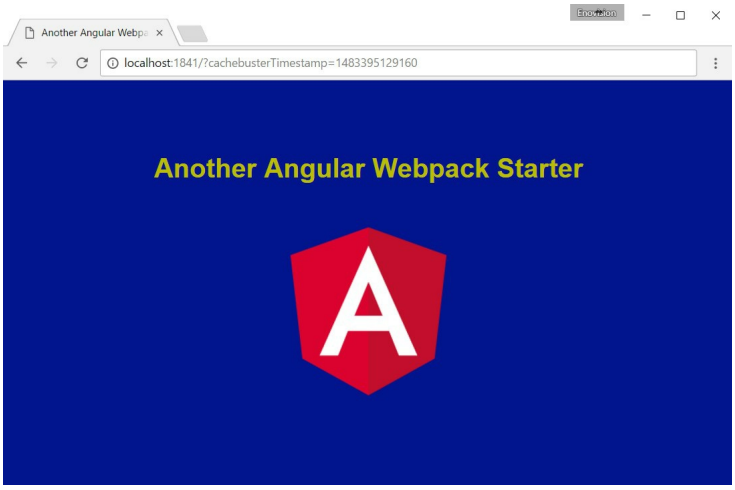
Another Angular Webpack Starter



This Angular 2 Webpack based starter has been build upon the official documentation from Angular.io.

See: https://angular.io/docs/ts/latest/guide/webpack.html

In this package I have included:

- SASS
- LESS
- Bootstrap

Please see contributing guidelines before reporting an issue.

Table of Contents

Folder structure

```
another-angular-webpack-starter/
+-- package.json
+-- package.copy.json
+-- tsconfig.json
+-- karma.conf.json
+-- webpack.config.json
+-- assets
    +-- css
    +-- images
    +-- js
    +-- sass
    +-- vendor
+-- build
    +-- development
    +-- production
+-- config
    +-- karma
    +-- lite-server
    +-- webpack
    +-- helpers.js
+-- node_modules
+-- src
    +-- app
    +-- app.ts
    +-- index.html
    +-- polyfills.ts
    +-- vendor.ts
```

root folder files

> package.json

Contains the dependencies for this package

> package.copy.json

Just a copy of package.json. In case you messed up the original one.

> tsconfig.json

Settings for the typescript handling/transpiler

> webpack.config.js

Imports the webpack.config.js in the folder **/config/webpack**

> karma.conf.js

Imports the karma.config.js in the folder **/config/karma**

assets

In this folder you can place your assets like
images, css, sass, other javascript and 'vendor' plugins or libs

build

In this folder the deliveries will be placed that come from:

production:

```
npm run build
```

development:

```
npm run build development
```

###config
In this folder are the configs for Karma, Webpack and the lite-server.
It has been placed into folders that fitt to their purpose.

The helpers.js file contains the function **root** to get the root folder
and is used in the other configs.

node_modules

As it says: node modules (npm packages)
This folder will be generated after you have executed the **npm install** commands
after cloning this package in your application folder.

src

The most important folder of all. This folder contains the source code of your
application.

Quick start

Clone/Download the repo then edit app.ts inside /src/app/app.ts

```
# clone our repo
$ git clone https://github.com/enovision/another-angular-webpack-starter.git your-app

# change directory to your app
$ cd your-app

# install the dependencies with npm
$ npm install

# start the server
$ npm start
```

go to http://localhost:1841 in your browser.

Guidelines

app.ts (in root folder)

The **main.ts** that you see often as the app starter has been replaced with **app.ts**.

If you prefer to have **main.ts** instead of **app.ts** then you can modify the **webpack.common.js**
file in the **config/webpack** folder (see below).

```
module.exports = {
  entry: {
    'polyfills': './src/polyfills.ts',
    'vendor': './src/vendor.ts',
    'app': './src/app.ts'   /* <-- change this */
  },
```

```
  ...
```

How to make SASS/LESS work

The instruction below describes the procedure based on **scss** (SASS) files, but the procedure is the same for **less** (LESS) files.

Within the **app** folder you will find the file **app.component.scss**. This file contains the import of the **scss** files in your assets folder.

```
@import "../../assets/sass/styles.scss";
@import "~bootstrap-sass/assets/stylesheets/bootstrap";
```

This file can be used as a dependency in your components (read further).

