

Lesson

Weekend

## Intermediate JavaScript (/intermediate-javascript)

### / Test-Driven Development and Environments with JavaScript

#### (/intermediate-javascript/test-driven-development-and-environments-with-javascript)

#### / Improving Development by Automating Clean Up Tasks

Text

Right now, the `dist` folder is empty. Run `$ npm run build` now and webpack will generate two files, `bundle.js` and `index.html`, and add them to the `dist` folder. In the future, anytime time we re-run `$ npm run build`, webpack creates a new `bundle.js` and `index.html` file and replaces any existing files in the `dist` folder.

Well, what if we change webpack's output to another name?

**webpack.config.js**

```
...  
  
module.exports = {  
  entry: './src/index.js',  
  output: {  
    filename: 'output.js', // new name!!  
    path: path.resolve(__dirname, 'dist')  
  },  
  ...  
}
```

If we re-run `$ npm run build`, our `dist` folder will now contains these three files:

```
dist/  
├─ index.html  
├─ bundle.js  
└─ output.js
```

Notably, webpack won't automatically delete the old `bundle.js` file. The same is true for any other assets that get added to our `dist` folder, like images. (We'll work with images later in this course section.)

In short, as we write code, add new assets, and change our config options, our `dist` folder will get cluttered. While we won't be changing the name of our bundle to `main.js` — we'll always stick with `bundle.js` — we can improve our developer experience by configuring webpack to automatically clean up after us. We'll do this with another plugin.

By the end of this lesson, you should have `CleanWebpackPlugin` installed and configured in your Shape Tracker project.

# Using CleanWebpackPlugin

We can configure webpack to use CleanWebpackPlugin (<https://www.npmjs.com/package/clean-webpack-plugin>) so that webpack clears the files inside of our `dist` folder before adding any new ones, each time run `$ npm run build` to bundle our code and generate an HTML file.

In the terminal, navigate to the root of the Shape Tracker project and enter the following command.

```
$ npm install clean-webpack-plugin@3.0.0 --save-dev
```

Then, we'll add the plugin to our configuration file:

```
webpack.config.js
```

```
...
const path = require('path');
const HtmlWebpackPlugin = require('html-webpack-plugin');
const { CleanWebpackPlugin } = require('clean-webpack-plugin'); // new line

module.exports = {
  entry: './src/index.js',
  output: {
    filename: 'bundle.js',
    path: path.resolve(__dirname, 'dist')
  },
  plugins: [
    new CleanWebpackPlugin(), // new line
    new HtmlWebpackPlugin({
      title: 'Shape Tracker',
      template: './src/index.html',
      inject: 'body'
    })
  ],
  module: {
    rules: [
      {
        test: /\.css$/,
        use: [
          'style-loader',
          'css-loader'
        ]
      }
    ]
  }
};
```

We've added two lines to our `webpack.config.js` file. The new lines have comments to indicate where they are.

Hopefully this pattern is starting to look familiar.

- First we require the new plugin and save it in a variable. This makes it so that we can call on the plugin within the `webpack.config.js` file.
- Then we add it to `module.exports` in the `plugins` section.
- Within `plugins`, we create a new instance of the plugin.

The default behavior for `CleanWebpackPlugin` is to clear the files inside of the `dist` folder. Now when we run `$ npm run build`, webpack will automatically clean out the contents of our `dist` folder before creating new bundle files.

Depending on your computer, you may or may not be able to actually see the files being deleted and then recreated. In fact, it looks like nothing is happening at all. If this is the case for you, there's additional configurations we can set for `CleanWebpackPlugin` that will confirm that our files are being deleted. Update `new CleanWebpackPlugin()` with this code:

#### **webpack.config.js**

```
new CleanWebpackPlugin({  
  verbose: true  
})
```

Setting `verbose: true` will tell webpack to include terminal output about `CleanWebpackPlugin`. With this configuration set, when we now run `$ npm run build`, we'll see two new lines in the terminal output:

- `clean-webpack-plugin: removed dist\bundle.js`
- `clean-webpack-plugin: removed dist\index.html`

See if you can find those new lines in the output below!

```

$ npm run build

> shape-tracker@1.0.0 build
> webpack --mode=development

clean-webpack-plugin: removed dist\bundle.js
clean-webpack-plugin: removed dist\index.html
Hash: 9f99a3aa8081bb8f3eaf
Version: webpack 4.46.0
Time: 765ms
Built at: 06/27/2022 1:48:28 PM
      Asset      Size  Chunks             Chunk Names
  bundle.js    18 KiB       0  [emitted]  main
  index.html   631 bytes       0  [emitted]
Entrypoint main = bundle.js
[./node_modules/css-loader/dist/cjs.js!./src/css/styles.css] 186
bytes {main} [built]
[./src/css/styles.css] 410 bytes {main} [built]
[./src/index.js] 789 bytes {main} [built]
[./src/triangle.js] 256 bytes {main} [built]
+ 2 hidden modules
Child html-webpack-plugin for "index.html":
   1 asset
  Entrypoint undefined = index.html
[./node_modules/html-webpack-plugin/lib/loader.js!./src/index.html] 805 bytes {0} [built]
[./node_modules/webpack/buildin/global.js] (webpack)/bu
ildin/global.js 472 bytes {0} [built]
[./node_modules/webpack/buildin/module.js] (webpack)/bu
ildin/module.js 497 bytes {0} [built]
+ 1 hidden module

```

**Note that adding the configuration `verbose: true` is optional.** You can add it if you want to your own Shape Tracker project, but we won't include it in the LearnHowToProgram's Shape Tracker project.

We could also configure this plugin to clean multiple directories and exclude specific files, as well, but that's for further exploration.

In the next lesson, we'll add another tool that's great for development: source maps.

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

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

Lesson 15 of 49

Last updated more than 3 months ago.

disable dark mode



© 2023 Epicodus (http://www.epicodus.com/), Inc.