Lesson | Weekend

Intermediate JavaScript (/intermediate-javascript) / Test-Driven Development and Environments with JavaScript (/intermediate-javascript/test-driven-development-and-environments-with-javascript) / Improving Development with a Live Development Server

Text

We'll improve our developer experience further by using a live development server for webpack. With a **live development server**, any time we make changes to our code, it's automatically re-bundled and our browser window is reloaded.

Remote students at Epicodus should already be well-familiar with the Live Server development server, which reloads the webpage every time we make a change to our code base — our JS, CSS, or HTML. However, Live Server won't automatically live reload our projects now that we're using webpack to generate HTML and bundle our JS and CSS. That's because our source code that we edit is completely separate from the output files that webpack generates and adds to the `dist` folder, and that we serve to run the webpage.

By the end of this lesson, we'll have added a new package to our Shape Tracker project to handle live reloading our code in the browser as we make changes to our source code. This package is called webpack-dev-server. **From now on, we'll no longer use Live Server, and instead use webpack-dev-server to serve our webpack projects.**

webpack Development Server

We'll start by installing webpack-dev-server with npm. Navigate to the root of the Shape Tracker directory and run the following command.

```
$ npm install webpack-dev-server@3.11.3 --save-dev --save-e
xact
```

Now let's update `webpack.config.js` to use webpack-dev-server.

webpack.config.js

```
...

module.exports = {
  entry: './src/index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist')
  },
  devServer: {                  // new line
    contentBase: './dist'       // new line
  },                            // new line

  ...

};
```

Writing a New npm Script

If we run `$ npm run build`, our code will be re-bundled by webpack, but our webpack-dev-server won't start. In order to get our server to start, we actually need to invoke the server to open. We'll do this by creating a new npm script in the `"scripts"` section of `package.json`.

Let's add another script called `"start"`:

package.json

```
...
  "scripts": {
    "build": "webpack --mode=development",
    "start": "webpack-dev-server --open --mode=development"
  },
...
```

If we run `$ npm run start` in the command line, webpack will automatically spin up a development server at http://localhost:8080/. Try making a small change to the application code in the `src` folder. For instance, change the background color in `css/styles.css` to `red`. Once you save, the page will be updated instantly.

Note that when we run `$ npm run start`, webpack will not create a bundle and a `dist` folder. We need to `$ npm run build` for that. So, if we were cloning a project down from GitHub and starting it for the first time, we'd need to be sure to run `$ npm run build` to build the project, and then `$ npm run start` to start the development server.

We can make our lives easier by modifying `$ npm run start` to do both things: build the project and then start our development server. We just need to make the following modification to our `"scripts"`:

package.json

```
...
  "scripts": {
    "build": "webpack --mode=development",
    "start": "npm run build && webpack-dev-server --open --
mode=development"
  },
...
```

Note that the double ampersand `&&` simply chains two commands together. In this case, we're calling our own npm `"build"` script, and then opening the webpack-dev-server.

If webpack-dev-server Doesn't Work

Up to this point, remote learning students have been using VS Code's Live Server extension. When using webpack, though, remote students should use the webpack development server if possible. To do so, just run `$ npm run start` during a VS Code Live Share session. The server should automatically be shared via VS Code Live Share.

However, some remote students have issues with sharing webpack's development server. If this is the case, you may also use the VS Code Live Server extension you've been using to share servers up to this point.

To do so, you'll need to follow these steps after you make a change to your source code and you want to see it reflected in the browser:

- Run `$ npm run build` to compile the latest code.
- Run the Live Server extension on `dist/index.html` .

Note: For this section's independent project, you should be using webpack's development server since you won't be live sharing your code with others.

In the next lesson, we'll add the last of our tooling to improve our development — a linter.

Previous (/intermediate-javascript/test-driven-development-and-environments-with-javascript/improving-development-with-source-maps-for-debugging)
Next (/intermediate-javascript/test-driven-development-and-environments-with-javascript/improving-development-by-linting-code)

Lesson 17 of 49
Last updated more than 3 months ago.

disable dark mode

(http://www.epicodus.com)