|  |
| --- |
|  |
| app.module.ts  **import** { BrowserModule } **from** '@angular/platform-browser'; **import** { NgModule } **from** '@angular/core'; **import** { FormsModule } **from** '@angular/forms'; // <-- NgModel lives here  **import { AppComponent } from './app.component'; import { HeroesComponent } from './heroes/heroes.component';**  @NgModule({  declarations: [  AppComponent,  HeroesComponent  ],  imports: [  BrowserModule,  FormsModule  ],  providers: [],  bootstrap: [AppComponent] }) **export class** AppModule { }  heroes.component.ts  **import** { Component, OnInit } **from** '@angular/core'; **import** { Hero } **from** '../hero';  @Component({  selector: 'app-heroes',  templateUrl: './heroes.component.html',  styleUrls: ['./heroes.component.css'] }) **export class** HeroesComponent **implements** OnInit {  /\*  hero = 'Windstorm';  \*/  hero: Hero = {  id: 1,  name: 'Windstorm'  };   **constructor**() { }   ngOnInit() {  }  }  index.html쪽에서는 app-root를 보여주기때문에  <body>  <app-root></app-root> </body>  app.component.ts가 보여진다. main.ts->app.module.ts에의해 불러지기때문에. heroes가 import되어있다.  **import** { Component } **from** '@angular/core';  @Component({  selector: 'app-root',  templateUrl: './app.component.html',  styleUrls: ['./app.component.css'] }) **export class** AppComponent {  title = 'Tour of Heroes'; }  **import** { enableProdMode } **from** '@angular/core'; **import** { platformBrowserDynamic } **from** '@angular/platform-browser-dynamic';  **import** { AppModule } **from** './app/app.module'; **import** { environment } **from** './environments/environment';  **if** (environment.production) {  enableProdMode(); }  platformBrowserDynamic().bootstrapModule(AppModule);  app.component.html  <h1>{{title}}</h1> <app-heroes></app-heroes> |

**if , for import**

|  |  |
| --- | --- |
|  | mock-heroes.ts  **import** { Hero } **from** './hero';  **export const** HEROES: Hero[] = [  { id: 11, name: 'Mr. Nice' },  { id: 12, name: 'Narco' },  { id: 13, name: 'Bombasto' },  { id: 14, name: 'Celeritas' },  { id: 15, name: 'Magneta' },  { id: 16, name: 'RubberMan' },  { id: 17, name: 'Dynama' },  { id: 18, name: 'Dr IQ' },  { id: 19, name: 'Magma' },  { id: 20, name: 'Tornado' } ];  hero.ts  **export class** Hero {  id: **number**;  name: **string**; } |
| heroes.component.ts  **import** { Component, OnInit } **from** '@angular/core'; **import** { Hero } **from** '../hero'; **import** { HEROES } **from** '../mock-heroes';  @Component({  selector: 'app-heroes',  templateUrl: './heroes.component.html',  styleUrls: ['./heroes.component.css'] }) **export class** HeroesComponent **implements** OnInit {   heroes = HEROES;   selectedHero: Hero;    **constructor**() { }   ngOnInit() {  }   onSelect(hero: Hero): **void** {  **this**.selectedHero = hero;  } } | heroes.component.html  <h2>My Heroes</h2> <ul class="heroes">  <li \*ngFor="let hero of heroes"  [class.selected]="hero === selectedHero"  (click)="onSelect(hero)">  <span class="badge">{{hero.id}}</span> {{hero.name}}  </li> </ul>  <div \*ngIf="selectedHero">   <h2>{{ selectedHero.name | uppercase }} Details</h2>  <div><span>id: </span>{{selectedHero.id}}</div>  <div>  <label>name:  <input [(ngModel)]="selectedHero.name" placeholder="name">  </label>  </div>  </div> |

