



Creating a React App

Use `create-react-app` to bootstrap a React application on your own computer

Creating a React App

Introduction

React is a user interface framework developed by Facebook. It has a quickly growing developer adoption rate and was ranked as the most loved language or technology in the [2019 Stackoverflow developer survey](#). This article will walk you through setting up your first React app and assumes you are familiar with text editors and command line navigation. We will be using the *Node package manager (npm)*, so you will need to have [Node installed](#).

1. Set up the boilerplate application

It is possible to manually create a React app, but Facebook has created a node module *create-react-app* to generate a boilerplate version of a React application.

Besides providing something that works out-of-the-box, this has the added benefit of providing a consistent structure for React apps that you will recognize as you move

[Next](#)[Get Help](#)

We will use npm to install the create-react-app command line interface (CLI) globally (`-g`).

Open up your terminal and run `npm install -g create-react-app` :

```
✓ ~/Documents/Projects
[22:34 $ npm install -g create-react-app
/usr/local/bin/create-react-app -> /usr/local/lib/node_modules/create-react-app/index.js
+ create-react-app@1.3.1
added 72 packages in 2.01s
```

Now that you have the CLI available for use, navigate to the parent directory that you would like to place the application within. Then, run `create-react-app` with the name you would like to use for your project (just no capital letters :-)).

```
create-react-app <name-of-app>
```

Upon completion, you will get some quick tips on how to use the application:

```
npm start
  Starts the development server.

npm run build
  Bundles the app into static files for production.

npm test
  Starts the test runner.

npm run eject
  Removes this tool and copies build dependencies, configuration files
  and scripts into the app directory. If you do this, you can't go back!
```

We suggest that you begin by typing:

```
cd myfirstreactapp
npm start
```

Happy hacking!

Before we run the app, lets take a look around the app structure and see what it contains.

2. Project structure

Next

Get Help

Change directories into the app you just created. If you list the contents of the directory including hidden files (`ls -la`), you should see the following structure:

```
.gitignore
README.md
node_modules
package-lock.json
package.json
public
src
```

`create-react-app` has taken care of setting up the main structure of the application as well as a couple of developer settings. Most of what you see will not be visible to the visitor of your web app. React uses a tool called `webpack` which transforms the directories and files here into static assets. Visitors to your site are served those static assets.

.gitignore

This is the standard file used by the source control tool `git` to determine which files and directories to ignore when committing code. While this file exists, `create-react-app` did not create a `git` repo within this folder. If you take a look at the file, it has taken care of ignoring a number of items (even **.DS_Store** for Mac users):

✓ ~/Documents/Projects/myfirstreactapp

[23:20 \$ cat .gitignore

See <https://help.github.com/ignore-files/> for more about ignoring files.

dependencies
/node_modules

testing
/coverage

production
/build

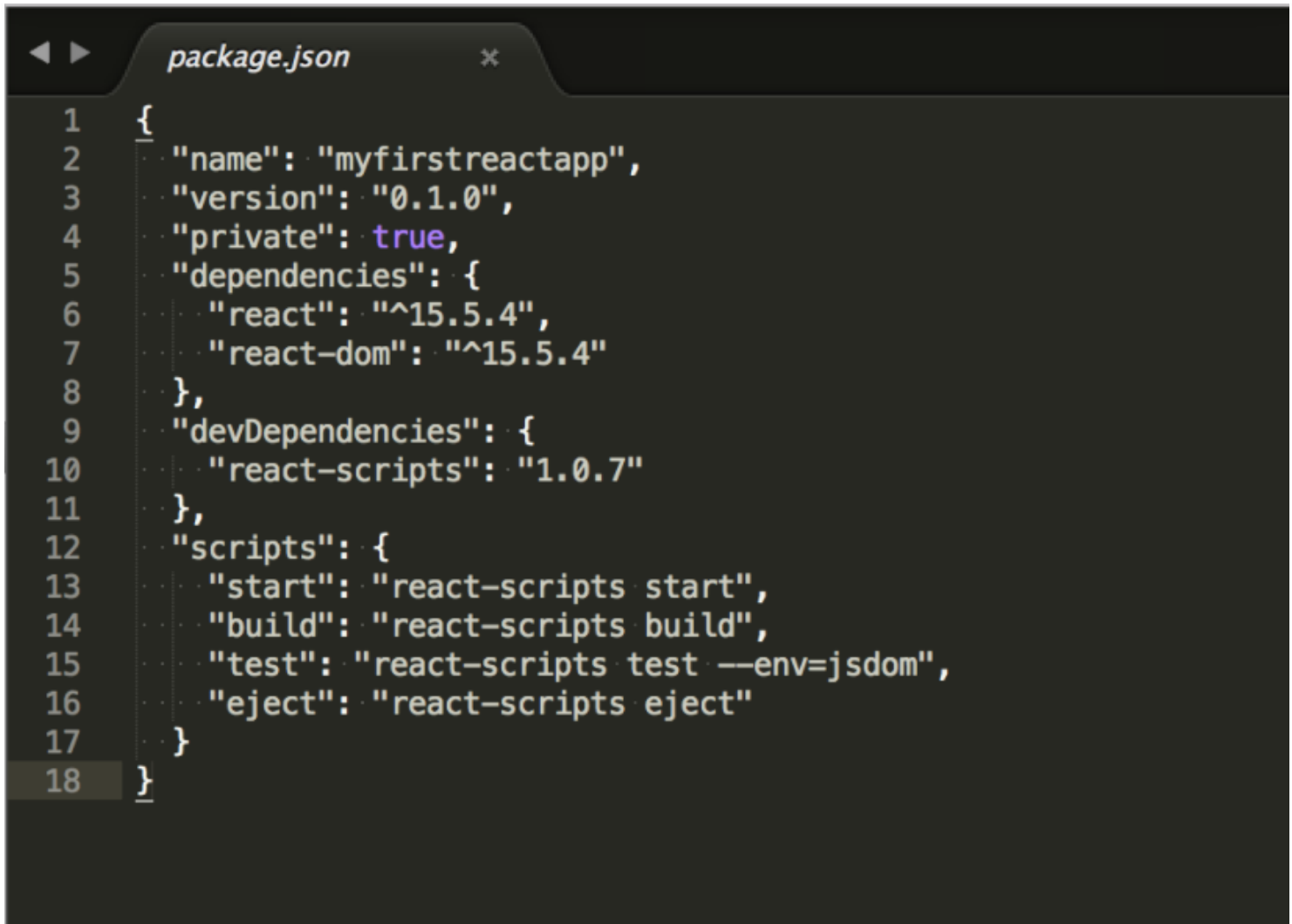
misc
.DS_Store
.env.local
.env.development.local
.env.test.local
.env.production.local

