy-applications-digital-ocean/
[52]https://github.com/facebookincubator/create-react-app

Introduction to JSX

Now we will get to know JSX, the syntax in React. As mentioned before, *create-react-app* has already bootstrapped a basic application for you, and all files come with their own default implementations. For now, the only file we will modify is the *src/App.js* file.

src/App.js

```
import React, { Component } from 'react';
import logo from './logo.svg';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <header className="App-header">
          <img src={logo} className="App-logo" alt="logo" />
          <p>
            Edit <code>src/App.js</code> and save to reload.
          </p>
          <a
            className="App-link"
            href="https://reactjs.org"
            target="_blank"
            rel="noopener noreferrer"
          >
            Learn React
          </a>
        </header>
      </div>
    );
  }
}

export default App;
```

Don’t worry if you’re confused by the import/export statements and class declaration now. These are features of JavaScript ES6 we will revisit in a later chapter.

In the file you should see a **React ES6 class component** with the name App. This is a component declaration. After you have declared a component, you can use it as an element anywhere in your application. It will produce an **instance** of your **component** or, in other words, the component gets instantiated.

Code Playground

```
// component declaration
class App extends Component {
  ...
}

// component usage (also called instantiation for a class)
// creates an instance of the component
<App />
```

The returned **element** is specified in the `render()` method. The components you instantiated earlier are made up of elements, so it is important to understand the differences between a component, an instance of a component, and an element.

You should see where the App component is instantiated, else you couldn't see the rendered output in a browser. The App component is only the declaration, but not the usage. You can instantiate the component anywhere in your JSX with `<App />`. You will see later where this happens in this application.

The content in the render block may look similar to HTML, but it is actually JSX. JSX allows you to mix HTML and JavaScript. It is powerful, but it can be confusing when you are used to separating the two languages. It is a good idea to start by using basic HTML in your JSX. Open the `App.js` file and remove all unnecessary HTML code as shown:

src/App.js

```
import React, { Component } from 'react';
import './App.css';

class App extends Component {
  render() {
    return (
      <div className="App">
        <h2>Welcome to the Road to learn React</h2>
      </div>
    );
  }
}

export default App;
```

Now, you only return HTML in your `render()` method without any JavaScript. Let's define the "Welcome to the Road to learn React" as a variable. A variable is set in JSX by curly braces.

src/App.js

```
import React, { Component } from 'react';
import './App.css';

class App extends Component {
  render() {
    var helloWorld = 'Welcome to the Road to learn React';
    return (
      <div className="App">
        <h2>{helloWorld}</h2>
      </div>
    );
  }
}

export default App;
```

Start your application on the command line with `npm start` to verify the changes you've made.

You might have noticed the `className` attribute. It reflects the standard `class` attribute in HTML. JSX had replaced a handful of internal HTML attributes, but you can find all the supported HTML attributes in React's documentation[53], which all follow the camelCase convention. On your way to learn React, expect to run across more JSX specific attributes.

Exercises:

- Define more variables and render them in JSX
  - Use a complex object to represent a user with a first name and last name
  - Render the user properties in JSX
- Read about JSX[54]
- Read about React components, elements and instances[55]

[53] https://reactjs.org/docs/dom-elements.html#all-supported-html-attributes
[54] https://reactjs.org/docs/introducing-jsx.html
[55] https://reactjs.org/blog/2015/12/18/react-components-elements-and-instances.html

ES6 const and let

Notice that we declared the variable `helloWorld` with a `var` statement. JavaScript ES6 comes with two more ways to declare variables: `const` and `let`. In JavaScript ES6, you will rarely find `var` anymore. A variable declared with `const` cannot be re-assigned or re-declared, and cannot be changed or modified. Once the variable is assigned, you cannot change it:

Code Playground

```
// not allowed
const helloWorld = 'Welcome to the Road to learn React';
helloWorld = 'Bye Bye React';
```

Conversely, a variable declared with `let` can be modified:

Code Playground

```
// allowed
let helloWorld = 'Welcome to the Road to learn React';
helloWorld = 'Bye Bye React';
```

TIP: Declare variables with `let` if you think you'll want to re-assign it later on.