**Injection @Input**

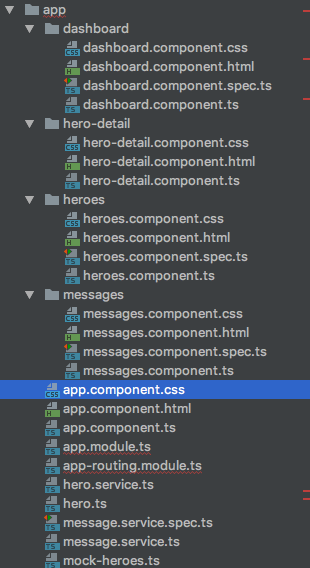
|  |  |
| --- | --- |
| hero.ts  **export class** Hero {  id: **number**;  name: **string**; } | mock-heros.ts  **import** { Hero } **from** './hero';  **export const** HEROES: Hero[] = [  { id: 11, name: 'Mr. Nice' },  { id: 12, name: 'Narco' },  { id: 13, name: 'Bombasto' },  { id: 14, name: 'Celeritas' },  { id: 15, name: 'Magneta' },  { id: 16, name: 'RubberMan' },  { id: 17, name: 'Dynama' },  { id: 18, name: 'Dr IQ' },  { id: 19, name: 'Magma' },  { id: 20, name: 'Tornado' } ]; |
| app.module.ts  **import** { BrowserModule } **from** '@angular/platform-browser'; **import** { NgModule } **from** '@angular/core'; **import** { FormsModule } **from** '@angular/forms';  **import** { AppComponent } **from** './app.component'; **import** { HeroesComponent } **from** './heroes/heroes.component'; **import** { HeroDetailComponent } **from** './hero-detail/hero-detail.component';  @NgModule({  declarations: [  AppComponent,  HeroesComponent,  HeroDetailComponent  ],  imports: [  BrowserModule,  FormsModule  ],  providers: [],  bootstrap: [AppComponent] }) **export class** AppModule { } | app.component.ts  **import** { Component } **from** '@angular/core';  @Component({  selector: 'app-root',  templateUrl: './app.component.html',  styleUrls: ['./app.component.css'] }) **export class** AppComponent {  title = 'Tour of Heroes'; } |
| hero.component.ts  **import** { Component, OnInit } **from** '@angular/core'; **import** { Hero } **from** '../hero'; **import** { HEROES } **from** '../mock-heroes';  @Component({  selector: 'app-heroes',  templateUrl: './heroes.component.html',  styleUrls: ['./heroes.component.css'] }) **export class** HeroesComponent **implements** OnInit {   heroes = HEROES;   selectedHero: Hero;   **constructor**() { }   ngOnInit() {  }   onSelect(hero: Hero): **void** {  **this**.selectedHero = hero;  } } | hero.component.html  <h2>My Heroes</h2>  <ul class="heroes">  <li \*ngFor="let hero of heroes"  [class.selected]="hero === selectedHero"  (click)="onSelect(hero)">  <span class="badge">{{hero.id}}</span> {{hero.name}}  </li> </ul>  <app-hero-detail **[hero]="selectedHero"**></app-hero-detail> |
| hero-detail.component.ts  **import** { Component, OnInit, Input } **from** '@angular/core'; **import** { Hero } **from** '../hero';  @Component({  selector: 'app-hero-detail',  templateUrl: './hero-detail.component.html',  styleUrls: ['./hero-detail.component.css'] }) **export class** HeroDetailComponent **implements** OnInit {  **@Input() hero: Hero;**   **constructor**() { }   ngOnInit() {  }  } | hero-detail.component.html  <div \*ngIf="hero">   <h2>{{ hero.name | uppercase }} Details</h2>  <div><span>id: </span>{{hero.id}}</div>  <div>  <label>name:  <input [(ngModel)]="hero.name" placeholder="name"/>  </label>  </div>  </div> |

