How to Webpack 5 with Babel - Setup Tutorial

OCTOBER 30, 2020 BY ROBIN WIERUCH - EDIT THIS POST

Follow on Twitter 17k Follow on Facebook

This tutorial is part 3 of 3 in the 'Frontend Setup'-series.

Part 1: How to set up a modern JavaScript project
Part 2: How to set up Webpack 5

Babel enables one writing code with JavaScript features that aren't supported by most browser yet. Perhaps you have heard about JavaScript ES6 (ES2015), ES7, and other versions of ECMAScript specification which are up and coming for the JavaScript language. At the time of reading this, various versions may be already included in the JavaScript language.

By using Babel, the code which isn't supported yet, will get transpiled back to vanilla JavaScript so that every environment (e.g. browser) can interpret it. In order to get Babel

running, you need to install two of its main dependencies on the command line:

```
npm install --save-dev @babel/core @babel/preset-env
```

Moreover, in case you have Webpack in place to bundle your JavaScript application, you will have to install a Webpack Loader for Babel:

```
npm install --save-dev babel-loader
```

Now, with all libraries (node packages) in place, you need to adjust your *package.json* and *webpack.config.js* (if necessary) to respect the Babel changes. These changes will include all packages you have installed before. First, in your *package.json*, include the Babel Preset:

```
{
  ...
  "babel": {
    "presets": [
      "@babel/preset-env"
    ]
  },
  ...
}
```

"[The] @babel/preset-env [preset] is a smart preset that allows you to use the latest JavaScript without needing to micromanage which syntax transforms (and optionally, browser polyfills) are needed by your target environment(s). This both makes your life easier and JavaScript bundles smaller!" (Source)

Second, your *webpack.config.js* file needs to include Babel in its build process as well. There, make use of the previously installed Loader for Babel. You need to tell Webpack on which files to use the loader (e.g. *.js* files) and optionally which folders to exclude from the process (e.g. *node_modules*):

```
const path = require('path');

module.exports = {
  entry: path.resolve(__dirname, './src/index.js'),
  module: {
    rules: [
      {
        test: /\.(js)$/,
        exclude: /node_modules/,
```

```
        use: ['babel-loader']
      }

    ]
  },
  resolve: {
    extensions: ['*', '.js']
  },
  output: {
    path: path.resolve(__dirname, './dist'),
    filename: 'bundle.js',
  },
  devServer: {
    contentBase: path.resolve(__dirname, './dist'),
  },
};
```

You can start your application again. Nothing should have changed except for that you can use upcoming ECMAScript features for JavaScript now. An optional step would be to extract your Babel configuration into a separate *.babelrc* configuration file. You can create this file in your project's root directory on the command line:

```
touch .babelrc
```

Then, add the configuration for Babel -- which you have previously added in your *package.json* -- in the *.babelrc* file. Don't forget to remove the configuration in the *package.json* afterward. It should be configured only once.

```
{
  "presets": [
    "@babel/preset-env"
  ]
}
```

Babel enables you to use future JavaScript features in your browser, because it transpiles it down to vanilla JavaScript. Try it yourself by installing your first plugin. Make sure to see that the JavaScript feature doesn't work at first in your *src/index.js* file, but once you installed the plugin for the feature and integrated it in your *.babelrc* file, it should be possible to run the JavaScript source code.

Exercises:

- Confirm your source code for the last section
- Try out a Babel Plugin yourself

- Try out a Babel Plugin yourself

  - Install a Babel Plugin via npm to your project to support an upcoming JavaScript feature
  - Add the Plugin to your *.babelrc* file
  - Try the new JavaScript feature in your *src/index.js* file
- Try Imports
  - Create another JavaScript file in your *src/* folder
  - Import the new JavaScript file in your *src/index.js* file
  - Add a logging statement to your new JavaScript file and check whether it shows up in the browser

This tutorial is part 1 of 2 in 'React Setup'-series.

Part 2: How to set up React with Webpack and Babel

This tutorial is part 1 of 2 in 'Webpack with ESLint'-series.

Part 2: How to use ESLint in Webpack

This tutorial is part 1 of 4 in 'Webpack with Style'-series.

Part 2: How to use CSS with Webpack
Part 3: How to use Webpack with SASS
Part 4: How to use Webpack with PostCSS

This tutorial is part 1 of 3 in 'Webpack with Font'-series.

Part 2: How to use CSS with Webpack
Part 3: How to use Webpack with Fonts

This tutorial is part 1 of 2 in 'Webpack with Images'-series.

Part 2: How to use Images with Webpack

This tutorial is part 2 of 3 in 'Webpack Advanced Setup'-series.

Part 1: How to set up Webpack 5

Part 3: How to set up an advanced Webpack application

Show Comments

KEEP READING ABOUT ESLINT›

HOW TO USE PRETTIER IN VS CODE

A brief step by step tutorial on how to install and use Prettier in VS Code. Prettier is an opinionated code formatter which ensures one unified code format. It can be used within VS Code by...

HOW TO USE ESLINT IN WEBPACK 5 - SETUP TUTORIAL

So far, you should have a working JavaScript with Webpack application. In this tutorial, we will take this one step further by introducing ESLint for an enforced unified code style without code smells...

THE ROAD TO REACT

Learn React by building real world applications. No setup configuration. No tooling. Plain React in 200+ pages of learning material. Learn React like **50.000+ readers**.

GET THE BOOK >

Get it on Amazon.

TAKE PART

NEVER MISS AN ARTICLE ABOUT WEB DEVELOPMENT AND JAVASCRIPT.

- ✔ Join 50.000+ Developers
- ✔ Learn Web Development with JavaScript
- ✔ Tips and Tricks
- ✔ Access Tutorials, eBooks and Courses
- ✔ Personal Development as a Software Engineer

Your email address SUBSCRIBE

View our Privacy Policy.

PORTFOLIO

Online Courses

Open Source

Tutorials

ABOUT

About me

What I use

How to work with me

How to support me

© Robin Wieruch

Contact Me Privacy & Terms