Sun-Li Beatteay

Follow 2K Followers About

How to Convert ES6 into ES5 using Babel

Make your ES6 code 100% browser supported

Sun-Li Beatteay Jul 6, 2017 · 3 min read

So you want to start using ES6 but are worried about browser support and backwards compatibility for your application? Not to worry, Babel can help us there.

Babel is a transpiler (translates code in one language to another computer language at the same abstraction level) that can turn our ES6 code into ES5. I'm going to assume you know how to use both node and npm and have them both installed on your machine. If you don't, start here.

use Babel in our terminal. Navigate to your project folder in your command line and type the following command:

```
$ npm install -D babel-cli
```

2. Write ES6 Code

If you already have ES6 code written then you're ahead of the game. If you don't, do that now. My suggestion would be to house your JavaScript files in a folder, such as `src` or `javascript`. I'm going to make an example file:

```
// src/example.js

class Hello {
  static world() {
    console.log('Hello, World!');
  }
}

Hello.world();
```

3. Install env preset and create .babelrc file

Now that we have our JavaScript code, we need to set up Babel to transpile it for us. Unfortunately, Babel can't do much out of the box. It needs the appropriate plugins and presets — curated groups of plugins — to do the work.

In our case, we'll be using `env` preset and installing it as a dev dependency in your project.

```
$ npm install -D babel-preset-env
```

Now we need our `.babelrc` file. Much like how npm relies on `package.json`, Babel uses `.babelrc` to keep track of the plugins and presets we'll be using.

Create your `.babelrc` file:

```
$ touch .babelrc
```

```
// .babelrc

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

4. Create a npm build command

You don't need to do this, but it'll make it faster and simpler if you need to transpile your code often.

Go to your `package.json` file where we'll create a npm `build` script using the `babel` command. The `babel` command takes two arguments: the path to your ES6 code and a path to where you want your ES5 code to go.

If you have all your JavaScript code housed in a directory, then you can add the `-d` flag to the command to tell Babel for look for directories instead of files. For my example, I have my JavaScript code in my `src` directory but want my ES5 code to be put in a `build` directory.

```
// package.json

...
"scripts": {
  "build": "babel src -d build",
},
...
```

5. Run the Babel command

With your `.babelrc` file created and your `build` command ready, just run it in your command line.

```
$ npm run build
```

You should see your converted JavaScript files in a `build` directory. Now all you need to do is include/require them in your application and you should be ready to go!

JavaScript Babeljs ES6 Nodejs NPM