Embedding SASS/LESS in components

```
import { Component, ViewEncapsulation } from '@angular/core';

@Component({
  selector: 'my-app',
  templateUrl: './app.component.html',
  encapsulation: ViewEncapsulation.None,
  styleUrls: [
    './app.component.less', /* <-- LESS */
    './app.component.scss', /* <-- SASS */
    './app.component.css'   /* <-- CSS */
  ]
})
export class AppComponent { }
```

You include the **app.component.scss**, **app.component.less** or **app.component.css** in your **styleUrls** object of your component. The principle for CSS is the same as for SASS files. When using **npm start** or when creating the build with **npm run build**, the SASS content will be dynamically replaced with the corresponding CSS.

ViewEncapsulation

Notice the **ViewEncapsulation** dependency that has been set to **false**. This is important if you have styles in your SASS on the **body** HTML tag. If **ViewEncapsulation** is not set to **false** by leaving out or setting it to **true** it will not apply the **body** styled CSS because Angular adds an extension to the CSS styles to make them unique for the application.

index.html and bootstrapping the app module

In the **index.html** you won't find any references to **css** or **javascript** files. In fact the **index.html** is very clean. You can however put additional links to css or javascript files, for it is just a normal HTML file.

In the **app.module.ts** file the **app.component.ts** is bootstrapped in a **@NgModule**. So the components that you are using in your application are more or less bound in this **app.module**.

The module itself is referenced in **app.ts** (src/app folder) at the line **platformBrowserDynamic().bootstrapModule(AppModule);**.

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
```

```
@NgModule({
  imports: [
    BrowserModule
  ],
  declarations: [
    AppComponent
  ],
  bootstrap: [AppComponent] /* <-- binding to component */
})
export class AppModule { }
```

Installation, start, test and build

In the **package.json** file you'll find the **scripts** section. These are npm scripts that can be used during development, testing and building.

```
...
"scripts": {
    "start": "webpack-dev-server --inline --progress --port 1841",
    "test": "karma start",
    "build": "rimraf dist && webpack --config config/webpack/webpack.prod.js --progress --profile --bail",
    "build-development": "webpack --config config//webpack/webpack.development.js --progress --profile --
bail",
    "production": "lite-server -c config/lite-server/lite-server-config.production.js",
    "development": "lite-server -c config/lite-server/lite-server-config.development.js"
  },
  ...
```

All commands have to be started in a terminal (like the DOS prompt or Linux terminal) and from within the root folder of the application (the folder where this README.md file resides).

start

Starting the application. It will also start a internal web server (webpack-dev-server) and a watch on any changes. The port is set to **1841**, but you can change that in the script 'start' property within the **packages.json** file.

When you don't change anything before the npm install, then the browser should look like the image shown at the top of this README.

```
<prompt>:\npm start
```

test

Runs a Karma test. See **config\Karma\karma.config.js** for more details.

```
<prompt>:\npm test
```

build

Make a production build of the application. The result will be put in the folder **build\production**.

```
<prompt>:\npm build
```

build-development

Make a development build of the application. The result will be put in the folder **build\development**.

```
<prompt>:\npm run build development
```

production

This will start a lite-server with the production build as application. This makes it possible to see your production build in a test server to do some final analysis. It will also automatically start BrowserSync, what makes it possible to run the application at the same time on another device (like an iPad or Android) within your network[1].

```
<prompt>:\npm run production
```

development

This will start a lite-server with the development build as application. It will also automatically start BrowserSync, what makes it possible to run the application at the same time on another device (like an iPad or Android) within your network[1].

```
<prompt>:\npm run development
```

[1] BrowserSync

By default BrowserSync is using its own IP-address to start an internal webserver. This can be modified by changing the

- lite-server-config.production.js
- lite-server-config.development.js

file in the folder **config\lite-server**.

Add a "host" property with the ip address of your development PC/Laptop. You can also change the default port **8000** to a value of your preference.
(see below).

```
module.exports = {
  "injectChanges": false,
  "host": "xxx.xxx.xxx.xxx", /* ip address of your own PC/Laptop */
  "port": 8000,
```

Some usefull npm know-how's

The following node/npm commands are useful to know:

```
/* Check if newer versions of your used node modules exist */
ncu

/* Update the package.json with the updated node module versions */
ncu -u

/* Upgrade newer version node modules automatically */
ncu -a
/* or */
ncu --upgradeAll
```

Contributing

See Contributing