Steps to create ext-web-components-angular application using @angular/cli:

1. Install @angular/cli by running the following command *npm install -g @angular/cli* and create the repository by using *ng new ext-web-components-angular-app* command.

2. Use node version 10.x.x by using *nvm use 10.x.x* command as it is compatible with angular 7.

3. Run the following command : *cd ext-web-components-angular-app*

4. Change the *start* key of *scripts* object in *package.json* file to

   "start": "webpack-dev-server --env.browser=true --env.emit=yes --env.verbose=no",

5. Add following packages to *dependencies* in package.json file.
   "@types/jest": "^24.0.15",
   "protractor": "^5.4.2",
   "rxjs": "^6.5.2"

6. To consume ext-web-components, we need to add following packages to package.json and run *npm install* to install these dependencies in our project.
   "devDependencies": {
   "@sencha/ext-web-components": "~7.0.0",
   "@sencha/ext": "~6.7.0",
   "@sencha/ext-modern": "~6.7.0",
   "@sencha/ext-modern-treegrid": "~6.7.0",
   "@sencha/ext-calendar": "~6.7.0",
   "@sencha/ext-charts": "~6.7.0",
   "@sencha/ext-d3": "~6.7.0",
   "@sencha/ext-exporter": "~6.7.0",
   "@sencha/ext-pivot": "~6.7.0",
   "@sencha/ext-pivot-d3": "~6.7.0",
   "@sencha/ext-ux": "~6.7.0",

```
    "@sencha/ext-modern-theme-material": "~6.7.0"
  },
```

7. Add following files to your d*evDependencies* in *package.json* file and run *npm install* to install webpack into your project.

```
"webpack": "^4.29.6",
"webpack-cli": "^3.2.3",
"webpack-dev-server": "^3.2.1",
"webpack-filter-warnings-plugin": "^1.2.1",
"@webcomponents/webcomponentsjs": "^2.2.10",
```

8. Add following to *devDependencies* in *package.json* file and run *npm install*.

```
"core-js": "^2.6.5",
"zone.js": "^0.8.29",
"typescript": "3.2.4",
"@types/core-js": "^2.5.0",
"@types/node": "^11.10.4",
"rimraf": "^2.6.3",
"cross-env": "^5.2.0",
"@ngtools/webpack": "^7.3.4",
"portfinder": "^1.0.20",
"css-loader": "^2.1.0",
"file-loader": "^3.0.1",
"html-loader": "^0.5.5",
"node-sass": "^4.11.0",
"raw-loader": "^1.0.0",
"sass-loader": "^7.1.0",
"style-loader": "^0.23.1",
```

This below plugin will generate an HTML5 file that will include all webpack bundles in the body in script tag.

"html-webpack-plugin": "^3.2.0",

This is an extension for html-webpack-plugin to programmatically insert or update <base href=" " />.It requires html-webpack-plugin as a dependency.

"base-href-webpack-plugin": "^2.0.0",

This plugin is used to watch file changes during developments of sencha extjs applications.

"@sencha/ext-webpack-plugin": "~7.0.0",

This plugin copies individual files or entire directories to the build directory.

"copy-webpack-plugin": "^5.0.0",

9. Add following command to polyfills.ts if not there :-
   import 'core-js/es7/reflect';
   Import 'zone.js/dist/zone';

10. Create a file named *webpack.config.js* in the root of your project to add webpack configuration which is used to bundle your application.

