const path = require('path');

const webpack = require('webpack');

const ExtractTextPlugin = require('extract-text-webpack-plugin');

const CheckerPlugin = require('awesome-typescript-loader').CheckerPlugin;

const bundleOutputDir = './wwwroot/dist';

module.exports = (env) => {

const isDevBuild = !(env && env.prod);

return [{

stats: { modules: false },

entry: { 'main': './ClientApp/boot.tsx' },

resolve: { extensions: ['.js', '.jsx', '.ts', '.tsx'] },

output: {

path: path.join(\_\_dirname, bundleOutputDir),

filename: '[name].js',

publicPath: 'dist/'

},

module: {

rules: [

{ test: /\.tsx?$/, include: /ClientApp/, use: 'awesome-typescript-loader?silent=true' },

{ test: /\.css$/, use: isDevBuild ? ['style-loader', 'css-loader'] : ExtractTextPlugin.extract({ use: 'css-loader?minimize' }) },

{ test: /\.(png|jpg|jpeg|gif|svg)$/, use: 'url-loader?limit=25000' }

]

},

plugins: [

new CheckerPlugin(),

new webpack.DllReferencePlugin({

context: \_\_dirname,

manifest: require('./wwwroot/dist/vendor-manifest.json')

})

].concat(isDevBuild ? [

// Plugins that apply in development builds only

new webpack.SourceMapDevToolPlugin({

filename: '[file].map', // Remove this line if you prefer inline source maps

moduleFilenameTemplate: path.relative(bundleOutputDir, '[resourcePath]') // Point sourcemap entries to the original file locations on disk

})

] : [

// Plugins that apply in production builds only

new webpack.optimize.UglifyJsPlugin(),

new ExtractTextPlugin('site.css')

])

}];

};