**Node**

Check Node version Command : npm –v

Update Node version Command : install npm@latest –g

**CLI**

Check CLI version Command : ng --version

Uninstall Command : npm uninstal -g @angular/cli

Clean Command : npm cache clean

Reinsatll CLI : npm install -g @angular/cli@latest

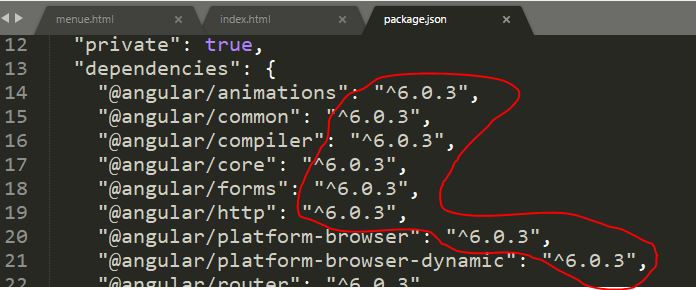
ng new ProjectName

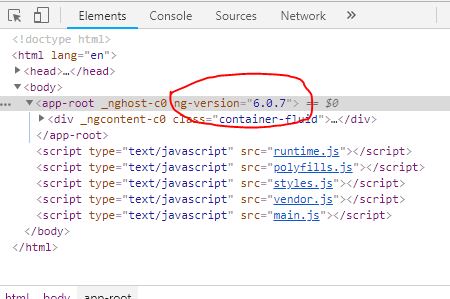
**Build**

ng build --prod --base-href ‘/es/’ (iis alias name)

ng serve –port 4203

**To know Angular version** of the application F12 / can see in pakage.json file





**Use the following commands to install PrimeNG packages in the project.**

1. npm install primeng --save
2. npm install primeicons --save
3. npm install --save font-awesome angular-font-awesome

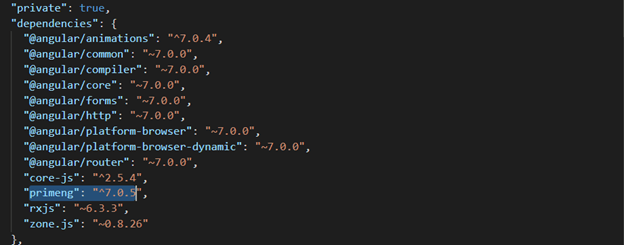
**Step 3**

Some PrimeNG components use Angular animations to improve the user interface, so we need to install Angular animation.

Install the Angular animation module in this project by using the following command.

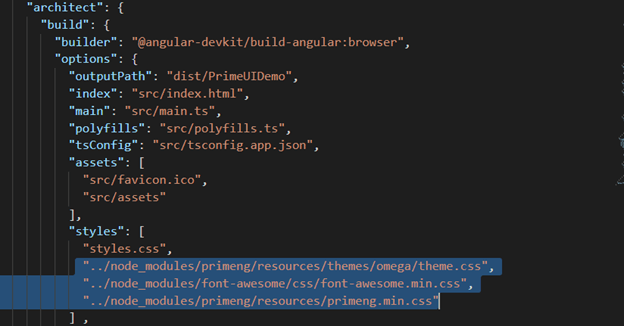
1. npm install --save @angular/animations

Now, open the package.json file and check if PrimeNG is installed.



**Step 4 - Style Configuration**

Configure required styles at the styles section in angular.json file or style.css.



"../node\_modules/font-awesome/css/font-awesome.min.css",

"../node\_modules/primeng/resources/themes/omega/theme.css",

"../node\_modules/primeng/resources/primeng.css",

**Step 5**

Configure PrimeNG module in the app.module.ts file. Let's import the required module in this file.

