Webpack Installation Steps

1. Create a folder with the name webpack-example and open the folder path into command prompt.
2. Type npm -init on command prompt which will create a package.json file in which we will save our project dependencies as well as development dependencies. Keep all settings default while creating package.json file.
3. We need angular packages to create our first Angular application. For that, we will install basic angular packages to create a simple application. Write below command on your command prompt -

npm install --save @angular/compiler @angular/core @angular/common @angular/platform-browser @angular/platform-browser-dynamic rxjs core-js zone.js

1. Let's get ready with webpack packages so that we can set our development environment for build process. This will include development build as well as production build. Write below commands on your command prompt -

npm install --save-dev webpack webpack-dev-server webpack-merge rimraf typescript @types/node

npm install --save-dev angular2-template-loader awesome-typescript-loader css-loader file-loader html-loader null-loader raw-loader style-loader to-string-loader

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

1. After installing all the development dependencies, now we will create webpack configuration. Open webpack-example folder into Visual Studio Code and add a new folder with the name config into webpack-example folder.
2. Add below three files into config folder -
   1. webpack.common.js
   2. webpack.dev.js
   3. webpack.prod.js
3. Now let's add webpack.config.js file at the root of your project that is webpack-example folder and write below code in the same -

module.exports = require('./config/webpack.dev.js');

1. Now open webpack.common.js file and add below code -

var webpack = require('webpack');

var HtmlWebpackPlugin = require('html-webpack-plugin');

const path = require('path');

const rootDir = path.resolve(\_\_dirname);

module.exports = {

entry: {

'polyfills':'./polyfills.ts',

'vendor':'./vendor.ts',

'app': './app/main.ts',

},

resolve: {

extensions: ['.ts', '.js']

},

module:{

rules:[

{

test:/\.ts$/,

loaders: [

{

loader: 'awesome-typescript-loader',

options: { configFileName: './tsconfig.json' }

}, 'angular2-template-loader'

]

},

{

test:/\.ts$/,

loaders: [

]

},

{

test: /\.html$/,

loader: 'html-loader'

}

]

},

plugins:[

new webpack.optimize.CommonsChunkPlugin({

name: ['app', 'vendor', 'polyfills']

}),

new HtmlWebpackPlugin({

template: './index.html'

}),

new webpack.ContextReplacementPlugin(

/angular(\\|\/)core/,

path.resolve(\_\_dirname)

)

]

};

1. Now let's open webpack.dev.js file from config folder and add below code -

//Dev env settings

var webpackMerge = require('webpack-merge');

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

var commonConfig = require('./webpack.common.js');

const path = require('path');

const rootDir = path.resolve(\_\_dirname);

module.exports = webpackMerge(commonConfig, {

devtool: 'cheap-module-eval-source-map',

output: {

path: path.resolve(rootDir, 'dist'),

publicPath: 'http://localhost:3000/',

filename: '[name].js',

chunkFilename: '[id].chunk.js'

},

plugins: [

new ExtractTextPlugin('[name].css')

],

devServer: {

historyApiFallback: true,

stats: 'minimal'

}

});

1. Now let's open webpack.prod.js file from config folder and add below code -

//Prod env settings

var webpackMerge = require('webpack-merge');

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

var commonConfig = require('./webpack.common.js');

const path = require('path');

const rootDir = path.resolve(\_\_dirname, '..');

module.exports = webpackMerge(commonConfig, {

output: {

path: path.resolve(rootDir, 'dist'),

publicPath: './',

filename: '[name].js',

chunkFilename: '[id].chunk.js'

},

plugins: [

new ExtractTextPlugin('[name].css')

],

});

1. Next step is to add two files into root folder of our application as below -
   1. polyfills.ts
   2. vendor.ts
2. Now add below code into our polyfills.ts -

import 'core-js/es6';

import 'core-js/es7/reflect';

import 'zone.js/dist/zone';

if (process.env.ENV === 'production') {

}

else {

Error['stackTraceLimit'] = Infinity;

require('zone.js/dist/long-stack-trace-zone');

}

1. Now add below code into our vendor.ts -

// Angular

import '@angular/core';

import '@angular/common';

import '@angular/platform-browser';

import '@angular/platform-browser-dynamic';

import 'rxjs';

//Other Vendors like jQuery, Bootstrap

1. Finally, we will now create our first Angular Application. Create a app folder at the root of our application. Add below three files in the app folder -
   1. app.module.ts
   2. app.component.ts
   3. main.ts
2. Now write below code into our app.module.ts -

