Webpack Structure

**webpack-demo**

This repo will demonstrate some webpack setups that can be used in your projects.

**Webpack to build combined assets**

Below each webpack cli command is explained.

**$ npm run webpackCombined**

This creates JS files that are combined into one file. The CSS files, from standard css or scss, are created as separate files or as combined, depending on how the entry point is defined.

Note: GLOBs can be used to create a collection of files of the same extension, which can be set as a single entry point and will combine all files into one CSS file. See line 134 of webpack.config.separate-output.js and line 146 of webpack.config.combined-output.js.

The output can be seen on:

[**http://localhost:10001/resulting-index-combined-js.html**](http://localhost:10001/resulting-index-combined-js.html)

**Webpack to build separate assets**

**$ npm run webpackSeparate**

This creates JS files that are separate, but referenced by a main JS file which needs to be used in the index.html (or whatever). All the CSS is put into same file, *styles.css*.

The output can be seen on:

[**http://localhost:10002/resulting-index-separate-js.html**](http://localhost:10002/resulting-index-separate-js.html)

**Merged webpack instance**

I created a final webpack instance consisting of a generic base webpack, inside of which global configs are defined, such as loaders to use, plugins to use, definitions of devMode etc. They are merged via an npm package, 'webpack-merge' (<https://www.npmjs.com/package/webpack-merge>), and the script can be fired which will start a build. The output of which is the same as the separate instance.

The output can be seen on:

[**http://localhost:10003/resulting-index-merged-webpack-instance.html**](http://localhost:10003/resulting-index-merged-webpack-instance.html)

**Each webpack instance**

Each webpack on the whole has he same setup, some of the entry points are defined differently, but this is to show the various ways that entry points can be defined; via an import in a js file, via a manual array of files or via a glob patter to find all files of a given type in a given location.

All imported and required node modules have been described with links to where they were taken from.

--

**Webpack structure**

The structure of each webpack is rather similar, so the general structure will be explained.

A webpack is made up of the sections

module.exports = {

    /\*\*

     \* Detect env from .env file in project root.

     \* If no enVars found, defaults to production,

     \* to prevent any chance of dev code making it live.

     \*/

    mode: envVars.NODE\_ENV || "production",

    entry: {

        /\*\*

         \* Main entry points,

         \* i.e. where all the sass files are referenced.

         \* 'entry' instances reference js files which can

         \* import js modules and also reference scss/css

         \* files, where scss files can also reference other scss files.

         \*/

    },

/\*\*

\* Resolve paths

\* <https://webpack.js.org/configuration/resolve/#root>

\*\*/

resolve: {

        /\*\*

         \* Path to modules,

         \*/

        modules: [

path.resolve(\_\_dirname, ‘node\_modules’),

path.resolve(\_\_dirname, ‘node\_modules/specific/module’),

],

        alias: {

            /\*\*

             \* Keys defined for image paths

             \* These keys can then be used in Scss files, js files etc

             \* This saves any requirement to make abs paths

\* (i.e. ../../../../file.js)

             \*/

            "$AliasName": path.resolve(\_\_dirname, "path/to/dir"),

        },

        /\*\*

         \* Extensions of files where these keys \*should\* be

\* accessible from

         \*/

        extensions: [".js", ".json", ".scss", ".vue", ".css"],

    },

    output: {

        /\*\*

         \* Path to output to.

         \* Defaults to ./dist if left out.

         \* The output can also be overridden via the

         \* command: webpack -o output/path/file.js

         \* every file emitted to your output.path

         \* directory will be referenced from the

         \* output.publicPath location..

\* <https://webpack.js.org/guides/public-path/>

         \*/

        filename: "[name].js",

        path: path.resolve(\_\_dirname, "dist "),

        publicPath: "/dist/",

    },

    optimization: {

/\*\*

\* Optimisations such as minify, chunkify

\* Terser is one plugin, it has a decent set

\* Options to optimise output.

\* <https://webpack.js.org/configuration/optimization/>

\*\*/

        minimizer: [ new TerserPlugin() ],

    },

    plugins: [

        /\*\*

         \* Clear out old files when recompiling

         \*/

        new CleanWebpackPlugin(),

        /\*\*

        \* Copy files during the build process, images etc

        \*/

        new copyWebpackPlugin(),

        /\*\*

        \* Register the vue loader plugin to use it

        \* to process vue files.

        \*/

        new VueLoaderPlugin(),

        /\*\*

        \* Register the vue loader plugin to use it

        \* to process vue files.

        \*/

        new TerserPlugin()

    ],

    module: {

        rules: [

            /\*\*

             \* Loaders to create and format the

             \* CSS on build

             \* NOTE: Build steps for module rules run in reverse!

             \* I.E. from bottom to top.

             \*/

            {

                test: /\.(sass|scss)$/,

                exclude: [path.resolve(\_\_dirname, "vue.app")],

                use: [

                    {

/\*\*

\* The file-loader resolves import/require()

\* on a file into a url and emits the file into the

\* output directory.

\*\*/

                        loader: "file-loader",

                        options: {

                            name: "[name].css",

                            context: "./",

                            outputPath: "./css/",

                        },

                    },

/\*\*

\* webpack loader to extract HTML and CSS from the bundle.