11. Add the following code snippet in your *webpack.config.js* file to use the above mentioned plugins into your project along with the Ext-Webpack-Plugin .

```
const path = require('path');
const AngularCompilerPlugin = require('@ngtools/webpack').AngularCompilerPlugin;
const HtmlWebpackPlugin = require('html-webpack-plugin');
const { BaseHrefWebpackPlugin } = require('base-href-webpack-plugin');
const ExtWebpackPlugin = require('@sencha/ext-webpack-plugin');
const CopyWebpackPlugin = require('copy-webpack-plugin');
const portfinder = require('portfinder');
const webpack = require('webpack');
const FilterWarningsPlugin = require('webpack-filter-warnings-plugin');
```

```
module.exports = function(env) {
  function get(it, val) {if(env == undefined) {return val;} else if(env[it] == undefined) {return val;} else {return env[it];}}
  var profile = get('profile', '');
  var emit = get('emit', 'yes');
  var environment = get('environment', 'development');
  var treeshake = get('treeshake', 'no');
  var browser = get('browser', 'yes');
  var watch = get('watch', 'yes');
  var verbose = get('verbose', 'no');
  var basehref = get('basehref', '/');
  // var build_v = get('build_v', '7.0.0.0');

  const isProd = environment === 'production';
  const outputFolder = 'build';
  portfinder.basePort = (env && env.port) || 1962;

  const resolve = {
    extensions: ['.ts', '.js', '.html']
  };

  return portfinder.getPortPromise().then(port => {
    const plugins = [
      new AngularCompilerPlugin({
        tsConfigPath: './tsconfig.json',
        mainPath: './src/main.ts',
        skipCodeGeneration: true
      }),
      new HtmlWebpackPlugin({template: 'index.html', hash: true, inject: 'body'}),
      new BaseHrefWebpackPlugin({ baseHref: basehref }),
      new ExtWebpackPlugin({
        framework: 'webcomponents',
```

```
            toolkit: 'modern',
            theme: 'theme-material',
            emit: emit,
            script: '',
            port: port,
            packages: [],
            profile: profile,
            environment: environment,
            treeshake: treeshake,
            browser: browser,
            watch: watch,
            verbose: verbose
        }),
         new CopyWebpackPlugin([{
            from:
'../node_modules/@webcomponents/webcomponentsjs/webcomponents-bundle.js',
            to: './webcomponents-bundle.js'
        }]),
        new webpack.ContextReplacementPlugin(
            /\@angular(\\|\/)core(\\|\/)fesm5/,
            path.resolve(__dirname, 'src'), {}
        ),
        new FilterWarningsPlugin({
            exclude: /System.import/
        }),
        new webpack.HotModuleReplacementPlugin()
    ];
    return {
        mode: environment,
        devtool: (environment === 'development') ? 'inline-source-map' : false,
        context: path.join(__dirname, './src'),
        entry: {
            polyfills: './polyfills.ts',
```

```
            main: './main.ts'
        },
        output: {
            path: path.join(__dirname, outputFolder),
            filename: '[name].js'
        },
        plugins: plugins,
        resolve: resolve,
        module: {
          rules: [
            { test: /\.(png|svg|jpg|jpeg|gif)$/, use: ['file-loader'] },
            { test: /\.ts$/, loader: '@ngtools/webpack' },
            { test: /\.(html)$/, use: { loader: 'html-loader' } },
            { test: /\.(css|scss)$/,use: [{ loader: 'style-loader' },{ loader: 'css-loader' },{
loader: 'sass-loader' }]}]
        },

        performance: { hints: false },
        stats: 'none',
        optimization: { noEmitOnErrors: true },
        node: false,
        devServer: {
          contentBase: outputFolder,
          hot: !isProd,
          historyApiFallback: true,
          host: '0.0.0.0',
          port: port,
          disableHostCheck: false,
          compress: isProd,
          inline:!isProd,
          stats: 'none'
        }
      };
```

```
  });
};
```

12. Add the following files to your *devDependencies* in *package.json* file and run *npm install* to install Babel and its plugins

```
"@babel/core": "^7.3.4",
"@babel/plugin-proposal-class-properties": "^7.3.4",
"@babel/plugin-proposal-decorators": "^7.3.0",
"@babel/plugin-proposal-export-namespace-from": "^7.2.0",
"@babel/plugin-proposal-function-sent": "^7.2.0",
"@babel/plugin-proposal-json-strings": "^7.2.0",
"@babel/plugin-proposal-numeric-separator": "^7.2.0",
"@babel/plugin-proposal-throw-expressions": "^7.2.0",
"@babel/plugin-syntax-dynamic-import": "^7.2.0",
"@babel/plugin-syntax-import-meta": "^7.2.0",
"@babel/plugin-transform-runtime": "^7.3.4",
"@babel/preset-env": "^7.3.4",
"@babel/runtime": "^7.3.4"
```

13. Create a new file named *.babelrc* in the root of your project to add babel configuration which is mainly used to convert ES6+ code into backwards compatible version of Javascript in old browsers or environments.

