Babel Configuration

In the last exercise, we were able to see what Babel can do for our projects. In this one, we will practice getting Babel set up.

Babel is a Node package, so we would begin by initializing our npm project using npm init -y. This would set us up with a **package.json** file.

Next, we would want to install @babel/cli and @babel/preset-env as devDependences using npm install -save-dev. @babel/cli provides the main babel functionality as well as the command line interface. @babel/preset-env is a standard way of configuring Babel to transpile all of the common ES6 features, instead of having to list them one by one.

Babel is configured using a file named .babelrc. With @babel/preset-env, the content of this file is simple:

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

The **.babelrc** file is saying to use the @babel/preset-env configuration. This will transpile the latest adopted JavaScript features into code supported by older browsers.

Next, we want to define the command we will use to run Babel on our project. We will do this by defining a build command in the scripts section of our **package.json**:

"scripts" : {  
  "build" : "babel src -d out"  
}

This is using the babel command (from @babel/cli) with the first argument src specifying where the code is we want to run babel on. Then we provide the -d flag to specify where we want Babel to store the output, in this case, the **out** folder.

With the command defined, we can run Babel using the build command:

npm run build

Let’s practice setting up Babel!

Create a **package.json** by using the init command.

All the necessary packages have been installed for you.

Checkpoint 2 Passed

Hint

Run the command in the terminal that initializes a npm project. The exact syntax is above.

**2.**

Then, inside of **package.json**, add the build command in the scripts section, specifying the source and output directories.

Checkpoint 3 Passed

Hint

We define our build command under scripts in **package.json**:

"scripts": {  
  "customCommand": "commandToRun"  
},

We want build to make babel run. Check above for the exact syntax.

**3.**

Next, create the **.babelrc** file and write the configuration provided above inside of it to use the @babel/preset-env.

Click “Run” when finished.

Checkpoint 4 Passed

Hint

The file should contain the configuration in the narrative above, a "presets" key that is assigned to use the @babel/preset-env configuration.

**4.**

Now we are ready to go! Run the build command in the terminal.

Checkpoint 5 Passed

Hint

We can run our custom commands with npm run x where x is the custom command.

**5.**

Finally, open up the output in **out/Dog.js**, check out the transpiled code.

Click “Run” when you’ve had a look.

Checkpoint 6 Passed

Hint

Make sure **out/Dog.js** is open before clicking “Run” in order to pass this checkpoint.

Targeting Different Browsers

In addition to providing a fast way to configure Babel, babel-preset-env allows us to provide a list of browsers we want to be supported using a file named **.browserslistrc**.

Within this file there are many ways in which we can specify our target list of browsers. The a good default configuration, which covers most of the browsers a developer could expect users to be on can be specified with:

defaults

However, we can also customize this list. For example, we could make sure that our project is supported by the last two versions of Internet Explorer with:

last 2 Explorer versions

Or maybe you want to make sure your application supports 99.5% of the users of the internet:

cover 99.5%

The [browserslist documentation](https://github.com/browserslist/browserslist" \t "_blank) provides additional syntax and examples for conifguring **.browserslist**.

After defining **.browserslist**, you can list all browsers supported by your **.browserlist** file using the following command:

npx browserslist

Let’s practice defining a list of browsers we want our projects to support!

**Instructions**

**1.**

First, create a **.browserslistrc** file.

Then, configure it to target 99.9% of internet users. (This is much more than practically necessary.)

Hint

Create the file using the terminal or the file explorer.

You can make **.browserslistrc** target a percentage of internet users with cover x% where x is the percentage of users you want to be able to run your site.

**2.**

Check the list of browsers now supported by your application, it should be pretty long.

Hint

The exact command can be found in the narrative above. Look for the command beginning with npx.

**3.**

Run our build command to make Babel transpile our code to be compatible with the list of browsers!

Hint

Run a custom command defined in **package.json** with npm run x where x is the command.

**4.**

Open up the output in **out/Dog.js**. This code should be compatible with all the browsers listed in the previous step. Click “Run” when you’ve had a look.

Hint

Make sure you have **out/Dog.js** open before clicking “Run” to pass this checkpoint.