

Write JavaScript for the Web

10 hours Medium

License

Last updated on 4/2/20



Configure Babel

There is no video for this chapter. 😊 Enjoy the text.

Throughout this course, we have been using a lot of "next generation" JavaScript: some of it is ES6 (like `let`, `const`, and arrow functions) and is fairly well supported since mid-2017, but some of it is even more futuristic and has poorer browser support (such as `async` functions). These features make our lives easier as developers, but we do need to make sure the end-user can use our awesome modern apps.

In walks Babel.

To put it simply, Babel converts next generation JavaScript into more browser-compatible JavaScript. Let's start by installing the CLI to your project.

From your project directory, run:

```
npm install babel-cli babel-preset-env --save-dev
```

This installs Babel as a development dependency for our project. Now we need to configure it to do exactly what we need.

Create a file in your project directory called `.babelrc`:

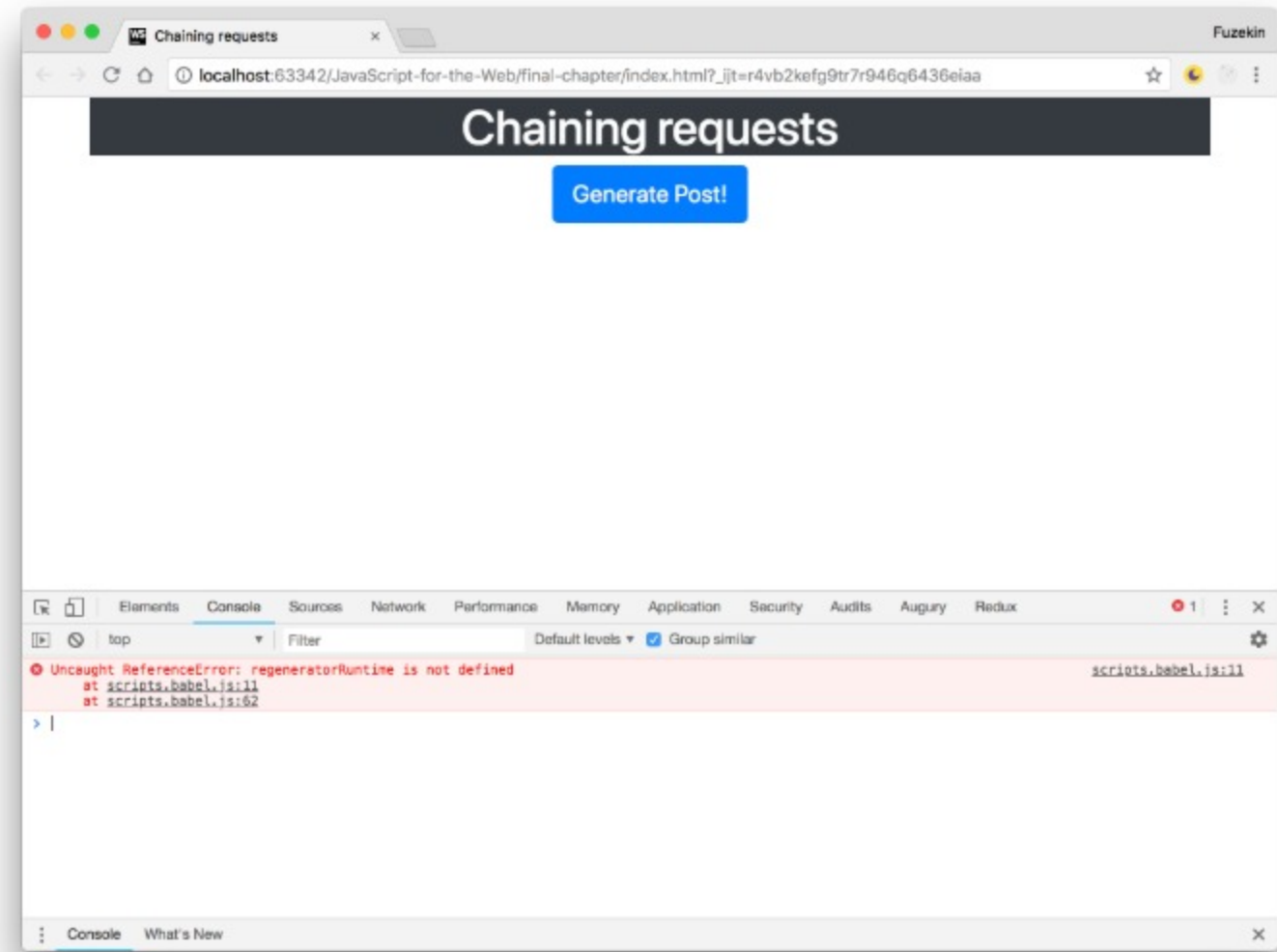
```
{
  "presets": ["env"]
}
```

That's the only configuration Babel needs! Let's ask Babel to transpile `scripts.js` and save the output to `scripts.babel.js`:

```
npx babel scripts.js --out-file scripts.babel.js
```

The `npx` command allows us to run locally installed packages!

Now if you open `scripts.babel.js`, you will see that our code has been transformed! Try importing our new `js` file into `index.html` instead of `scripts.js` and see if it works.



Uh-oh, it seems something went wrong with our Babel transpilation!

It turns out it has to do with our use of `async` functions, but not to worry, there is a simple solution: the `babel-polyfill`.

For more information about the `babel-polyfill`, check out the [official documentation](#).

All we need to do is add the browser version of the `babel-polyfill` to our `index.html` to allow our `async` functions to work properly. Add the following line to `index.html`:

```
<script src="./node_modules/babel-polyfill/browser.js"></script>
```

Now if you retest our app, everything should be functioning properly!

Summary

In this chapter, we covered:

- how to install Babel and its dependencies to our project
- how to configure Babel with `.babelrc`
- how to use the `babel-polyfill` to successfully use `async` functions

Using Babel has allowed us to produce far better backwards compatibility for our next generation JavaScript code. In the next and final chapter of this course, we are going to take a look at how we can integrate Babel into a full project build using Gulp.

MARK THIS CHAPTER AS UNFINISHED


MANAGE YOUR DEPENDENCIES BUILD AND RUN YOUR PROJECT

Build in your local environment

- 1. Manage your dependencies
- 2. Configure Babel
- 3. Build and run your project
- 4. Get some practice building your project with Gulp
- 5. Course summary

Twitter Facebook Email

Teacher

 **Will Alexander**
Scottish developer, teacher and musician based in Paris.

