NG-BUILD

Here we are going to discuss about ng-build, this makes us to build the compressed files for all the files present. Here we can expect the final output as a library or application. If we want a library the configuration, ts-config, watch options are applied.

In ng-build we have 2 modes

* Development Mode
* Production Mode

Before entering into the modes we should include one topic about source maps which plays an important role in the build of a project.

SourceMap:

Source map maintains details of the compressed files, where the running of the program was easy.

We can manually maintain the source file by cli command with an extension of –sm, it takes one parameter which is of type bool. True value makes the source map to include in the destination folder.

Development Mode:

The default mode for ng-build is development mode and this mode creates a source map for every file of java script and css file.

Eg : ng build

If you are not interested to include the source maps in the destination folder use –sm with false as a value to it.

Eg : ng build -- dev - sm false

Production Mode:

Here we should explicitly run this mode by passing –prod to command.

In production mode we don’t have source maps in our destination folder, so for further execution we must pass the -- sm value as true.

In production mode the compressed files are of the types from which they are compressed. i.e., the .css files are compressed to .css. But in development mode there is no particular constraint on files so all the files are converted to .js format.

To convert the files to the same format after compressed also in development mode use the command – ec with true value.

Eg: ng build -- dev - ec true

In development mode we are not expecting any other plugin methods to be involved, which are done by production mode.

In production mode we can reduce the final bundle.js file by following methods.

-> Minification & Uglification

-> Tree Shaking

-> Uglification

Minification:

Removing of excess whitespaces, comments and optional tokens like curly braces and semicolons.

Uglification:

After transforming code it uses the short variables and function names.

TreeShaking:

Remove the unused code from the final bundle. It plays an important role in the loading of the project.

AOT Compilation:

In this the angular components are pre-compiled .

WEBPACK

Webpack is used for bundling process of all the files into a single file at the destination folder. Here all files are loaded asynchronously at run time.

Here the compression of the files is done but the minifying and other techniques are performed on the project.

To make the use of those methods we use – p at the end of the command.

Eg: ng webpack ‘src/js/app.js’ ‘dist/bundle.js’

The project will run in our local system where the file gets loaded and executed in our machine.

Here we have a extra package called webpack-dev-server for running our project on a sever and see the results. In this package we must include the command with some options in it.

Eg: ng webpack -- entry ‘./src/js/app.js’ -- output – filename ‘./dist/bundle.js’

The things we must consider when we learn webpack are:

->Entry

->Output

->Module Loader

->Plugins

Entry:

Here we include the path of the program, where we want to start.

Output:

Here the values are passed as an object with two parameters like path, filename and publicpath.

In path we use path package and a resolve method to load the bundle at the desired place dynamically.

Module:

Here we can load the module we need by mentioning the rules, where we specify the type of file and type of loader to load on the file.

Plugins:

Here the extra info is passed for custom modification to the final file. If plugins are not used then the minification and other methods are done with regular fashion.

Synatx:

Module.exports = {

Entry : ‘./src/js/app.js’,

Output : {

Path : path.resolve(\_\_dirname,’dist’),

Filename:’bundle.js’,

Publicpath:’./dist’

},

Module:{

Rules:[

{

Test:/\.css$/,

Loader:’css-loader

}

]

},

Plugins:[

//Enter the custom plugins needed.

]

}

Run Angular on Node Server:

For running of angular project on node server follow the 3 steps below.

->After creating the complete angular project, build the project using ng-build, as discussed earlier the compressed files are created at dist folder of root level. Here all files are converted to .js format.

Eg : ng build

->Now copy the compressed folder to the node project created, here there is no need to write extra logic about the program.one middleware should be included in the node program.

app.use(express.static('formSibling'));

Here the project name should be added in the place of ‘formSibling’.

->Now open the sever and OMG you can see your angular project running on node server.