npm-debug.log*
yarn-debug.log*
yarn-error.log*

package.json

Next

Get Help



```
1 {
2   "name": "myfirstreactapp",
3   "version": "0.1.0",
4   "private": true,
5   "dependencies": {
6     "react": "^15.5.4",
7     "react-dom": "^15.5.4"
8   },
9   "devDependencies": {
10    "react-scripts": "1.0.7"
11  },
12  "scripts": {
13    "start": "react-scripts start",
14    "build": "react-scripts build",
15    "test": "react-scripts test --env=jsdom",
16    "eject": "react-scripts eject"
17  }
18 }
```

This file outlines all the settings for the React app.

- `name` is the name of your app
- `version` is the current version
- `"private": true` is a failsafe setting to avoid accidentally publishing your app as a public package within the npm ecosystem.
- `dependencies` contains all the required node modules and versions required for the application. Here, it contains two dependencies, which allow us to use `react` and `react-dom` in our JavaScript. In the screenshot above, the `react` version specified is `^15.5.4`. This means that npm will install the most recent major version matching `15.x.x`. In contrast, you may also see something like `~1.2.3` in **package.json**, which

[Next](#)[Get Help](#)

- `devDependencies` contains useful node modules and versions for using the React app in a development environment. Here, it contains one dependency, `react-scripts`, which provides a set of useful development scripts for working with React.
- `scripts` specifies aliases that you can use to access some of the `react-scripts` commands in a more efficient manner. For example running `npm test` in your command line will run `react-scripts test --env=jsdom` behind the scenes.

node_modules

This directory contains dependencies and sub-dependencies of packages used by the current React app, as specified by **package.json**. If you take a look, you may be surprised by how many there are.

Running `ls -1 | wc -1` within the **node_modules/** directory will yield more than 800 subfolders. This folder is automatically added to the **.gitignore** for good reason! Don't worry, even with all these dependencies, the basic app will only be around 50 KB after being minified and compressed for production.

package-lock.json

This file contains the exact dependency tree installed in **node_modules/**. This provides a way for teams working on private apps to ensure that they have the same version of dependencies and sub-dependencies. It also contains a history of changes to `package.json`, so you can quickly look back at dependency changes.

public

This directory contains assets that will be served directly without additional processing by webpack. **index.html** provides the entry point for the web app. You will also see a favicon (header icon) and a **manifest.json**.

[Next](#)[Get Help](#)

from the Android UI). You can read more about it [here](#).

src

This contains the JavaScript that will be processed by webpack and is the heart of the React app. Browsing this folder, you see the main App JavaScript component (**App.js**), its associated styles (**App.css**), and test suite (**App.test.js**). **index.js** and its styles (**index.css**) provide an entry into the App and also kicks off the **registerServiceWorker.js**. This service worker takes care of caching and updating files for the end-user. It allows for offline capability and faster page loads after the initial visit. More on this methodology is available in the [Create React App readme](#).

As your React app grows, it is common to add a **components/** directory to organize components and component-related files and a **views** directory to organize React views and view-related files.

