1. Set-up React project with webpack-babel

<https://www.valentinog.com/blog/react-webpack-babel/>

1. Set-up react-redux with react app

<https://react-redux.js.org/introduction/quick-start>

<https://medium.com/@rajaraodv/webpack-the-confusing-parts-58712f8fcad9>

<https://developerhandbook.com/webpack/configure-webpack-4-hmr-scratch/>

<https://webpack.js.org/guides/hot-module-replacement/#via-the-nodejs-api>

<https://medium.com/codingthesmartway-com-blog/create-a-rest-api-with-json-server-36da8680136d>

**How to set up React, webpack, and Babel: setting up the project**

1. Start off by creating a directory for the project:
2. Open command prompt and run below command.

*mkdir webpack-react-tutorial*

1. Inizialize the project by running:

npm init -y

and you’re good to go.

## How to set up React, webpack, and Babel: setting up webpack

*Out of the box, webpack won't require you to use a configuration file. However, it will assume the entry point of your project is src/index and will output the result in dist/main.js minified and optimized for production.*

*Usually your projects will need to extend this functionality, for this you can create a webpack.config.js file in the root folder and webpack will automatically use it.*

*All the available configuration options are specified below.*

1. **webpack** it’s an incredibly powerful tool. While you can get by without touching a single line of configuration there will be a time for some custom setup of sort. Sooner or later you may want to learn webpack. Why not starting now? [Learning webpack is valuable](https://www.valentinog.com/blog/webpack-tutorial/) not only for working with React but for configuring every frontend project. webpack ingests raw React components for producing JavaScript code that (almost) every browser can understand.
2. Let’s install it by running:

*npm i webpack --save-dev*

1. You will also need webpack-cli. Pull it in with:

*npm i webpack-cli --save-dev*

1. Next up add the webpackcommand inside package.json
2. **"scripts"**: {
3. **"build"**: "webpack --mode production"
4. }

5. At this point there is no need to define a configuration file for webpack. Older webpack’s version did automatically look for a configuration file. Since version 4 that is no longer the case: you can start developing straigh away. In the next section we’ll install and configure Babel for transpiling our code.

## How to set up React, webpack, and Babel: setting up Babel

1. **React components are mostly written in JavaScript ES6**. ES6 is a nice improvement over the language but older browsers cannot understand the new syntax. Take the **class keyword** for example. Stateful React components are declared as classes (I guess it will be no longer the case sooner or later). So for getting ES6 to work in older browser we need some kind of transformation. And that transformation is called **transpiling**. Webpack doesn’t know how to transform ES6 JavaScript to ES5 but it has this concept of **loaders**: think of them as of transformers. A webpack loader takes something as the input and produces something else as the output. Babel-loader is the Webpack loader responsible for taking in the ES6 code and making it understandable by the browser of choice. Obsviusly babel-loader makes use of Babel. And Babel must be configured to use a bunch of presets
2. **babel preset env** for compiling Javascript ES6 code down to ES5 (please note that babel-preset-es2015 is now deprecated)
3. **babel preset react** for compiling JSX and other stuff down to Javascript. Let’s pull in the dependencies with below command

*npm i @babel/core babel-loader @babel/preset-env @babel/preset-react --save-dev*

1. Don’t forget to configure Babel! Create a new file named .babelrc inside the project folder.
2. {
3. **"presets"**: ["@babel/preset-env", "@babel/preset-react"]
4. }
5. At this point we’re ready to define a minimal webpack configuration.
6. Create a file named webpack.config.js and fill it like the following:
7. module.exports = {
8. entry: {
9. main: './index.js'
10. },
11. output: {
12. path: \_\_dirname + '/dist',
13. filename: 'app.js'
14. },
15. module: {
16. rules: [
17. {
18. test: /\.(js|jsx)$/,
19. exclude: /node\_modules/,
20. use: {
21. loader: "babel-loader"
22. }
23. }
24. ]
25. }
26. }.button('reset')
27. });

## How to set up React, webpack, and Babel: writing React components

1. Before touching any code let’s pull in React by running:

npm i react react-dom

A question I get a lot is “should I install react and react-dom as dev dependencies or not?” It doesn’t matter for the final result. webpack will still produce a bundle with your JavaScript application. I have this bad habit of putting react and react-dom in devDependencies so you’ll find that style a lot in my tutorials.

1. Now we can start with code. Create an app.js file inside ”./src/views’ with below code.
2. import React from 'react';
3. const App = () => (
4. <div>Hello App</div>
5. );
6. export default App;

And now it’s time to wire things up! webpack expects the entry point to be ./index.js. Create the file and place an import directive into it for requiring the container component:

import React, {Component} from 'react';

import ReactDOM from 'react-dom';

import App from './src/views/app';

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

With this in place we’re ready to create our bundle by running:

npm run build

Give Webpack a second and see the bundle come to life! The bundle will be placed into ./dist/main.js.

## How to set up React, webpack, and Babel: webpack dev server

* 1. You don’t want to type npm run build every time you change a file. It takes only 3 lines of configuration to have a **development server** up and running. Once configured webpack will launch your application inside a browser. Also, every time you save a file after a modification webpack wev server will automagically refresh the browser’s window.
  2. To set up webpack dev server install the package with:

npm i webpack-dev-server --save-dev

* 1. Open up package.json to add the start script:

1. **"scripts"**: {
2. **"start"**: "webpack-dev-server --open --mode development",
3. **"build"**: "webpack --mode production"
4. }

4. Now, by running: npm start you should see webpack launching your application inside the browser.

## How to set up webpack dev server using different config files