Service

<http://han41858.tistory.com/33>

|  |  |
| --- | --- |
|  | App.module.ts  **import** { BrowserModule } **from** '@angular/platform-browser'; **import** { NgModule } **from** '@angular/core'; **import** { FormsModule } **from** '@angular/forms';  **import** { AppComponent } **from** './app.component'; **import** { HeroesComponent } **from** './heroes/heroes.component'; **import** { HeroDetailComponent } **from** './hero-detail/hero-detail.component'; **import** { HeroService } **from** './hero.service'; **import** { MessageService } **from** './message.service'; **import** { MessagesComponent } **from** './messages/messages.component';  @NgModule({  declarations: [  AppComponent,  HeroesComponent,  HeroDetailComponent,  MessagesComponent  ],  imports: [  BrowserModule,  FormsModule  ],  providers: [ HeroService, MessageService ],  bootstrap: [ AppComponent ] }) **export class** AppModule { } |
| Heroes.component.ts  **import** { Component, OnInit } **from** '@angular/core';  **import** { Hero } **from** '../hero'; **import** { HeroService } **from** '../hero.service';  @Component({  selector: 'app-heroes',  templateUrl: './heroes.component.html',  styleUrls: ['./heroes.component.css'] }) **export class** HeroesComponent **implements** OnInit {   selectedHero: Hero;   heroes: Hero[];   **constructor**(**private** heroService: HeroService) { }   ngOnInit() {  **this**.getHeroes();  }   onSelect(hero: Hero): **void** {  **this**.selectedHero = hero;  }   getHeroes(): **void** {  **this**.heroService.getHeroes()  .subscribe(heroes => **this**.heroes = heroes);  } } | Horo.service.ts  **import** { Injectable } **from** '@angular/core';  **import** { Observable } **from** 'rxjs/Observable'; **import** { of } **from** 'rxjs/observable/of';  **import** { Hero } **from** './hero'; **import** { HEROES } **from** './mock-heroes'; **import** { MessageService } **from** './message.service';  @Injectable() **export class** HeroService {   **constructor**(**private** messageService: MessageService) { }   getHeroes(): Observable<Hero[]> {  // *Todo: send the message \_after\_ fetching the heroes* **this**.messageService.add('HeroService: fetched heroes');  **return** of(HEROES);  } } |

Roution



|  |  |
| --- | --- |
| App.module.ts  **import** { NgModule } **from** '@angular/core'; **import** { BrowserModule } **from** '@angular/platform-browser'; **import** { FormsModule } **from** '@angular/forms';  **import** { AppComponent } **from** './app.component'; **import** { DashboardComponent } **from** './dashboard/dashboard.component'; **import** { HeroDetailComponent } **from** './hero-detail/hero-detail.component'; **import** { HeroesComponent } **from** './heroes/heroes.component'; **import** { HeroService } **from** './hero.service'; **import** { MessageService } **from** './message.service'; **import** { MessagesComponent } **from** './messages/messages.component';  **import** { AppRoutingModule } **from** './app-routing.module';  @NgModule({  imports: [  BrowserModule,  FormsModule,  AppRoutingModule  ],  declarations: [  AppComponent,  DashboardComponent,  HeroesComponent,  HeroDetailComponent,  MessagesComponent  ],  providers: [ HeroService, MessageService ],  bootstrap: [ AppComponent ] }) **export class** AppModule { } | App-routiong.module.ts  **import** { NgModule } **from** '@angular/core'; **import** { RouterModule, Routes } **from** '@angular/router';  **import** { DashboardComponent } **from** './dashboard/dashboard.component'; **import** { HeroesComponent } **from** './heroes/heroes.component'; **import** { HeroDetailComponent } **from** './hero-detail/hero-detail.component';  **const** routes: Routes = [  { path: '', redirectTo: '/dashboard', pathMatch: 'full' },  { path: 'dashboard', component: DashboardComponent },  { path: 'detail/:id', component: HeroDetailComponent },  { path: 'heroes', component: HeroesComponent } ];  @NgModule({  imports: [ RouterModule.forRoot(routes) ],  exports: [ RouterModule ] }) **export class** AppRoutingModule {} |
| App.component.html  <h1>{{title}}</h1> <nav>  <a routerLink="/dashboard">Dashboard</a>  <a routerLink="/heroes">Heroes</a> </nav> <router-outlet></router-outlet> <app-messages></app-messages> |  |
|  |  |

