**CREATING LIBRARIES IN ANGULAR**

To create libraries from Angular v6 and above, Angular CLI has the support for it.

1. In a new or existing project, create a library using the command: `ng generate library <library-name>`. Angular CLI runs ng-packagr in the background to create and build libraries.
2. This will create a new folder inside your application called ‘projects’. This will contain another folder with the name of your library.
3. The different files generated will be:

* **tsconfig.lib** – contains compiler configuration information
* **tsconfig**.**spec** – Testing related file
* **tslint** – Linting information file
* **package.json** – contains the name, version of the libraries, and the peerdependencies
* **ngPackage** – contains the entry and destination information
* **karma.conf** – the testing framework
* **src folder** Contains a test file and a public api file. This file is very important since it exposes the components, services, modules that we create in the library to applications.

Contains a **Lib** folder which has the code for the component, module, service, directive etc.

1. Once you have your library ready with what it needs to perform, you can build your library by using ng build and the name of your library. `ng build <library-name>`. Alternatively, you can give a script in the package.json to build your library.

Example:     "build-myLib": "ng build myLib"

1. A new folder gets created called ‘dist’ which contains the folder of the name of your library and this folder will contain all the different bundles, ESM5, ESM2015, UMD, Flat esm5, flat esm2015. It will also contain a public\_api.d.ts file which exposes your components, a new package.json for your library now. And a lib folder which contains the d.ts files of your components, services, and modules.
2. After having built our library, we can use it inside our local application.

Outside the projects folder, you can see the src folder, which is your actual application. This is where we want to use the library!

1. Inside the app.module.ts, we import the component module from the library’s dist folder.

import {myLibModule} from '../../dist/my-Lib';

We will also include it inside the imports array inside NgModule

1. Having imported the module from the library, we can use the component and the service since the module contains that information.
2. To use it on our application and have a look and feel, we can use the selector of the component from the library on the template of our App component.

**USING THIS LIBRARY in a separate application locally**

1. There are three ways of doing that. Using:

* **npm install**
* **npm link**
* **npm pack**

1. **Using npm install**

In another application locally, if you want to use the same library, you can install it using npm install <library-path>

Example: **npm install C:\Users\NishuGoel\angularLib\dist\my-Lib**

This adds the library as a dependency inside the package.json of the application.

Like this:     "my-Lib": "file:../angularLib/dist/my-Lib",

1. An important thing to notice here is that since the library has some peer dependencies, you will see the installed library in the node\_modules has a special mark to it, which basically asks you to resolve the peerdependencies.   
   To do this, we simply add a flag inside the angular.json file inside build options:

**“preserveSymlinks”: true**

1. **Using npm link**

If you simply want to link the library to a specific folder, not really installing it inside the application. This is like using npm install but you don’t need to install the library every time there is a change since you are just creating a link with the library on its own path.

**To do this, delete the installed library from the node\_modules folder and delete the entries from the package.json too.**

Then similarly, use :

**npm link <path>**

Example: **npm link C:\Users\NishuGoel\angularLib\dist\my-Lib**

This will not update your package.json but just create a link to the specific folder.

1. **Using npm pack**

After removing all the old references to the library, lets look at the usage of the library as a pack. To do this, we move inside the build path of the library in the workspace where we created the library i.e. **dist/myLib and use npm pack.**

This creates a number of files with our major focus on the created tgz file which is like a pack of the library functionality. This pack can be given to any other local application now.

Going to the consuming app, we simply use :

npm install <path to the pack file>

Example: **npm install ../angularLib/dist/myLib/my-lib-0.0.1.tgz**

This will get added to the package.json as a zipped pack and we can then use it inside our application.

1. Also, now the flag “preserveSymlinks” can be removed since the tgz file acts as an actual published standalone library.

**Finally, Publishing the library Publicly!**

1. Login to npm is the first step. You can do that both from the website npmjs.com and the CLI suing npm login.
2. Define the version and name. To give versioning, the semver rules are followed. Major, minor, patch version. The version can be increased both manually as well as through the CLI using npm version patch/minor/major
3. Define the name starting with something like @your-name/lib-name to keep it unique. Do not forget to change the name of library inside the package.json
4. Publish using the –public flag

**npm publish –access public**

1. It will ask for an OTP and finally publish it to npm. Your library is now ready for use by anybody across the world.
2. To consume it in any application now, use:

**npm install @nishugoel/myLib**

And you are all good to go!