Insert the Timer Module in the Pomodoro App

In order to integrate the timer into our application, we'll need to import it.

Exercise:

Wrap the stopwatch timer implemented in Exercise 15 in an ES6 module and import it in the Pomodoro App.

Source code:

Use the PomodoroTracker7 folder as a starting point. The end result is in PomodoroTracker8.

Solution:

This is an easy exercise in case you know how to use ES6 modules.

All you need to do is write `export default` in front of the class name:

```
export default class Timer {
    //...
}
```

Save the code in `Timer.js`, and place the JavaScript file in the same folder where your `pomodoro.js` file is. In your `index.html` file, reference `pomodoro.dist.js` instead of `pomodoro.js` in the script tag:

```
<script src="js/pomodoro.dist.js"></script>
```

When importing the `Timer` in another file, we have to prepend the statement:

```
import Timer from './Timer';
```

Let's also add some example code in the `pomodoro.js` file to test whether the `Timer` constructor function is accessible. We will count down from `25`

minutes, and display the results in the developer tools console.

```
new Timer( 25 * 60, console.log ).start();
```

in `pomodoro.js`. We need some setup work though, to make the browser understand these statements. Although there are multiple options to transpile ES6 modules, we will use the approach you learned in ES6 in Practice: we will install Webpack.

```
npm install webpack webpack-cli --save-dev
npm install babel-loader babel-core --save-dev
npm install babel-preset-es2015 babel-preset-stage-2 --save-dev
```

Once the installation is done, let's create a `webpack.config.js` file:

```
module.exports = {
    entry  : './js/pomodoro.js',
    output : {
        path     : __dirname,
        filename : 'pomodoro.dist.js'
    },
    module : {
        rules: [ {
                test   : /.js$/,
                loader : 'babel-loader',
                exclude: /node_modules/,
                query: {
                    presets: [
                        'es2015',
                        'stage-2'
                    ]
                }
            }
        ]
    }
};
```

If you are using an earlier version of webpack, you need to replace `rules` with the name `loaders`.
Let's place some code in our `package.json` file so that we can call Webpack from `npm`:

```
"scripts": {
    "webpack": "webpack",
    "webpack:watch": "webpack --watch"
  },
```

Now that the setup is done, we can run `npm run webpack`.

After creating the distribution file, you can load `index.html` in your browser. Open the developer tools, and check if you have access to the `Timer` class in the console. If you have access to `Timer`, you have finished this exercise successfully.