Http

|  |  |
| --- | --- |
| App.module.ts  **import** { NgModule } **from** '@angular/core'; **import** { BrowserModule } **from** '@angular/platform-browser'; **import** { FormsModule } **from** '@angular/forms'; **import** { HttpClientModule } **from** '@angular/common/http';  **import** { HttpClientInMemoryWebApiModule } **from** 'angular-in-memory-web-api'; **import** { InMemoryDataService } **from** './in-memory-data.service';  **import** { AppRoutingModule } **from** './app-routing.module';  **import** { AppComponent } **from** './app.component'; **import** { DashboardComponent } **from** './dashboard/dashboard.component'; **import** { HeroDetailComponent } **from** './hero-detail/hero-detail.component'; **import** { HeroesComponent } **from** './heroes/heroes.component'; **import** { HeroSearchComponent } **from** './hero-search/hero-search.component'; **import** { HeroService } **from** './hero.service'; **import** { MessageService } **from** './message.service'; **import** { MessagesComponent } **from** './messages/messages.component';  @NgModule({  imports: [  BrowserModule,  FormsModule,  AppRoutingModule,  HttpClientModule,   // The HttpClientInMemoryWebApiModule module intercepts HTTP requests  // and returns simulated server responses.  // Remove it when a real server is ready to receive requests.  HttpClientInMemoryWebApiModule.forRoot(  InMemoryDataService, { dataEncapsulation: **false** }  )  ],  declarations: [  AppComponent,  DashboardComponent,  HeroesComponent,  HeroDetailComponent,  MessagesComponent,  HeroSearchComponent  ],  providers: [ HeroService, MessageService ],  bootstrap: [ AppComponent ] }) **export class** AppModule { } | In-memory-data.service.ts  **import** { InMemoryDbService } **from** 'angular-in-memory-web-api';  **export class** InMemoryDataService **implements** InMemoryDbService {  createDb() {  **const** heroes = [  { id: 11, name: 'Mr. Nice' },  { id: 12, name: 'Narco' },  { id: 13, name: 'Bombasto' },  { id: 14, name: 'Celeritas' },  { id: 15, name: 'Magneta' },  { id: 16, name: 'RubberMan' },  { id: 17, name: 'Dynama' },  { id: 18, name: 'Dr IQ' },  { id: 19, name: 'Magma' },  { id: 20, name: 'Tornado' }  ];  **return** {heroes};  } } |
| Dashboar.component.ts  **import** { Component, OnInit } **from** '@angular/core'; **import** { Hero } **from** '../hero'; **import** { HeroService } **from** '../hero.service';  @Component({  selector: 'app-dashboard',  templateUrl: './dashboard.component.html',  styleUrls: [ './dashboard.component.css' ] }) **export class** DashboardComponent **implements** OnInit {  heroes: Hero[] = [];   **constructor**(**private** heroService: HeroService) { }   ngOnInit() {  **this**.getHeroes();  }   getHeroes(): **void** {  **this**.heroService.getHeroes()  .subscribe(heroes => **this**.heroes = heroes.slice(1, 5));  } } | HeroService  **import** { Injectable } **from** '@angular/core'; **import** { HttpClient, HttpHeaders } **from** '@angular/common/http';  **import** { Observable } **from** 'rxjs/Observable'; **import** { of } **from** 'rxjs/observable/of'; **import** { catchError, map, tap } **from** 'rxjs/operators';  **import** { Hero } **from** './hero'; **import** { MessageService } **from** './message.service';  **const** httpOptions = {  headers: **new** HttpHeaders({ 'Content-Type': 'application/json' }) };  @Injectable() **export class** HeroService {   **private** heroesUrl = 'api/heroes'; // URL to web api   **constructor**(  **private** http: HttpClient,  **private** messageService: MessageService) { }   */\*\* GET heroes from the server \*/* getHeroes (): Observable<Hero[]> {  **return this**.http.get<Hero[]>(**this**.heroesUrl)  .pipe(  tap(heroes => **this**.log(`fetched heroes`)),  catchError(**this**.handleError('getHeroes', []))  );  }   */\*\* GET hero by id. Return `undefined` when id not found \*/* getHeroNo404<Data>(id: **number**): Observable<Hero> {  **const** url = `${**this**.heroesUrl}/?id=${id}`;  **return this**.http.get<Hero[]>(url)  .pipe(  map(heroes => heroes[0]), // returns a {0|1} element array  tap(h => {  **const** outcome = h ? `fetched` : `did not find`;  **this**.log(`${outcome} hero id=${id}`);  }),  catchError(**this**.handleError<Hero>(`getHero id=${id}`))  );  }   */\*\* GET hero by id. Will 404 if id not found \*/* getHero(id: **number**): Observable<Hero> {  **const** url = `${**this**.heroesUrl}/${id}`;  **return this**.http.get<Hero>(url).pipe(  tap(\_ => **this**.log(`fetched hero id=${id}`)),  catchError(**this**.handleError<Hero>(`getHero id=${id}`))  );  }   /\* GET heroes whose name contains search term \*/  searchHeroes(term: **string**): Observable<Hero[]> {  **if** (!term.trim()) {  // if not search term, return empty hero array.  **return** of([]);  }  **return this**.http.get<Hero[]>(`api/heroes/?name=${term}`).pipe(  tap(\_ => **this**.log(`found heroes matching "${term}"`)),  catchError(**this**.handleError<Hero[]>('searchHeroes', []))  );  }   //////// Save methods //////////   */\*\* POST: add a new hero to the server \*/* addHero (hero: Hero): Observable<Hero> {  **return this**.http.post<Hero>(**this**.heroesUrl, hero, httpOptions).pipe(  tap((hero: Hero) => **this**.log(`added hero w/ id=${hero.id}`)),  catchError(**this**.handleError<Hero>('addHero'))  );  }   */\*\* DELETE: delete the hero from the server \*/* deleteHero (hero: Hero | **number**): Observable<Hero> {  **const** id = **typeof** hero === 'number' ? hero : hero.id;  **const** url = `${**this**.heroesUrl}/${id}`;   **return this**.http.delete<Hero>(url, httpOptions).pipe(  tap(\_ => **this**.log(`deleted hero id=${id}`)),  catchError(**this**.handleError<Hero>('deleteHero'))  );  }   */\*\* PUT: update the hero on the server \*/* updateHero (hero: Hero): Observable<**any**> {  **return this**.http.put(**this**.heroesUrl, hero, httpOptions).pipe(  tap(\_ => **this**.log(`updated hero id=${hero.id}`)),  catchError(**this**.handleError<**any**>('updateHero'))  );  }   */\*\*  \* Handle Http operation that failed.  \* Let the app continue.  \** ***@param*** *operation - name of the operation that failed  \** ***@param*** *result - optional value to return as the observable result  \*/* **private** handleError<T> (operation = 'operation', result?: T) {  **return** (error: **any**): Observable<T> => {   // *TODO: send the error to remote logging infrastructure* console.error(error); // log to console instead   // *TODO: better job of transforming error for user consumption* **this**.log(`${operation} failed: ${error.message}`);   // Let the app keep running by returning an empty result.  **return** of(result **as** T);  };  }   */\*\* Log a HeroService message with the MessageService \*/* **private** log(message: **string**) {  **this**.messageService.add('HeroService: ' + message);  } } |
|  |  |
|  |  |

