Notes - Webpack - Basic Usage

- Dr Nick Hayward

A brief introduction to using Webpack bundler.

Contents

Intro

Webpack is a utility for bundling modules and project code for distribution release of browser compatible apps.

Getting started details may be found at the following URL,

- https://webpack.js.org/guides/getting-started/

Setup

We may use Webpack to prepare apps with ES2015, CommonJS &c. modules for publication online.

Start by creating an npm project at the root of the project's CWD,

```
npm init
```

and answer the series of questions for the current project. Or, we may simply accept the defaults using the `-y` flaf,

```
npm init -y
```

Install Webpack and dependencies,

```
npm install webpack webpack-cli --save-dev
```

Then modify `package.json`

- add `"private": true,`
- remove `"main": "index.js",`

The `private` property ensures this NPM project is not published to NPM.

We remove the `main` property to allow Webpack to manage project dependencies and bundling.

Sample project directory structure

Then, we'll create a sample application structure for use with Webpack.

e.g.

```
|- /dist
                - index.html
                - main.js
|- /node_modules
|- /src
                - index.js
|- package-lock.json
|- package.json
```

Initial code

Then, we may add some initial project code to

- `/dist/index.html`
- `/src/index.js`

and add a reference to the `main.js` file in `index.html` for distribution usage of the app.

This file, `dist/main.js`, will hold the minified and optimised JS code for the app from `src/index.js`.

Develop the JS code

Add any required JS code to `src/index.js`.

This is the entry point for dependencies and modules, which are then processed by Webpack into the app's `dist/main.js` file.

So, we may `import` and `export` any required NPM or custom modules, and load the app via this file, `src/index.js`.

Run Webpack

We may then test build this code and app structure with Webpack.

In the root of the app's CWD, issue the following terminal command

```
```

```
npx webpack
```

This will process the `src` directory files, and any dependencies, and output the distribution app in the `dist` directory.

Run the app

After successful running Webpack, and minifying and optimising our app's code, we may run the app from the `dist` directory.

The `dist` directory is the root of the app's directory structure.

Custom config file

Webpack allows us to also specify a custom config file for build settings.

At the root of the project directory, we may now create a config file,

```
|- webpack.config.js
```

and then some initial settings,

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

module.exports = {
        mode: 'production',
  entry: './src/index.js',
  output: {
    filename: 'main.js',
    path: path.resolve(__dirname, 'dist')
  }
};
```

So, we might modify this config to support an alternative start file, entry, for the app's JavaScript logic. We may also modify the structure of the distribution output, again specifying alternative filenames and directories.

To run this file, we may, for example, explicitly call the config filename, `webpack.config.js` with the webpack command.

```
npx webpack --config webpack.config.js
```

As this config filename is the default, we do not need to explictly append the `--config` flag and filename to this command. However, we may still use this flag with an alternative config filename.

NPM `scripts` property

We may also add a call to `webpack` as a `build` property in the `scripts` object of NPM's package.json file.

e.g.

```
...
"scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
                "build": "webpack"
  },
...
```

Then, we may simply issue the standard build command using `npm run`,

```
npm run build
```