Lesson Weekend

Intermediate JavaScript (/intermediate-javascript) / Test-Driven Development and Environments with JavaScript (/intermediate-javascript/test-driven-development-and-environments-with-javascript) / Future Project Structure

Text

In this lesson, we'll provide an overview of what our file's project structure will look like by the end of this course section. We'll also briefly describe the purpose of each file. **Do not add any files to your project just yet.** We will add them as they are needed. This lesson just provides an introduction to how our file structure will change, including contrasting it with our project's current file structure.

Here's the structure of our project so far:

```
shape-tracker/
├── index.html
├── src
│   ├── index.js
│   └── triangle.js
└── css
    └── styles.css
```

Here is what a "built" project will look like by the **end of this section**:

```
shape-tracker/
├── dist
│   └── bundle.js
├── __tests__
│   ├── circle.test.js
│   └── triangle.test.js
├── src
│   ├── assets
│   │   └── images
│   │       └── image.jpg
│   ├── css
│   │   └── styles.css
│   ├── circle.js
│   ├── index.html
│   ├── index.js
│   └── triangle.js
├── package.json
├── package-lock.json
├── webpack.config.js
├── node_modules
├── .gitignore
├── .babelrc
├── .eslintrc
└── README.md
```

That is a lot more files! We will be creating some of these files manually and some will automatically be generated for us. **Once again, do not add any of these files now.** This lesson is just an overview of what our completed environment will look like at the end of this section.

Let's briefly summarize the purpose of each file. Remember, we will be exploring all these files in more depth soon — so these summaries aren't meant to be comprehensive.

- **`dist`** and **`bundle.js`**: webpack will automatically bundle all of the separate files (our JS, HTML, CSS, images, and other files) into one single file called `bundle.js`. Then, this file will be added to a directory called `dist`, which is short for distribution. This is where our finished code goes! We will never create this folder or file manually — webpack will do it for us. If we want to open our project in the browser, this is the file we'll open.
- **`__tests__`:** All of the tests we write will go in this directory. And yep, two underscores before and after is the naming convention! Each JavaScript file that contains business logic will have its own corresponding test file. Our finished project will have logic files for two shapes, meaning we will have a `circle.test.js` and a `triangle.test.js`.
- **`src`**: Our `src` directory will contain *all* of our source code, and not just our JavaScript files. We'll put our HTML and CSS files in here, too. webpack will look in the `src` directory and bundle everything together for us, turning it into `dist/bundle.js`.
- **`package.json`:** This file has a list of all of our project's dependencies (external JavaScript libraries that we call 'packages' that our project needs to function) so we can easily and automatically install them. It also contains other important information about our project.
- **`package-lock.json`:** This file will automatically be generated for us when we install our dependencies. It includes an exact list of all the packages that are currently installed. This list will differ

from the more general list in `package.json`, as many of the dependencies we'll use rely on yet other dependencies which will automatically be installed for us.

- **`webpack.config.js`:** This will hold our webpack configuration, providing specific instructions on how webpack should process and bundle our source code. This includes processes like minifying code to make it smaller and concatenating files together into one single file.
- **`node_modules`:** This directory contains all the dependencies our project needs. It is automatically generated for us and we should never edit anything in it, though sometimes we may need to delete this directory and rebuild it if we are having issues with our environment.
- **`.gitignore`:** This will include all the files that we don't ever want to push to GitHub, which include automatically generated files and files with sensitive information (like passwords). Since we'll automatically generate some of our files, there's no reason for those generated files to be in GitHub. Other developers can clone our project and automatically generate those files, too.
- **`.babelrc`:** This file holds our Babel configuration. Babel is a transpiler. A **transpiler** is a tool that translates more advanced versions of a language into another version of the same language. Babel handles translating more recent versions of JavaScript (like ES6 and ES versions 2016 — 2018) into an earlier version of JavaScript that all browsers implement. We'll only be using Babel to make sure Jest (our testing framework) can run properly, but Babel can also allow us to use cutting-edge JavaScript tools in browsers that don't have support for those tools yet.
- **`.eslintrc`:** This file holds our ESLint configuration. ESLint is a linter for JavaScript. A **linter** is a tool that analyzes code to alert developers about bugs and badly-written code.
- **`README.md`:** Our projects should always include a README. It becomes even more important once we start using an environment because we need to provide exact instructions

for other developers about how to set up the same environment when they clone our projects.

That's a lot of files.

It may seem over the top for smaller projects. Chances are, though, that down the road you won't just be building little projects. Enterprise applications are a lot bigger and more involved than this. Also, in the real world, it's very rare that you'll be building a project from scratch — you'll be jumping right into a much larger codebase.

In the next lesson, we'll actually start building out our environment further — starting with creating a `package.json` file.

Previous (/intermediate-javascript/test-driven-development-and-environments-with-javascript/basic-project-structure)
Next (/intermediate-javascript/test-driven-development-and-environments-with-javascript/git-best-practices-and-adding-a-gitignore-file)