### [Angular QuickStart - 4. 어플리케이션 코드 구조 (작성중)](http://han41858.tistory.com/33)

한장현 2017.01.26 17:47

- 이 문서는 작성중입니다. 내용이 생각보다 많이 긴 관계로.... 차근차근 번역하는 중입니다.

Angular 어플리케이션을 구성하는 기본 요소

Angluar는 HTML 클라이언트 어플리케이션을 구성하는 프레임워크이며, JavaScript를 직접 사용하거나 TypeScript와 같이 JavaScript로 컴파일되는 언어를 사용할 수 있다.

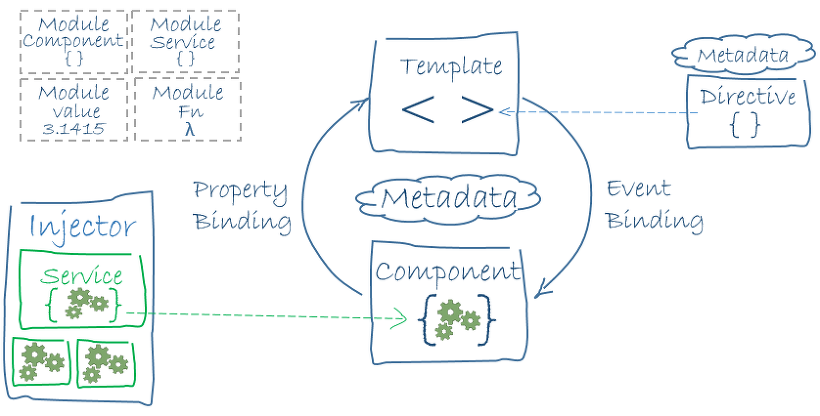
Angular 프레임워크는 수많은 라이브러리들로 이루어지는데, 이 중 일부는 코어 모듈이고 일부는 옵션 모듈이다.