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

@NgModule({

imports: [

BrowserModule,

],

declarations: [AppComponent],

providers: [/\* TODO: Providers go here \*/],

bootstrap: [AppComponent],

})

export class AppModule { }

1. Now open app.component.ts file and add below code -

import { Component } from '@angular/core';

@Component({

selector: 'my-app',

template: `

<h1>Welcome To Synechron Pvt. Ltd.</h1>

<hr/>

<h4>Bangalore Development Division!</h4>

`

})

export class AppComponent { }

1. Now open main.ts file and add below code -

import { platformBrowserDynamic } from '@angular/platform-browser-dynamic';

import { AppModule } from './app.module';

platformBrowserDynamic().bootstrapModule(AppModule);

1. After creating our Angular application, we need to configure Typescript compiler which will compile our Typescript code into plain JavaScript code.
2. Add a tsconfig.json file in the root folder of our application and write below code -

{

"compilerOptions": {

"target": "es5",

"module": "commonjs",

"strict": true,

"sourceMap": false,

"noImplicitAny": true,

"moduleResolution": "node",

"experimentalDecorators": true,

"emitDecoratorMetadata": true,

"noStrictGenericChecks": true,

"skipLibCheck": true,

"typeRoots": [

"node\_modules/@types"

],

"lib":[

"es2015",

"dom"

]

}

}

1. Next step is to configure our application's run command where we can start our application in development mode or produce a production build. For this, you will open package.json file and replace script section with below code -

"scripts": {

"start": "webpack-dev-server --inline --progress --port 3000",

"build": "rimraf dist & webpack --config ./config/webpack.prod.js"

},

1. Finally, we will add index.html page into our root folder and use our Angular application by adding <my-app></my-app> component selector into the body of our Html page as shown below -

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<meta http-equiv="X-UA-Compatible" content="ie=edge">

<title>Webpack Example</title>

</head>

<body>

<my-app>loading...</my-app>

</body>

</html>

1. Now, go back to command prompt and type a command npm start. You will see your application build run successfully. Once the build is complete, open your browser and type http://localhost:3000

Configure Bootstrap CSS Framework

1. Now to configure Bootstrap CSS framework, we will have to first install bootstrap framework and its dependencies. Bootstrap is dependent on jQuery. So, let install jQuery and Bootstrap. Open your command prompt and write below command -

npm install --save jquery bootstrap@3.3.7

1. Next, open vendor.ts file and below code -

// Angular

import '@angular/core';

import '@angular/common';

import '@angular/platform-browser';

import '@angular/platform-browser-dynamic';

import 'rxjs';

//Other Vendors like jQuery, Bootstrap

import 'jquery';

import 'bootstrap/dist/js/bootstrap';

import 'bootstrap/dist/css/bootstrap.min.css';

1. Now open webpack,common.js file from config folder and add below code -

var webpack = require('webpack');

var HtmlWebpackPlugin = require('html-webpack-plugin');

const path = require('path');

const rootDir = path.resolve(\_\_dirname);

module.exports = {

entry: {

'polyfills':'./polyfills.ts',

'vendor':'./vendor.ts',

'app': './app/main.ts',

},

resolve: {

extensions: ['.js', '.ts']

},

module:{

rules:[

{

test:/\.ts$/,

loaders: [

{

loader: 'awesome-typescript-loader',

options: { configFileName: './tsconfig.json' }

}, 'angular2-template-loader'

]

},

{

test:/\.ts$/,

loaders: [

]

},

{

test: /\.html$/,

loader: 'html-loader'

},

{

test: /\.css$/,

exclude: path.resolve(\_\_dirname, 'app'),

loader: "style-loader!css-loader?root=."

},

{

test: /\.css$/,

include: path.resolve(\_\_dirname, 'app'),

loader: 'raw-loader'

},

{

test: /\.(eot|svg|ttf|woff|woff2)$/,

loader: 'file-loader?name=public/fonts/[name].[ext]'

}

]

},

plugins:[

new webpack.optimize.CommonsChunkPlugin({

name: ['app', 'vendor', 'polyfills']

}),

new HtmlWebpackPlugin({

template: './index.html'

}),

new webpack.ContextReplacementPlugin(

/angular(\\|\/)core/,

path.resolve(\_\_dirname)

),

new webpack.ProvidePlugin({

$: 'jquery',

jQuery: 'jquery',

"window.jQuery": 'jquery'

})

]

};

1. Finally, start the build of your application by running npm start command on your command prompt and browse the application by using http://localhost:3000