Lesson | Weekend

Intermediate JavaScript (/intermediate-javascript) / Test-Driven Development and Environments with JavaScript (/intermediate-javascript/test-driven-development-and-environments-with-javascript) / Bundling CSS with webpack Loaders

Text

In the last lesson, we bundled our JavaScript files with webpack. That's a good start — but we can also use webpack for bundling other files that aren't JavaScript, as long as we are using the right webpack loader.

In this lesson, we'll learn how to bundle our CSS and use webpack loaders for CSS.

Even though we'll only be working with one CSS file in our project, large scale projects often work with many CSS files, which means bundling them can make a noticeable difference in terms of performance.

By the end of this lesson, you should have webpack configured to bundle Shape Tracker's CSS along with its JS files.

Bundling CSS

First, we need to install two new project dependencies with npm at their pinned versions. In the terminal, navigate to the root of the Shape Tracker project and enter the following command.

```
$ npm install style-loader@1.3.0 css-loader@3.6.0 --save-dev
```

This will add two more dev dependencies to our `package.json` file. Our `node_modules` directory and `package-lock.json` file will also be updated automatically.

webpack Loaders

Next let's update our `webpack.config.js` file:

webpack.config.js

```
const path = require('path');

module.exports = {
  entry: './src/index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist')
  },
  module: {
    rules: [
      {
        test: /\.css$/,
        use: [
          'style-loader',
          'css-loader'
        ]
      }
    ]
  }
};
```

Here we configure our webpack module further by adding specific **rules**. The `test` line tells webpack where to look for files that match a certain file type. Note that the `test` line uses a regular expression to find files with a `.css` extension.

The `use` line then directs webpack to use the specified **loaders (https://webpack.js.org/concepts/loaders/)**. We have two loaders specified, and webpack knows to look for them in our project's dependencies.

A **loader** preprocesses a file before it's actually loaded by webpack. We need these loaders because webpack only understands JavaScript, not other assets such as CSS. The loaders will actually transform our CSS into JavaScript code so webpack can bundle it.

Adding `css/styles.css` to the `src` Directory

Next, let's move our `css` directory into our `src` directory. After all, it's part of our source code, too. Finally, we need to import our CSS into our `index.js` file so it is available for webpack to bundle. To import it, we'll add the following import statement:

src/index.js

```
import './css/styles.css';

...
```

Finally, we can remove the following line from `index.html`:

```
<link rel="stylesheet" href="./../css/styles.css">
```

The `<head>` now looks like this:

dist/index.html

```
<head>
  <script type="text/javascript" src="bundle.js"></script>
  <title>Shape Tracker</title>
</head>
```

It's time to `$ npm run build`. Now if we refresh the browser, we'll see our new CSS rule has been applied. No need for separate `<link>` tags for styles in our HTML!

Previous (/intermediate-javascript/test-driven-development-and-environments-with-javascript/bundling-javascript)
Next (/intermediate-javascript/test-driven-development-and-environments-with-javascript/processing-html-with-a-webpack-plugin)

Lesson 13 of 49
Last updated more than 3 months ago.