**How to Set up Loacl Development Environment for React with Webpack 4 and Babel 7**

It is universally acknowledged that setting up a local dev environment is hard work. Especially for javascript frameworks that get constantly updated. The documentation for the older version suddenly stops working and you have to make sure you are installing the correct versions.

In this post, I will show you how to set up local dev environment for [React](https://reactjs.org/) with [webpack](https://webpack.js.org/concepts/) 4 and [babel](https://babeljs.io/) 7. Webpack allows you to create a local server. It also enables you to do hot loading, which means the website you are building gets updated immediately after you change the code without restarting the server. We can also easily add browser support for ES6 and [JSX](https://reactjs.org/docs/introducing-jsx.html). This is pretty much my favourite local dev set up to make a React Website.

First of all, let’s clarify the version of the packages I am using here. If you are using different versions, the chance is that you need to do a few tweaks.

**Dev Dependencies**

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

**Dependencies**

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

As prerequisite, you need to have Node.js and npm installed, of course.

You can find the whole solution in the our GitHub repo [here](https://github.com/mydatahack/react-webpack-dev-demo).

Ok, let’s get started.

1. **Project set up**

Here is the project structure looks like. For CSS, you can drop any css file to the css folder. We are using Bootstrap here.

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

1. **Package Installation**

Let’s install the packages. @babel/preset-env converts ES6 to ES5 so that most browser can render your React code. [Polyfill](https://babeljs.io/docs/en/babel-polyfill) gives you greater support for browser that doesn’t support ES5.

For React, you need both react and react-dom to render React elements.

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

[/cc]

1. **Configure webpack-dev-server**

First of all, you need to create webpack.config.js file. And add the code below. Mode is required. Add 'development'. You need two entries, one for polyfill and the other for the main javascript file. Output requires absolute path for the project folder. We set the filename for the compiled Javascript as index.js. We also need to set the rules so as to include all javascript files and exclude node modules.

<pre class="theme:vs2012-black font:verdana font-size:14px toolbar:2 lang:js decode:true ">

</pre>

1. **Configure package.json**

We need to add two presets for babel. Then add script so that the dev server can start with npm start.

<pre class="theme:vs2012-black font:verdana font-size:14px toolbar:2 lang:js decode:true ">

</pre>

1. **Create a simple HTML page**

This is just a simple html example that uses Bootstrap 4, including a target div for React Element.

[cc lang="java" tab\_size="4" lines="-1"]

[/cc]

1. **Import React and create React element**

In client.js, we can now write any Javascript code. In this example, I created a simple React button element. The key is to import react and react-dom modules. Nothing fancy, but this should render fine.

<pre class="theme:vs2012-black font:verdana font-size:14px toolbar:2 lang:js decode:true ">

</pre>

1. **Start the server and check it!**

You can start the server with npm start. Go to the URL <http://localhost:3000>.

Now, we have a local React dev environment!

If you want to do more reading, [this one](https://www.valentinog.com/blog/react-webpack-babel/) is a pretty good tutorial.