Creating Angular 6 Application from scratch using Webpack 4

1. Open Command Prompt and make a new project directory and cd into it:

   mkdir hello_angular

   cd hello_angular

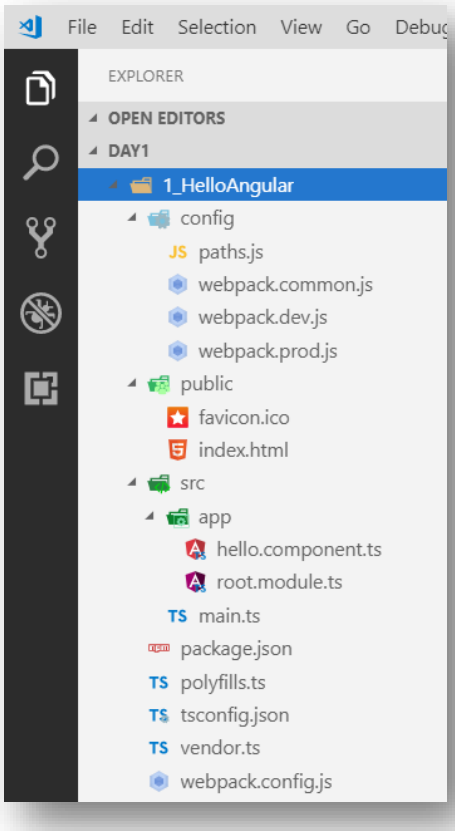
2. Create a package.json file:

   npm init -y

3. Open package.json file and copy paste the below contents in it:

```
{
  "name": "hello_angular",
  "version": "1.0.0",
  "description": "Hello World in Angular",
  "private": true,
  "scripts": {
    "start": "webpack-dev-server --hot --env dev",
    "build": "webpack --env prod"
  },
  "author": "Manish Sharma",
  "license": "ISC"
}
```

4. Open Folder in Visual Studio Code and create the files and folder as per below screen:



5. After the Files & Folders are created, we must do package installations using the following commands:

Production Dependencies

npm install --save rxjs zone.js @angular/core@6 @angular/common@6 @angular/compiler@6 @angular/platform-browser@6 @angular/platform-browser-dynamic@6 core-js

Development Dependencies

npm install --save-dev webpack webpack-cli webpack-dev-server webpack-merge

npm install --save-dev typescript@2.7

npm install --save-dev html-loader file-loader angular2-template-loader awesome-typescript-loader

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

npm install --save-dev @types/node

We can also install, all the dev dependencies using one command as follows:

npm install --save-dev webpack webpack-cli webpack-dev-server webpack-merge typescript@2.7 html-loader file-loader angular2-template-loader awesome-typescript-loader clean-webpack-plugin html-webpack-plugin terser-webpack-plugin @types/node

NOTE: After all the installation is done, please remove exponential (^) symbol which comes in the package.json file

Understanding of packages for development

webpack	webpack is a static module bundler for modern JavaScript applications. When webpack processes your application, it internally builds a dependency graph which maps every module your project needs and generates one or more bundles.
webpack-cli	Webpack's Command Line Interface
webpack-dev-server	Serves a webpack app. Updates the browser on changes.
webpack-merge	webpack-merge provides a merge function that concatenates arrays and merges objects creating a new object. If functions are encountered, it will execute them, run the results through the algorithm, and then wrap the returned values within a function again. This behaviour is particularly useful in configuring webpack although it has uses beyond it. Whenever you need to merge configuration objects, webpack-merge can come in handy.
typescript	TypeScript is a language for application scale JavaScript development
angular2-template-loader	Angular2 webpack loader that inlines your angular2 templates and stylesheets into angular components.
awesome-typescript-loader	TypeScript loader for Webpack
file-loader	A file loader module for webpack
html-loader	Exports HTML as string. HTML is minimized when the compiler demands.
clean-webpack-plugin	A webpack plugin to remove your build folder(s) before building
html-webpack-plugin	The HtmlWebpackPlugin simplifies creation of HTML files to serve your webpack bundles. This is especially useful for webpack bundles that include a hash in the filename which changes every compilation.