Note that a variable declared directly with `const` cannot be modified. However, when the variable is an array or object, the values it holds can get updated through indirect means:

Code Playground

```
// allowed
const helloWorld = {
  text: 'Welcome to the Road to learn React'
};
helloWorld.text = 'Bye Bye React';
```

There are varying opinions about when to use *const* and when to use *let*. I would recommend using `const` whenever possible to show the intent of keeping your data structures immutable, so you only have to worry about the values contained in objects and arrays. Immutability is embraced in the React ecosystem, so `const` should be your default choice when you define a variable, though it's not really about immutability, but about assigning variables only once. It shows the intent of not changing (re-assigning) the variable even though its content can be changed.

src/App.js

```
import React, { Component } from 'react';
import './App.css';

class App extends Component {
  render() {
    const helloWorld = 'Welcome to the Road to learn React';
    return (
      <div className="App">
        <h2>{helloWorld}</h2>
      </div>
    );
  }
}

export default App;
```

In your application, we will use `const` and `let` over `var` for the rest of the book.

Exercises:

- Read about ES6 const[56]
- Read about ES6 let[57]
- Gain an understanding of immutable data structures:
  - Why do they make sense in programming?
  - Why are they embraced in React and its ecosystem?

[56]https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/const
[57]https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/let

ReactDOM

The App component is located in your entry point to the React world: the *src/index.js* file.

src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import App from './App';
import './index.css';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);
```

`ReactDOM.render()` uses a DOM node in your HTML to replace it with JSX. It's a way to integrate React in any foreign application easily, and you can use `ReactDOM.render()` multiple times across your application. You can use it to bootstrap simple JSX syntax, a React component, multiple React components, or an entire application. In a plain React application, you would only use it once to bootstrap the component tree.

`ReactDOM.render()` expects two arguments. The first argument is for rendering the JSX. The second argument specifies the place where the React application hooks into your HTML. It expects an element with an `id='root'`, found in the *public/index.html* file.

Code Playground

```
ReactDOM.render(
  <h1>Hello React World</h1>,
  document.getElementById('root')
);
```

During implementation, `ReactDOM.render()` takes your App component, though it can also pass simple JSX. It doesn't require an component instance.

Exercises:

- Open the *public/index.html* to see where the React application hooks into your HTML
- Read about rendering elements in React[58]

[58]https://reactjs.org/docs/rendering-elements.html

Hot Module Replacement

Hot Module Replacement can be used in the *src/index.js* file to improve your experience as a developer. By default, *create-react-app* will cause the browser to refresh the page whenever its source code is modified. Try it by changing the `helloWorld` variable in your *src/App.js* file, which should cause the browser to refresh the page. There is a better way of handling source code changes during development, however.

Hot Module Replacement (HMR) is a tool for reloading your application in the browser without the page refresh. You can activate it in *create-react-app* by adding the following configuration to your *src/index.js* file:

src/index.js

```
import React from 'react';
import ReactDOM from 'react-dom';
import './index.css';
import App from './App';

ReactDOM.render(
  <App />,
  document.getElementById('root')
);

if (module.hot) {
  module.hot.accept();
}
```

Again, change the `helloWorld` variable in your *src/App.js* file. The browser shouldn't refresh, but the application will reload and show the correct output. HMR comes with multiple advantages:

Imagine you are debugging your code with `console.log()` statements. These statements will stay in your developer console, even though you changed your code, because the browser doesn't refresh the page anymore. In a growing application, page refreshes delay productivity; HMR removes this obstacle by eliminating the incremental time loss it takes for a browser to reload.

The most useful benefit of HMR is that you can keep the application state after the application reloads. For instance, assume you have a dialog or wizard in your application with multiple steps, and you are on step 3. Without HMR, you make changes to the source code and your browser refreshes the page. You would then have to open the dialog again and navigate from step 1 to step 3 each time. With HMR your dialog stays open at step 3, so you can debug from the exact point you're working on. With the time saved from page loads, this makes HMR an invaluable tool for React developers.

Exercises:

- Change your *src/App.js* source code a few times to see HMR in action
- Watch the first 10 minutes of Live React: Hot Reloading with Time Travel[59] by Dan Abramov

[59]https://www.youtube.com/watch?v=xsSnOQynTHs

Complex JavaScript in JSX

So far, you have rendered a few primitive variables in your JSX. Now, we will render a list of items. The list will contain sample data in the beginning, but later we will learn how to fetch the data from an external API.

First you have to define the list of items:

src/App.js