**App.module.ts fille**

1. import { BrowserModule } from '@angular/platform-browser';
2. import { NgModule } from '@angular/core';
3. import { BrowserAnimationsModule } from "@angular/platform-browser/animations";
4. import { AppRoutingModule } from './app-routing.module';
5. import { AppComponent } from './app.component';
6. import { AccordionModule } from 'primeng/components/accordion/accordion';
7. import {OrderListModule} from 'primeng/orderlist';
8. @NgModule({
9. declarations: [
10. AppComponent
11. ],
12. imports: [
13. BrowserModule,
14. AppRoutingModule,BrowserAnimationsModule,AccordionModule,OrderListModule
15. ],
16. providers: [],
17. bootstrap: [AppComponent]
18. })
19. export **class** AppModule { }

**Step 7**

Now, openapp.component.html and add PrimeNG accordion, a list. Add the following lines in this file.

1. <p-accordion>
2. <p-accordionTab header="India">
3. <p-orderList [value]="Indiastate">
4. <ng-template let-state pTemplate="item">
5. <div class="ui-helper-clearfix">
6. <div style="font-size:16px;float:left;margin:5px">{{state}}</div>
7. </div>
8. </ng-template>
9. </p-orderList>
10. </p-accordionTab>
11. <p-accordionTab header="Australia">
12. <p-orderList [value]="Ausstate">
13. <ng-template let-Ausstate pTemplate="item">
14. <div class="ui-helper-clearfix">
15. <div style="font-size:16px;float:left;margin:5px">{{Ausstate}}</div>
16. </div>
17. </ng-template>
18. </p-orderList>
19. </p-accordionTab>
20. <p-accordionTab header="SriLanka">
21. <p-orderList [value]="Slstate">
22. <ng-template let-Slstate pTemplate="item">
23. <div class="ui-helper-clearfix">
24. <div style="font-size:16px;float:center;margin:5px">{{Slstate}}</div>
25. </div>
26. </ng-template>
27. </p-orderList>
29. </p-accordionTab>
30. </p-accordion>

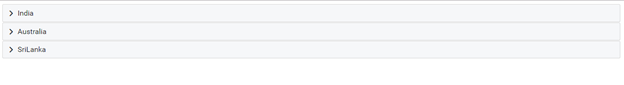
**Step 8**

Open app.component.ts file and add the following code.

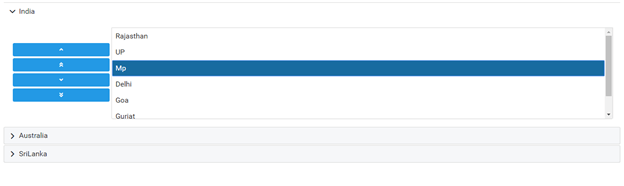
1. import { Component } from '@angular/core';
2. @Component({
3. selector: 'app-root',
4. templateUrl: './app.component.html',
5. styleUrls: ['./app.component.css']
6. })
7. export **class** AppComponent {
8. Indiastate = [
9. "Rajasthan",
10. "UP",
11. "Mp",
12. "Delhi",
13. "Goa",
14. "Gurjat",
15. "Punjab"
16. ];
17. Ausstate = [
19. "New South Wales",
20. "Queensland",
21. "South Australia",
22. "Tasmania"
23. ];
24. Slstate = [
25. "Kandy",
26. "Galle",
27. "Kegalle",
28. "Mannar"
29. ];
30. }

**Step 9**

Run the project and check the result.



 Click on the country name and check the list of states.



**Summary**

In this article, we learned the basics of PrimeNG, a free and open source library of UI components for Angular applications.

<https://www.primefaces.org/primeng/#/dialog>

<https://stackoverflow.com/questions/38846232/how-to-implement-signin-with-google-in-angular-2-using-typescript>

<https://github.com/abacritt/angularx-social-login>

<https://abacritt.github.io/angularx-social-login/>

4418552472-lggk1tbdk2o0e7a8fftn2khlaubum0m3.apps.googleusercontent.com

zuzwdEfEHJ9ETJ\_Z63qxFYuD

4418552472-lggk1tbdk2o0e7a8fftn2khlaubum0m3.apps.googleusercontent.com

<https://code-maze.com/authentication-aspnetcore-jwt-1/>

# Angular 7 Social Login

Social login and authentication module for Angular 7 (supports Angular 4+). Supports authentication with **Google** and **Facebook**. Can be extended to other providers also.

