Webpack introduction slide

This slide give you introduction to webpack setup including webpack dev server, CSS Loader and HTML templating.

By Sumit Kohli
FullStack Consultant
https://in.linkedin.com/in/sumit-kohli-234a962

What is webpack

Webpack takes a group of different assets, different files - of different types; JavaScript, images, SASS etc and combines them altogether (bundles them) into a smaller group of files,

Dependency Managment

it also manages dependencies of an application, so that codes that need to load first - load first.

Lets set up project

Folder setup

mkdir webpackdemo

mkdir src –run inside webpackdemo

mkdir dist –run inside webpackdemo

touch index.js inside src

touch webpack.config.js – inside webpack demo

Run the command inside webpackdemo. “npm init -y “

Install webpack and webpackcli and initialize package.json

“npm i --save webpack webpack-cli “ this command to run inside webpackdemo folder

`Webpack` allows us to produce bundled JavaScript modules and the the `webpack-cli` is a tool that provides a set of commands for webpack

Webpack config file basic config

```
const path = require("path");
module.exports = {
  mode: "development", // none,production,development are values
  entry: {
    main: path.resolve(__dirname, "src/index.js"),
  },
  output: {
    path: path.resolve(__dirname, "dist"),
    //filename: "index.bundle.js", // using static name
    // OR
    // using the entry name, sets the name to main, as specific in the entry object.
    filename: "[name].bundle.js",
    clean: true,
  },
};
```

Webpack config with dynamic name

We add contenthash in name to clear browser cache when it loads a new build

```
const path = require("path");
module.exports = {
  mode: "development",
  entry: {
    main: path.resolve(__dirname, "src/index.js"),
  },
  output: {
    path: path.resolve(__dirname, "dist"),
    filename: "[name].[contenthash].js",
    clean: true,
  },
};
```

Note: The hash will change only when code has changed. Else it remains same!

Webpack Plugins

`HTMLWebpackPlugin` :Makes it possible to dynamically create HTML files

To Install :

npm install --save-dev html-webpack-plugin

Now you need to include HtmlWebpackPlugin with require and then add it to plugins array where we call our plugins

Add the following code after output settings object.

```
plugins: [
  new HtmlWebpackPlugin({
    title: "Adding HTML File",
    filename: "index.html",
  }),
],
```

WHen you run npm run build you see a HTML file is created in the dist folder. And in the index.html file, you will notice that the unique js file that had been bundled has been automatically appended:`

What if you have existing html file?

If you already have html file then you can achieve this by attaching a template option into the plugins option of the config file, then define a template file in the src folder which would be the value of the template option:

Below is the configuration for it:

```
 plugins: [
    new HtmlWebpackPlugin({
      title: "Testing html file",
      filename: "index.html",
      template: path.resolve(__dirname, "src/base.html"),
    }),
  ],
```

What are Loaders?

Webpack does not know what an image file is, or HTML, CSS, SASS, or SVG files are, including other types of files; it does not know how to handle them.For handling and bundling such assets, we use loaders.
Lets install them first.

```
npm install --save-dev style-loader css-loader
```

To understand CSS loaders, lets first create a css file named style.css in src folder.

```
h1 {
        border: 4px solid blue;
        padding: 10px;
        margin: 10px;
}
```

we have created a rule that provides an array of loaders for loading any CSS files in the `src` folder; being read from right to left. The CSS-loader looks for the file, turns it into a module and gives it to JavaScript, while the `style-loader` takes the imported module, and injects it into the `dist/index.html` file.

Setting config for css and style loader

In the `loaders` option in the config file, we can now set rules, for loading various types of files as modules

Add to webpack.config.js file

```
module: {
  rules: [{ test: /\.css$/, use: ["style-loader", "css-loader"] }],
 },
```

Let setup dev server.

We need to install webpack dev server to server our html file.
npm i –D webpack-dev-server

Add these config to webpack config file

```
devServer: {
    contentBase: path.resolve(__dirname, "dist"),
    port: 3001, // default is often 8080
    open: true,
    hot: true,
    watchFiles: [path.resolve(__dirname, 'src')],
  },
```

Add this command to package.json file. **"dev": "webpack serve"**

Run the command npm run dev and on port . now you can see the application running on port 3001 with css loader working!