```
import React, { Component } from 'react';
import './App.css';

const list = [
  {
    title: 'React',
    url: 'https://reactjs.org/',
    author: 'Jordan Walke',
    num_comments: 3,
    points: 4,
    objectID: 0,
  },
  {
    title: 'Redux',
    url: 'https://redux.js.org/',
    author: 'Dan Abramov, Andrew Clark',
    num_comments: 2,
    points: 5,
    objectID: 1,
  },
];

class App extends Component {
  ...
}
```

The sample data represents information we will fetch from an API later on. Items in this list each have a title, a url, and an author, as well an identifier, points (which indicate how popular an article is), and a count of comments.

Now you can use the built-in JavaScript map functionality[60] in JSX, which iterates over a list of items to display them according to specific attributes. Again, we use curly braces to encapsulate the JavaScript expression in JSX:

[60]https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

src/App.js

```
class App extends Component {
  render() {
    return (
      <div className="App">
        {list.map(function(item) {
          return <div>{item.title}</div>;
        })}
      </div>
    );
  }
}

export default App;
```

Using JavaScript alongside HTML in JSX is very powerful. For a different task you may have used `map` to convert one list of items to another. This time, we used `map` to convert a list of items to HTML elements.

Code Playground

```
const array = [1, 4, 9, 16];

// pass a function to map
const newArray = array.map(function (x) { return x * 2; });

console.log(newArray);
// expected output: Array [2, 8, 18, 32]
```

So far, only the `title` is displayed for each item. Let's experiment with more of the item's properties:

src/App.js

```
class App extends Component {
  render() {
    return (
      <div className="App">
        {list.map(function(item) {
          return (
            <div>
              <span>
                <a href={item.url}>{item.title}</a>
              </span>
              <span>{item.author}</span>
              <span>{item.num_comments}</span>
              <span>{item.points}</span>
            </div>
          );
        })}
      </div>
    );
  }
}

export default App;
```

Note how the `map` function is inlined in your JSX. Each item property is displayed with a `<span>` tag, and the url property of the item is in the `href` attribute of the anchor tag.

React will display each item, but you can still do more to help React embrace its full potential. By assigning a key attribute to each list element, React can identify modified items when the list changes. These sample list items come with an identifier:

src/App.js

```
{list.map(function(item) {
  return (
    <div key={item.objectID}>
      <span>
        <a href={item.url}>{item.title}</a>
      </span>
      <span>{item.author}</span>
      <span>{item.num_comments}</span>
      <span>{item.points}</span>
    </div>
  );
})}
```

Make sure that the key attribute is a stable identifier. Avoid using the index of the item in the array, because the array index is not stable. If the list changes its order, for example, React will not be able to identify the items properly.

src/App.js

```
// don't do this
{list.map(function(item, key) {
  return (
    <div key={key}>
      ...
    </div>
  );
})}
```

Start your app in a browser, and you should see both items of the list displayed.

Exercises:

- Read about React lists and keys[61]
- Recap the standard built-in array functionalities in JavaScript[62]
- Use more JavaScript expressions on your own in JSX

[61] https://reactjs.org/docs/lists-and-keys.html
[62] https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/

ES6 Arrow Functions

JavaScript ES6 introduced arrow functions expressions, which are shorter than a function expressions.

Code Playground

```
// function declaration
function () { ... }

// arrow function declaration
() => { ... }
```

You can remove the parentheses in an arrow function expression if it only has one argument, but you have to keep the parentheses if it gets multiple arguments:

Code Playground

```
// allowed
item => { ... }

// allowed
(item) => { ... }

// not allowed
item, key => { ... }

// allowed
(item, key) => { ... }
```

You can also write `map` functions more concisely with an ES6 arrow function:

src/App.js