Check out the [demo](https://abacritt.github.io/angularx-social-login/).

## Getting started

### Install via npm

npm install --save angularx-social-login

### Import the module

In your AppModule, import the SocialLoginModule

import { SocialLoginModule, AuthServiceConfig } from "angularx-social-login";

import { GoogleLoginProvider, FacebookLoginProvider } from "angularx-social-login";

let config = new AuthServiceConfig([

{

id: GoogleLoginProvider.PROVIDER\_ID,

provider: new GoogleLoginProvider("Google-OAuth-Client-Id")

},

{

id: FacebookLoginProvider.PROVIDER\_ID,

provider: new FacebookLoginProvider("Facebook-App-Id")

}

]);

export function provideConfig() {

return config;

}

@NgModule({

declarations: [

...

],

imports: [

...

SocialLoginModule

],

providers: [

{

provide: AuthServiceConfig,

useFactory: provideConfig

}

],

bootstrap: [...]

})

export class AppModule { }

### Sign in and out users

import { AuthService } from "angularx-social-login";

import { FacebookLoginProvider, GoogleLoginProvider } from "angularx-social-login";

@Component({

selector: 'app-demo',

templateUrl: './demo.component.html',

styleUrls: ['./demo.component.css']

})

export class DemoComponent implements OnInit {

constructor(private authService: AuthService) { }

signInWithGoogle(): void {

this.authService.signIn(GoogleLoginProvider.PROVIDER\_ID);

}

signInWithFB(): void {

this.authService.signIn(FacebookLoginProvider.PROVIDER\_ID);

}

signOut(): void {

this.authService.signOut();

}

}

### Subscribe to the authentication state

You are notified when user logs in or logs out. You receive a SocialUser object when the user logs in and a null when the user logs out. SocialUser object contains basic user information such as name, email, photo URL, etc.

import { AuthService } from "angularx-social-login";

import { SocialUser } from "angularx-social-login";

@Component({

selector: 'app-demo',

templateUrl: './demo.component.html',

styleUrls: ['./demo.component.css']

})

export class DemoComponent implements OnInit {

private user: SocialUser;

private loggedIn: boolean;

constructor(private authService: AuthService) { }

ngOnInit() {

this.authService.authState.subscribe((user) => {

this.user = user;

this.loggedIn = (user != null);

});

}

}

### Display the user information

<img src="{{ user.photoUrl }}">

<div>

<h4>{{ user.name }}</h4>

<p>{{ user.email }}</p>

</div>

## Specifying custom scope

const fbLoginOptions: LoginOpt = {

scope: 'pages\_messaging,pages\_messaging\_subscriptions,email,pages\_show\_list,manage\_pages',

return\_scopes: true,

enable\_profile\_selector: true

}; // https://developers.facebook.com/docs/reference/javascript/FB.login/v2.11

const googleLoginOptions: LoginOpt = {

scope: 'profile email'

}; // https://developers.google.com/api-client-library/javascript/reference/referencedocs#gapiauth2clientconfig

let config = new AuthServiceConfig([

{

id: GoogleLoginProvider.PROVIDER\_ID,

provider: new GoogleLoginProvider("Google-OAuth-Client-Id", googleLoginOptions)

},

{

id: FacebookLoginProvider.PROVIDER\_ID,

provider: new FacebookLoginProvider("Facebook-App-Id", fbLoginOptions)

}

]);

## Building with AoT

If you are facing issue in building your app with AoT, check [this document](https://github.com/abacritt/angularx-social-login/blob/master/README-AOT.md).