\* <https://webpack.js.org/loaders/extract-loader/#root>

\*\*/

                    {

                        loader: "extract-loader",

                    },

/\*\*

\* https://The css-loader interprets @import and url() like

import/require() and will resolve them.

\* <https://webpack.js.org/loaders/css-loader/#root>

\*\*/

                    {

                        loader: "css-loader",

                        options: {

                            sourceMap: devMode ? true : false,

                            importLoaders: 1,

                            url: true,

                        },

                    },

/\*\*

\* https://Loader for webpack to process CSS with PostCSS

\* <https://webpack.js.org/loaders/postcss-loader/#root>

\*\*/

                    {

                        loader: "postcss-loader",

                        options: {

                            parser: "postcss-scss",

                            ident: "postcss",

                            plugins: () => getPostCssPlugins(),

                            minimize: devMode ? false : true,

                        },

                    },

/\*\*

\* https://Loads a Sass/SCSS file and compiles it to CSS.

\* <https://webpack.js.org/loaders/sass-loader/#root>

\*\*/

                    {

                        loader: "sass-loader",

                        options: {

                            sourceMap: devMode ? true : false,

                            sassOptions: {

                                outputStyle: devMode

                                    ? "expanded"

                                    : "compressed",

                            },

                            /\*\*

                             \* Pass in the env var to the sass files,

                             \* defined in this case as $environment.

                             \* As many other vars can be defined and passed

                             \* into the sass in the same way

                             \*/

                            prependData: (loaderContext) => {

                                const {

                                    resourcePath,

                                    rootContext,

                                } = loaderContext;

                                const relativePath = path.relative(

                                    rootContext,

                                    resourcePath

                                );

                                if (

                                    relativePath ===

                                    "assets\\scss\\webpack-demo-scss.alt.scss"

                                ) {

                                    return (

                                        "$MyVar:" +

                                        envVars.MyVar +

                                        "; $MyVar2: 'something-for-alt-scss';"

                                    );

                                }

                                return (

                                    "$MyVar:'" +

                                    envVars.MyVar +

                                    "'; $MyVar2: 'something-for-scss';"

                                );

                            },

                        },

                    },

                ],

            },

            /\*\*

             \* This is for standard CSS files, i.e. not sass/less

             \* or any other css files that need compiling into CSS

             \*/

            {

                test: /\.css$/,

                exclude: [path.resolve(\_\_dirname, "vue.app")],

                use: [

                    {

                        loader: "file-loader",

                        options: {

                            name: "[name].css",

                            context: "./",

                            outputPath: "./css/",

                        },

                    },

                    {

                        loader: "extract-loader",

                    },

                    {

                        loader: "css-loader",

                        options: {

                            sourceMap: devMode ? true : false,

                            url: false,

                        },

                    },

                    {

                        loader: "postcss-loader",

                        options: {

                            parser: "postcss-scss",

                            ident: "postcss",

                            plugins: () => getPostCssPlugins(),

                            minimize: devMode ? false : true,

                        },

                    },

                ],

            },

            /\*\*

             \* Loaders for processing VUE files to resolve paths,

             \* follow imports, extract CSS and extract and compile

             \* any SCSS

             \*/

            {

                test: /\.js$/,

                use: [

                    {

             /\*\*

             \* This package allows transpiling JavaScript files using Babel

             \* and https://webpack.webpack.js.org/loaders/babel-loader/#root

             \*\*/

                        loader: "babel-loader",

                        query: {

                            presets: [require.resolve("babel-preset-env")],

                        },

                    },

                ],

                include: [path.resolve(\_\_dirname, "vue.app")],

                exclude: [path.resolve(\_\_dirname, "node\_modules")],

            },

             /\*\*

             \* vue-loader is a loader for webpack that allows you to author

\* Vue components in a format called Single-File Components

\* (SFCs).

\* <https://github.com/vuejs/vue-loader>

             \*\*/

            {

                test: /\.vue$/,

                loader: "vue-loader",

                include: [path.resolve(\_\_dirname, "vue.app")],

                options: {

                    plugins: (loader) => {

                        new VuetifyLoaderPlugin();

                    },

                },

                exclude: [path.resolve(\_\_dirname, "node\_modules")],

            },

             /\*\*

             \* Vue-style-loader is used to dynamically inject CSS into the

             \* document as style tags

             \* <https://github.com/vuejs/vue-style-loader>

             \*\*/

            {

                test: /\.s?css$/,

                include: [path.resolve(\_\_dirname, "vue.app")],

                use: ["vue-style-loader", "css-loader", "sass-loader"],

            },

        ],

    },

};

**Package.json**

The scripts have been added here:

* "wpsep": Fire a build process for the separate file output.
* "wpcom": Fire a build process for the combined file output.
* "wpmerged": Fire the build process using the merged webpack instance.
* "httpserver": Simple local server to make specified folders available on a given port in a browser.
* "srvcom": Serve the combined files output on the specified port.
* "srvsep": Serve the separate files output on the specified port.
* "srvmerged": Serve the merged webpack instance files output on the specified port.
* "sep": Build and then serve the separate files output on the specified port.
* "combined": Build and then serve the combined files output on the specified port.
* "merged": Build and then serve the merged webpack instance files on the specified port."

The above commands are executed as

**$npm run [script-name]**