3. Starting the React app development server

As was stated in the success message when you ran `create-react-app`, you just need to run `npm start` in your app directory to begin serving the development server. It should auto-open a tab in your browser that points to `http://localhost:3000/` (if not, manually visit that address). You will be greeted with the React welcome banner:



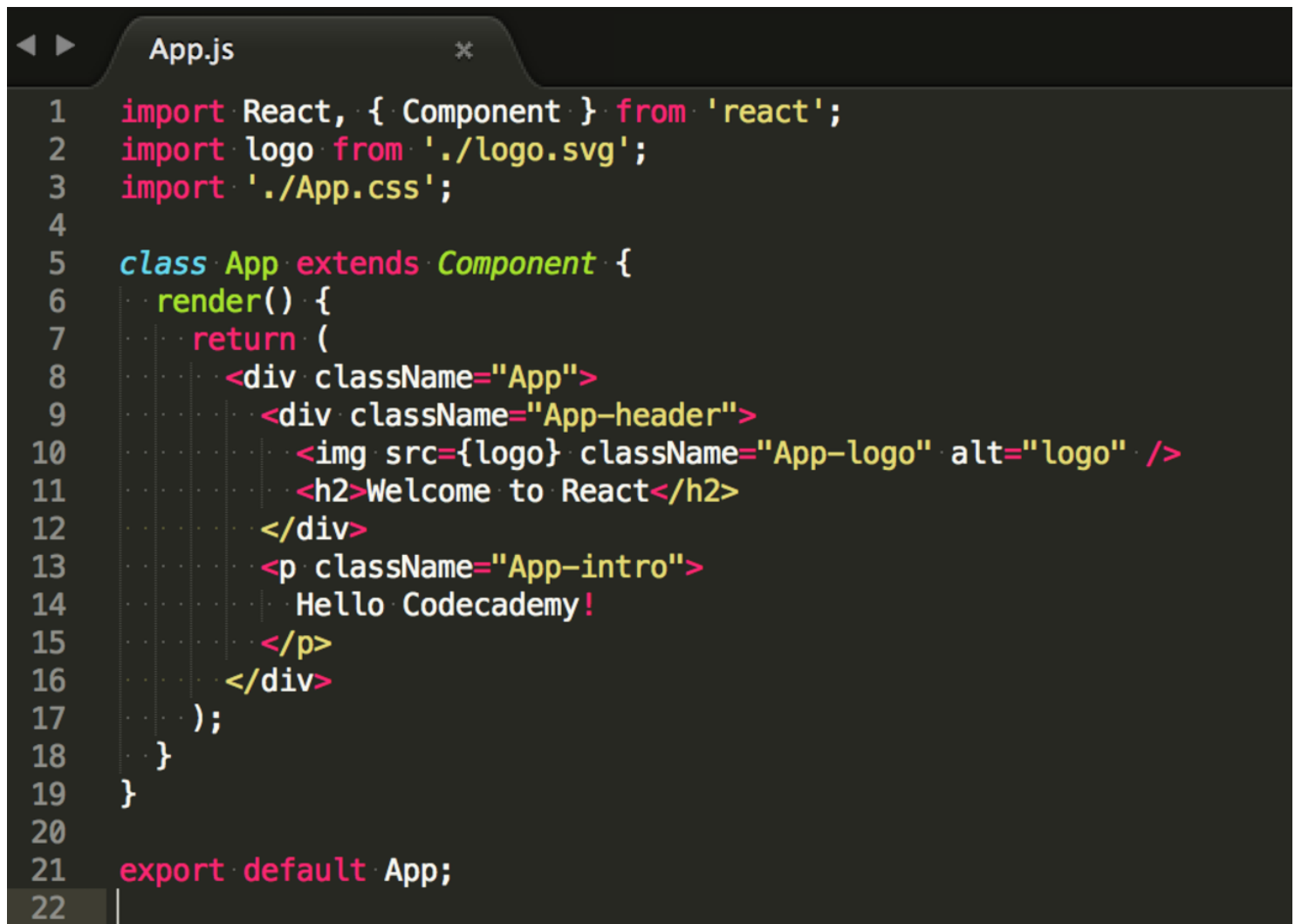
Welcome to React

[Next](#)[Get Help](#)

As stated, any changes to the source code will live-update here. Let's see that in action.

Leave the current terminal tab running (it's busy serving the React app) and open **src/App.js** in your favorite text editor. You'll see what looks like a mashup of JavaScript and HTML. This is JSX, which is how React adds XML syntax to JavaScript. It provides an intuitive way to build React components and is compiled to JavaScript at runtime. We'll delve deeper into this in other content, but for now, let's make a simple edit and see the update in the browser.

Change the main paragraph text to read: `Hello Codecademy!` in **App.js** and save the file.



```
1 import React, { Component } from 'react';
2 import logo from './logo.svg';
3 import './App.css';
4
5 class App extends Component {
6   render() {
7     return (
8       <div className="App">
9         <div className="App-header">
10           <img src={logo} className="App-logo" alt="logo" />
11           <h2>Welcome to React</h2>
12         </div>
13         <p className="App-intro">
14           Hello Codecademy!
15         </p>
16       </div>
17     );
18   }
19 }
20
21 export default App;
22
```

If you left the terminal running, you should be able to switch over to your browser and

[Next](#)[Get Help](#)



Welcome to React

Hello Codecademy!

Congratulations! You're now up and running with React. You can clean up any unnecessary files and begin adding functionality for your application.

[Next](#)[Get Help](#)