## Running the demo app

cd demo

npm install

ng serve

Interceptor

ng g class token-interceptor

import { Injectable, Injector } from '@angular/core';

import {

HttpEvent,

HttpInterceptor,

HttpHandler,

HttpRequest

} from '@angular/common/http';

import { Observable } from 'rxjs/Rx';

import 'rxjs/add/observable/throw';

import 'rxjs/add/operator/catch';

export class TokenInterceptor implements HttpInterceptor {

publicAPI = 'api/connect/token';

constructor(public localStorage: LocalStorageService) { }

intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {

const token = this.oauthService.getAccessToken();

if (request.url.indexOf(this.publicAPI) === -1) {

request = request.clone({

setHeaders: {

Authorization: `Bearer ${token}`

}

});

}

return next.handle(request);

}

}

Added storage services

Added interceptor

Added

npm install @angular/cdk –save

## Lazing Loading

ng g module [module-name] –routing

"module": "esnext",

{ path: '', loadChildren: () => import('./pages/web/web.module').then(m => m.WebModule) },

## Json FileImport

"resolveJsonModule": true,

"esModuleInterop": true,

## Read Token

Use [**@auth0/angular-jwt**](https://github.com/auth0/angular2-jwt)

Step - 1 : Install using npm

npm install @auth0/angular-jwt

Step - 2 : Import the package

import { JwtHelperService } from '@auth0/angular-jwt';

Step - 3 : Create an instance and use

const helper = new JwtHelperService();

const decodedToken = helper.decodeToken(myRawToken);

// Other functions

const expirationDate = helper.getTokenExpirationDate(myRawToken);

const isExpired = helper.isTokenExpired(myRawToken);

## Host in iis

ng build --base-href "/Tricolor.web/"

ng build --base-href "/toh/" –prod

## 404 error on refresh

## Method — 1

In app.module.ts:

* Add imports:

import { HashLocationStrategy, LocationStrategy } from '[@angular/common](http://twitter.com/angular/common)';

* And in NgMoudle provider, add:

{provide: LocationStrategy, useClass: HashLocationStrategy}

## Method — 2

Use RouterModule.forRoot with the {useHash: true} argument.

**Example:**(from [angular docs](https://angular.io/docs/ts/latest/guide/router.html#!#browser-url-styles)) (app-routing.module.ts)

import { NgModule } from '@angular/core';  
...const routes: Routes = [//routes in here];@NgModule({  
 imports: [  
 BrowserModule,  
 FormsModule,  
 RouterModule.forRoot(routes, { useHash: true })  
 ],  
 bootstrap: [AppComponent]  
})  
export class AppModule { }

## PDF

* npm install jspdf jspdf-autotable

npm install xlsx

npm install file-saver –save

npm i @types/node

"scripts": [

"node\_modules/jspdf/dist/jspdf.es.min.js",

"node\_modules/jspdf-autotable/dist/jspdf.plugin.autotable.js"

]

The problem (as outlined in [typescript getting error TS2304: cannot find name ' require'](https://stackoverflow.com/questions/31173738/typescript-getting-error-ts2304-cannot-find-name-require)) is that the type definitions for node are not installed.

With a projected genned with @angular/cli 1.x, the specific steps should be:

**Step 1**:

Install @types/node with either of the following:

- npm install --save @types/node

- yarn add @types/node –D

**Step 2**: Edit your **src/tsconfig.app.json** file and add the following in place of the empty "types": [], which should already be there:

...

"types": [ "node" ],

"typeRoots": [ "../node\_modules/@types" ]

...

…………………………………………….

**Finally I got solution for this, check my App module file :**

import { AppComponent } from './app.component';

declare var require: any;

export function highchartsFactory() {

const hc = require('highcharts');

const dd = require('highcharts/modules/drilldown');

dd(hc);

return hc;

}

@NgModule({