	You can either let the plugin generate an HTML file for you, supply your own template using lodash templates, or use your own loader.
terser-webpack-plugin	This plugin uses terser to minify your JavaScript. Terser is JavaScript parser, mangler, optimizer and beautifier toolkit for ES6+
@types/node	This package contains type definitions for Node.js

6. Open and copy paste the following code in each file, we have created earlier:

config/paths.js

```
const path = require('path');
const fs = require('fs');

const appDirectory = fs.realpathSync(process.cwd());

const resolvePath = function(relativePath) {
    return path.resolve(appDirectory, relativePath);
}

module.exports = {
    appRootPath: appDirectory,
    appBuildPath: resolvePath('dist')
}
```

config/webpack.common.js

```
const webpack = require('webpack');
const HtmlWebpackPlugin = require('html-webpack-plugin');
const path = require('path');

module.exports = function (env) {
    return {
        entry: {
            'polyfills': './polyfills.ts',
            'vendor': './vendor.ts',
            'app': './src/main.ts',
        },

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

        module: {
            rules: [
                {
                    test: /\.ts$/,
                    loaders: [
                        {
                            loader: 'awesome-typescript-loader',
                            options: { configFileName: './tsconfig.json' }
                        }, 'angular2-template-loader'
```

```
                ]
            },
            {
                test: /\.html$/,
                use: [
                    {
                        loader: "html-loader",
                        options: {
                            minimize: true,
                            caseSensitive: true,
                            removeAttributeQuotes: false
                        }
                    }
                ]
            },
            {
                test: /[\/\\]@angular[\/\\]core[\/\\].+\.js$/,
                parser: { system: true }
            },
            {
                test: /\.(ico)$/,
                use: "file-loader?name=[name].[ext]"
            }
        ]
    },

    plugins: [
        new HtmlWebpackPlugin({
            template: "./public/index.html",    // Input FileName
            filename: "./index.html"    // Output FileName
        }),
        new webpack.ContextReplacementPlugin(
            /angular(\\|\/)core/,
            path.resolve(__dirname)
        )
    ],

    optimization: {
        splitChunks: {
            chunks: "all"
        }
    }
};
}
```

config/webpack.dev.js

```
const webpackMerge = require('webpack-merge');
const commonConfig = require('./webpack.common.js');

module.exports = function(env){
    return webpackMerge(commonConfig(env), {
        mode: 'development',
```

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

        output: {
            publicPath: 'http://localhost:3000/',
            filename: '[name].js',
            chunkFilename: '[id].chunk.js'
        },

        devServer: {
            inline: true,
            port: 3000,
            historyApiFallback: true,
            clientLogLevel: 'none',
            stats: 'minimal',
            open: 'Chrome'
        }
    });
}
```

config/webpack.prod.js

```
const webpackMerge = require('webpack-merge');
const CleanWebpackPlugin = require('clean-webpack-plugin');
const TerserPlugin = require('terser-webpack-plugin');

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

const paths = require('./paths');

module.exports = function (env) {
    return webpackMerge(commonConfig(env), {
        mode: 'production',

        output: {
            path: paths.appBuildPath,
            publicPath: './',
            filename: 'static/js/[name].[chunkhash:8].js',
            chunkFilename: 'static/js/[name].[chunkhash:8].chunk.js'
        },

        plugins: [
            new CleanWebpackPlugin(['dist'], { root: paths.appRootPath })
        ],

        optimization: {
            minimizer: [
                new TerserPlugin({
                    terserOptions: {
                        parse: {
                            ecma: 8,
```

```
                        },
                        compress: {
                            ecma: 5,
                            warnings: false,
                            comparisons: false,
                            inline: 2,
                        },
                        mangle: {
                            safari10: true,
                        },
                        output: {
                            ecma: 5,
                            comments: false,
                            ascii_only: true,
                        },
                    },
                    parallel: true,
                    cache: true,
                    sourceMap: false,
                })
            ]
        }
    });
}
```

public/index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <link rel="shortcut icon" href="./favicon.ico">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <meta http-equiv="X-UA-Compatible" content="ie=edge">
    <title>Angular 6 Demos</title>
</head>

<body>
    <hello>Loading Hello Component...</hello>
</body>

</html>
```

vendor.ts

```
import "@angular/core";
import "@angular/common";
import "@angular/compiler";
import "@angular/platform-browser";
import "@angular/platform-browser-dynamic";

import 'rxjs';
```

polyfills.ts

```
import "core-js/es6";
import "core-js/es7/reflect";
import 'zone.js/dist/zone';
```

webpack.config.js

```
module.exports = function (env) {
    return require(`./config/webpack.${env}.js`)(env);
}
```

