Using babel 7 with node

Will Willems Follow
Aug 16 · 3 min read

So you want to use the new babel 7 with node? Our goal here will be to set up a minimal node application that is able to run locally and has a build command for remote deployment. Lets get started!

Medium kinda forces me to make these... I know...

There are a couple of differences with the babel you might be familiar with and v7.

1. **Babels packages are now scoped** just like many other large organisations Babel has renamed it's NPM packages. This means `babel-cli` for example has been renamed to `@babel/cli` .
2. **No messing around with presets anymore.** You can just use `@babel/preset-env` now and optionally define your requirements in the config file.
3. `babel-node` **has been moved** from the CLI to it's own package: `@babel/node`

Ok so we have the most important things down. You can optionally read more about the changes on babels website but here is what we need to do next:

Setting up the application structure

We'll keep it as simple as possible:

First you want to run `npm init` to create a `package.json` file inside the directory and depending on if you are using Git you might want to run `git init`

We'll set up two directories, one to develop in and one to deploy our compiled assets.

```
your-project-directory
|--dist
|--src
|--package.json
```

For this example we'll add just a simple file inside the src directory called `server.js`

```
your-project-directory
|--dist
|--src
|  |--server.js
|--package.json
```

We'll need to add some babel packages to our project with `npm install --save-dev @babel/core @babel/cli @babel/preset-env @babel/node`. These respectively take care of babels general working, the usage of babel in the command line, the ability to use the newest JS features and the usage of babel with node.

For easy development we'll also add the `nodemon` package using `npm install --save-dev` which reloads node for us automatically when one of our files is changed.

Finally we just need to tell babel to use the `@babel/preset-env` package by creating a `.babelrc` file in our project root:

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

Your project structure should now look like this:

```
your-project-directory
|--dist
|--node_modules
|--src
|  |--server.js
|--package.json
|--.babelrc
```

Adding scripts to package.json

Now for the final step we'll add our commands to the `package.json` file.

- Add `nodemon --exec babel-node src/server.js` as the `start` script. This tells the nodemon package to watch for file changes, reload when it detects them and use babel-node to run the file `src/server.js` . We'll use this while developing locally.
- Add `babel src --out-dir dist` as the `build` script. This tells babel to compile the files from the `src` directory and place them in the `dist` directory.
- Add `node dist/server.js` as the `serve` script. This enables us to run our compiled code on a server, the reason we are not just using `nodemon` for this is it uses quite a bit more memory than just using `node` and adds some startup time to the process which is fine for some applications but can be a huge performance hit in others.

Your `package.json` should now probably look something like this:

```
{
  "name": "my-app",
  "version": "1.0.0",
  "description": "",
  "main": "src/server.js",
  "scripts": {
    "start": "nodemon --exec babel-node src/server.js",
    "build": "babel src --out-dir dist",
    "serve": "node dist/server.js"
  },
  "author": "",
  "license": "ISC",
  "dependencies": {
  },
  "devDependencies": {
    "@babel/cli": "^7.0.0-rc.1",
    "@babel/core": "^7.0.0-rc.1",
    "@babel/node": "^7.0.0-rc.1",
    "@babel/preset-env": "^7.0.0-rc.1",
```

```
    "nodemon": "^1.18.3"
  }
}
```

Happy developing and good luck!

If you have any questions or suggestions I'll see you in the comments or alternatively you can hit me up via twitter.