14. Add the following code snippet to your *.babelrc* file to include the plugins installed.

```
{
  "presets": [
    [
      "@babel/preset-env",
      {
        "modules": false
      }
    ]
```

```
],
"plugins": [
  "@babel/plugin-transform-runtime",
  "@babel/plugin-syntax-dynamic-import",
  "@babel/plugin-syntax-import-meta",
  "@babel/plugin-proposal-class-properties",
  "@babel/plugin-proposal-json-strings",
  [
    "@babel/plugin-proposal-decorators",
    {
      "legacy": true
    }
  ],
  "@babel/plugin-proposal-function-sent",
  "@babel/plugin-proposal-export-namespace-from",
  "@babel/plugin-proposal-numeric-separator",
  "@babel/plugin-proposal-throw-expressions"
],
"ignore": [
  "build"
],
"env": {
  "test": {
    "presets": [
      "@babel/preset-env"
    ],
    "plugins": [
      "@babel/plugin-syntax-dynamic-import",
      "@babel/plugin-syntax-import-meta",
      "@babel/plugin-proposal-class-properties",
      "@babel/plugin-proposal-json-strings",
      [
        "@babel/plugin-proposal-decorators",
```

```
          {
            "legacy": true
          }
        ],
        "@babel/plugin-proposal-function-sent",
        "@babel/plugin-proposal-export-namespace-from",
        "@babel/plugin-proposal-numeric-separator",
        "@babel/plugin-proposal-throw-expressions"
      ]
    }
  }
}
```

15. Replace *index.html content with below code* in the *src* folder

```
<!doctype html>
<html lang="en">
<head>
 <meta charset="utf-8">
<script src="webcomponents-bundle.js"></script>
 <title>ExtWebComponentsAngularApp</title>
 <base href="/">

 <meta name="viewport" content="width=device-width, initial-scale=1">
 <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
 <app-root></app-root>
</body>
</html>
```

16. Import custom elements schema by using following command inside app.module.ts :
    `import {CUSTOM_ELEMENTS_SCHEMA} from '@angular/core';`
    and add attribute `schemas: [ CUSTOM_ELEMENTS_SCHEMA ],` into @NgModule

17. Import ext-web-components by using following command inside app.module.ts to consume custom ext-web-components :
    import '@sencha/ext-web-components/dist/ext-panel.component';

18. Replace styleUrls: ['./app.component.css'] with styles: [``] as we are not using any css inside app.component.ts.

19. Ext-web-components can be now inserted into app.component.html file.
    Example: Add the following code snippet in app.component.html file:-

```
<ext-panel
    height="100%"
    shadow="true"
    bodyPadding="20px"
  >
    <h1>About this App</h1>
    <p>
      Lorem ipsum dolor sit amet, consectetur adipiscing elit.
      Etiam eget leo sed mi imperdiet dictum a id turpis.
      Suspendisse a ante eu lorem lacinia vestibulum.
      Suspendisse volutpat malesuada ante, sed fermentum massa auctor in.
      Praesent semper sodales feugiat.
      Class aptent taciti sociosqu ad litora torquent per conubia nostra,
      per inceptos himenaeos.
      Mauris mauris ante, suscipit id metus non, venenatis tempor tortor.
      In ornare tempor ipsum.
      Sed rhoncus augue urna, ut dapibus odio fringilla vitae.
      Phasellus malesuada mauris ut nulla varius sodales.
      Sed et leo luctus, venenatis felis sit amet, vehicula nibh.
      Curabitur fringill a fringilla nibh, porttitor lacinia urna vestibulum eu.
      Integer ac aliquet risus. Curabitur imperdiet quis purus at consectetur.
            Sed ornare vitae felis a scelerisque.
```

```
        Donec mi purus, auctor sit amet molestie nec, imperdiet auctor mauris.
        </p>
    </ext-panel>
```

20. Now run *npm start* in the root of your directory.

If you don’t want to follow the above mentioned steps, then you can consume the boilerplate code ext-web-components-boilerplate-angular and start creating your application by consuming ext-web-components directly.