Let's create an angular component and render it on the HTML page

1. Open src/app/hello.component.ts and write the following code.

```
import { Component } from "@angular/core";

@Component({
    selector: 'hello',
    template: `<h2>Hello World!</h2>`
})
export class HelloComponent { }
```

2. Open src/app/root.module.ts and write the following code.

```
import { NgModule } from "@angular/core";
import { BrowserModule } from "@angular/platform-browser";
import { HelloComponent } from "./hello.component";

@NgModule({
    imports: [BrowserModule],
    declarations: [HelloComponent],
    bootstrap: [HelloComponent]
})
export class RootModule { }
```

3. Open src/main.ts

```
import { platformBrowserDynamic } from "@angular/platform-browser-dynamic";
import { RootModule } from "./app/root.module";

platformBrowserDynamic().bootstrapModule(RootModule);
```

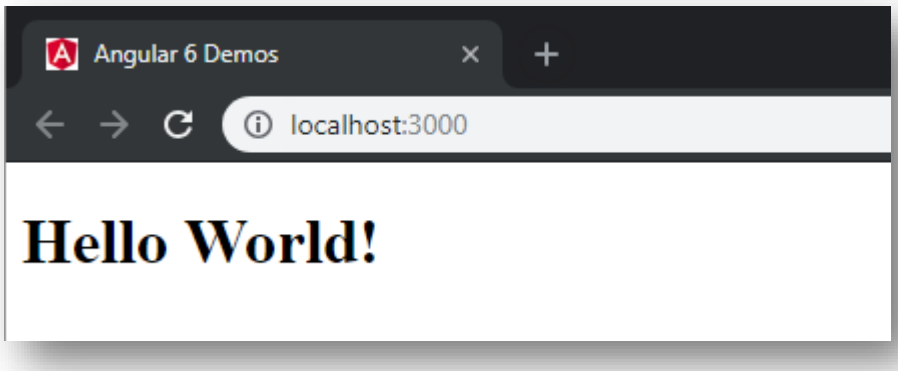
Let's run the application:

Open Command Prompt or terminal window on the folder 'hello_angular' and type **npm start,** you will se the following screen is shown, which shows compiled successfully.

```
> webpack-dev-server --hot --env dev

i ｢wds｣: Project is running at http://localhost:3000/
i ｢wds｣: webpack output is served from http://localhost:3000/
i ｢wds｣: 404s will fallback to /index.html
i ｢wdm｣: wait until bundle finished: /
i ｢atl｣: Using typescript@2.7.2 from typescript
i ｢atl｣: Using tsconfig.json from C:/Users/Manish Sharma/Desktop/Reskilling-Dec/Angular/Day1/hel
lo_angular/tsconfig.json
i ｢atl｣: Checking started in a separate process...
i ｢atl｣: Time: 970ms
i ｢wdm｣:    510 modules
i ｢wdm｣: Compiled successfully.
```

Chrome Browser will open and automatically navigate to localhost:3000, you will see the following output



Production Build:

Open Command Prompt on the folder 'hello_angular' and type **npm run build,** after webpack processing is complete, you must see a folder named **dist,** which is the final deployment unit.

If you want to locally deploy and test the dist folder outputcls:

1. npm install -g http-server (You have to do it only once)
2. cd in the dist folder and run the following command

   cd dist

   http-server .

```
Starting up http-server, serving .
Available on:
  http://172.21.77.1:8080
  http://192.168.15.216:8080
  http://127.0.0.1:8080
Hit CTRL-C to stop the server
```

3. Open browser and access "http://localhost:8080"