const path = require('path');

module.exports = {

entry: "./app/entry", // string | object | array

// Here the application starts executing

// and webpack starts bundling

output: {

// options related to how webpack emits results

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

// the target directory for all output files

// must be an absolute path (use the Node.js path module)

filename: "bundle.js", // string

// the filename template for entry chunks

publicPath: "/assets/", // string

// the url to the output directory resolved relative to the HTML page

library: "MyLibrary", // string,

// the name of the exported library

libraryTarget: "umd", // universal module definition

// the type of the exported library

/\* Advanced output configuration (click to show) \*/

},

module: {

// configuration regarding modules

rules: [

// rules for modules (configure loaders, parser options, etc.)

{

test: /\.jsx?$/,

include: [

path.resolve(\_\_dirname, "app")

],

exclude: [

path.resolve(\_\_dirname, "app/demo-files")

],

// these are matching conditions, each accepting a regular expression or string

// test and include have the same behavior, both must be matched

// exclude must not be matched (takes preferrence over test and include)

// Best practices:

// - Use RegExp only in test and for filename matching

// - Use arrays of absolute paths in include and exclude

// - Try to avoid exclude and prefer include

issuer: { test, include, exclude },

// conditions for the issuer (the origin of the import)

enforce: "pre",

enforce: "post",

// flags to apply these rules, even if they are overridden (advanced option)

loader: "babel-loader",

// the loader which should be applied, it'll be resolved relative to the context

// -loader suffix is no longer optional in webpack2 for clarity reasons

// see webpack 1 upgrade guide

options: {

presets: ["es2015"]

},

// options for the loader

},

{

test: /\.html$/,

use: [

// apply multiple loaders and options

"htmllint-loader",

{

loader: "html-loader",

options: {

/\* ... \*/

}

}

]

},

{ oneOf: [ /\* rules \*/ ] },

// only use one of these nested rules

{ rules: [ /\* rules \*/ ] },

// use all of these nested rules (combine with conditions to be useful)

{ resource: { and: [ /\* conditions \*/ ] } },

// matches only if all conditions are matched

{ resource: { or: [ /\* conditions \*/ ] } },

{ resource: [ /\* conditions \*/ ] },

// matches if any condition is matched (default for arrays)

{ resource: { not: /\* condition \*/ } }

// matches if the condition is not matched

],

/\* Advanced module configuration (click to show) \*/

},

resolve: {

// options for resolving module requests

// (does not apply to resolving to loaders)

modules: [

"node\_modules",

path.resolve(\_\_dirname, "app")

],

// directories where to look for modules

extensions: [".js", ".json", ".jsx", ".css"],

// extensions that are used

alias: {

// a list of module name aliases

"module": "new-module",

// alias "module" -> "new-module" and "module/path/file" -> "new-module/path/file"

"only-module$": "new-module",

// alias "only-module" -> "new-module", but not "only-module/path/file" -> "new-module/path/file"

"module": path.resolve(\_\_dirname, "app/third/module.js"),

// alias "module" -> "./app/third/module.js" and "module/file" results in error

// modules aliases are imported relative to the current context

},

/\* alternative alias syntax (click to show) \*/

/\* Advanced resolve configuration (click to show) \*/

},

performance: {

hints: "warning", // enum

maxAssetSize: 200000, // int (in bytes),

maxEntrypointSize: 400000, // int (in bytes)

assetFilter: function(assetFilename) {

// Function predicate that provides asset filenames

return assetFilename.endsWith('.css') || assetFilename.endsWith('.js');

}

},

devtool: "source-map", // enum

// enhance debugging by adding meta info for the browser devtools

// source-map most detailed at the expense of build speed.

context: \_\_dirname, // string (absolute path!)

// the home directory for webpack

// the entry and module.rules.loader option

// is resolved relative to this directory

target: "web", // enum

// the environment in which the bundle should run

// changes chunk loading behavior and available modules

externals: ["react", /^@angular\//],

// Don't follow/bundle these modules, but request them at runtime from the environment

stats: "errors-only",

// lets you precisely control what bundle information gets displayed

devServer: {

proxy: { // proxy URLs to backend development server

'/api': 'http://localhost:3000'

},

contentBase: path.join(\_\_dirname, 'public'), // boolean | string | array, static file location

compress: true, // enable gzip compression

historyApiFallback: true, // true for index.html upon 404, object for multiple paths

hot: true, // hot module replacement. Depends on HotModuleReplacementPlugin

https: false, // true for self-signed, object for cert authority

noInfo: true, // only errors & warns on hot reload

// ...

},

plugins: [

// ...

],

// list of additional plugins

/\* Advanced configuration (click to show) \*/

}