```
{list.map(item => {
  return (
    <div key={item.objectID}>
      <span>
        <a href={item.url}>{item.title}</a>
      </span>
      <span>{item.author}</span>
      <span>{item.num_comments}</span>
      <span>{item.points}</span>
    </div>
  );
})}
```

You can remove the *block body*, the curly braces, with the ES6 arrow function. In a *concise body*, an implicit return is attached; thus, you can remove the `return` statement. This will happen often in this book, so be sure to understand the difference between a block body and a concise body when using arrow functions.

src/App.js

```
{list.map(item =>
  <div key={item.objectID}>
    <span>
      <a href={item.url}>{item.title}</a>
    </span>
    <span>{item.author}</span>
    <span>{item.num_comments}</span>
    <span>{item.points}</span>
  </div>
)}
```

Your JSX should look more concise and readable now, as it omits the `function` statement, the curly braces, and the return statement.

Exercises:

- Read about ES6 arrow functions[63]

[63]https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Functions/Arrow_functions

ES6 Classes

JavaScript ES6 introduced classes, which are commonly used in object-oriented programming languages. JavaScript, always flexible in its programming paradigms, allows functional programming and object-oriented programming to work side-by-side.

While React embraces functional programming, e.g. immutable data structures and function compositions, classes are used to declare ES6 class components. React mixes the good parts of both programming paradigms.

Consider the following Developer class to examine a JavaScript ES6 class without a component.

Code Playground

```
class Developer {
  constructor(firstname, lastname) {
    this.firstname = firstname;
    this.lastname = lastname;
  }

  getName() {
    return this.firstname + ' ' + this.lastname;
  }
}
```

A class has a constructor to make it instantiable. The constructor takes arguments and assigns them to the class instance. A class can also define functions. Because the function is associated with a class, it is called a method, or a class method.

The Developer class is only the class declaration we use here, as you can create multiple instances of a class by invoking it. It is similar to the ES6 class component, which has a declaration, but you have to use it somewhere else to instantiate it:

Code Playground

```
const robin = new Developer('Robin', 'Wieruch');
console.log(robin.getName());
// output: Robin Wieruch
```

React uses JavaScript ES6 classes for ES6 class components, which you have already used at least once so far:

src/App.js

```
import React, { Component } from 'react';

...

class App extends Component {
  render() {
    ...
  }
}
```

When you declare the App component it extends from another component. In object-oriented programming, the term "extends" refers to the principle of inheritance, which means that functionality can be passed from one class to another. The App class extends from the Component class, meaning it inherits functionality from the Component class. The Component class is used to extend a basic ES6 class to a ES6 component class. It has all the functionalities that a component in React needs. The render method is one function you have already used. You will learn about other component class methods as we move along.

The `Component` class encapsulates all the implementation details of a React component, which allows developers to use classes as components in React.

Methods exposed by a React `Component` are its public interface. One of these methods must be overridden, while the others don't need to be overridden. You will learn about these when we discuss lifecycle methods later. The `render()` method has to be overridden, because it defines the output of a React `Component`, so it must be defined. These are the basics of JavaScript ES6 classes, and how they are used in React to extend them to components.

Exercises:

- Read about JavaScript fundamentals before learning React[64]
- Read about ES6 classes[65]

[64]https://www.robinwieruch.de/javascript-fundamentals-react-requirements/
[65]https://developer.mozilla.org/en/docs/Web/JavaScript/Reference/Classes

Congratulations, you have learned to bootstrap your first React application! Let's recap:

- **React**
  - Create-react-app bootstraps a React application
  - JSX mixes up HTML and JavaScript to define the output of React components in their render methods
  - Components, instances, and elements are different items in React
  - `ReactDOM.render()` is an entry point for a React application to hook React into the DOM
  - Built-in JavaScript functionalities can be used in JSX
  - Map can be used to render a list of items as HTML elements
- **ES6**
  - Variable declarations with `const` and `let` can be used for specific use cases
  - Use const over let in React applications
  - Arrow functions can be used to keep your functions concise
  - Classes are used to define components in React by extending them

Now that you've completed the first chapter, it's advisable to experiment with the source code you have written so far and see what changes you can make on your own. You can find the source code in the official repository[66].

[66]https://github.com/the-road-to-learn-react/hackernews-client/tree/5.1