Lesson Weekend

Intermediate JavaScript (/intermediate-javascript) / Test-Driven Development and Environments with JavaScript (/intermediate-javascript/test-driven-development-and-environments-with-javascript) / Adding A Production Dependency: Bootstrap

Text

We're almost done setting up our environment. Let's finish up by adding a dependency for our production code: Bootstrap!

The following instructions should inform you on the general process for adding any other JavaScript or style library that's required for your project to function in production. There's *a lot* to explore out there! While other libraries may require slightly different configurations, the process of installing a package and importing the code into `index.js` should remain the same.

By the end of this lesson, we'll have Bootstrap installed and configured in our project.

Bootstrap

Since the full capabilities of Bootstrap requires both JS and CSS files, we'll need to be sure to include both files in our project to use Bootstrap's styles AND interactive components.

Bootstrap relies on one dependency, Popper.js — but it is downloaded when installing Bootstrap via its npm package (https://github.com/twbs/bootstrap/tree/v5.2.3). Popper.js (https://popper.js.org/) is a tool that calculates the position of an element on the page, including pop ups and notifications. Bootstrap uses Popper.js for only some of its functionality, for example, dropdowns.

Installation

To install Bootstrap, we're going to follow these steps:

1. Install Bootstrap as a dependency, but not a dev dependency, because we'll use Bootstrap in production.
2. Import Bootstrap into the JS files you would like to use it in.

The version of Bootstrap that we'll use in our project is Bootstrap version 5.2.3. If you prefer, you can explore Bootstrap's most recent version instead of using the pinned version below.

In the root of the Shape Tracker directory, run the following command:

```
$ npm install bootstrap@5.2.3
```

If we open up `package.json` we can see that Bootstrap have been added to our list of dependencies:

package.json

```
{
  ...
  "dependencies": {
    "bootstrap": "^5.2.3",
  }
  ...
}
```

Now all we need to do is import Bootstrap into `index.js` so that it will be included in our bundle:

src/index.js

```
import 'bootstrap';
```

That takes care of adding the `js` files for Bootstrap. Now we need to import the CSS files. We are going to work with the minified CSS for Bootstrap. To do that we simply add another import statement to `index.js`:

src/index.js

```
...
import 'bootstrap';
import 'bootstrap/dist/css/bootstrap.min.css';
import './css/styles.css';
```

Note: In order for both your Bootstrap styles *and* your custom styles to render properly, make sure to place the `styles.css` import *after* the bootstrap import statements. This enables our custom styles to override Bootstrap styles where necessary. This is exactly what we did previously with our scripts prior to using webpack and npm.

With these configurations in place, we can now use Bootstrap in our projects!

Previous (/intermediate-javascript/test-driven-development-and-environments-with-javascript/improving-development-by-linting-code)
Next (/intermediate-javascript/test-driven-development-and-environments-with-javascript/configuration-reference-suggested-workflows-and-optional-review)

Lesson 19 of 49
Last updated January 19, 2023

disable dark mode

Epicodus (http://www.epicodus.com)