Angular 어플리케이션을 구성하기 위해서는, Angular 스타일의 마크업을 사용하여 템플릿을 만들거나, 이 템플릿을 관리하는 컴포넌트 클래스를 작성하기도 하고, 서비스에 어플리케이션 공통 로직을 작성할 수 있고, 컴포넌트와 서비스들을 묶어서 모듈로 구성할 수도 있다.

루트 모듈을 부트스트랩하는 것으로 앱을 실행 된다. Angular는 당신이 지정한 동작에 따라서 브라우저에 내용물을 표시하고 사용자의 입력에 반응할 것이다.

물론 위에서 말한 것보다 더 복잡한 과정이 있다. 이 문서를 따라가면서 자세한 내용을 알 수 있을 것이다.

자 이제, 전체 그림을 보는 것에서부터 시작하자.



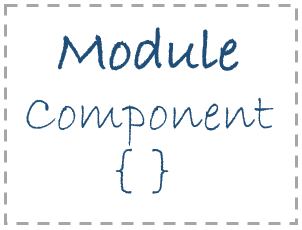
이 다이어그램은 Angular 어플리케이션을 구성하는 8개의 중요한 영역을 나타내고 있다.

* 모듈
* 컴포넌트
* 템플릿
* 메타데이터
* 데이터 바인딩
* 디렉티브
* 서비스
* 의존성 주입

이제 하나씩 살펴보자.

이 페이지에서 언급된 코드들은 [라이브 예제 페이지](https://angular.io/resources/live-examples/architecture/ts/eplnkr.html)에서도 확인할 수 있다.

모듈



Angular 앱은 그 자체로 모듈이고, Angular 모듈이나 NgModule이라고 하는 독자적인 모듈 구조를 사용한다.

 Angular 모듈은 복잡한 영역이다. 이 페이지에서도 모듈에 대해 다루지만 [Angular 모듈 페이지](https://angular.io/docs/ts/latest/guide/ngmodule.html)에서 좀 더 자세한 내용을 다룬다.

 모든 Angular 앱은 반드시 [루트 모듈](https://angular.io/docs/ts/latest/guide/appmodule.html)이라는 Angular 모듈 클래스를 가지며, 편의상 AppModule이라고 부른다.

 어플리케이션이 작은 경우 루트 모듈이 유일한 모듈일 수도 있지만, 대부분의 앱들은 어플리케이션 도메인, 동작 흐름, 연관성 등을 고려하여 좀 더 많은 개수의 모듈을 갖는다.

 루트 모듈이든 일반 모듈이든, Angular 모듈은 @NgModule 데코레이터가 붙는 클래스로 선언된다.

 데코레이터는 JavaScript 클래스를 변형하는 함수다. Angular는 클래스가 어떤 특징을 가지며 어떻게 동작해야 하는지 메타데이터를 사용하는 데코레이터를 여러개 구현해두고 있다. 데코레이터에 대해 좀 더 알아보려면 [이곳](https://medium.com/google-developers/exploring-es7-decorators-76ecb65fb841#.x5c2ndtx0)을 참고하자.

 NgModule도 모듈을 정의하는 메타데이터 객체 하나를 사용하는 데코레이터 함수다. 가장 중요한 프로퍼티는 아래와 같다.

* declarations - 이 모듈에서 사용하는 뷰 클래스를 정의한다. Angular에는 컴포넌트, 디렉티브, 파이프 세 종류의 뷰 클래스가 있다.
* exports - 다른 모듈이나 컴포넌트 템플릿에서 접근할 수 있도록 외부로 공개 선언한다.
* import - export로 공개된 클래스를 다른 컴포넌트 템플릿의 this 모듈에 선언해서 사용할 때 사용한다.
* providers - 전역에서 사용되는 서비스를 해당 객체에서 사용할 수 있도록 생성하고 지정한다. 프로바이더는 앱의 모든 곳에서 접근할 수 있다.
* bootstrap - 루트 컴포넌트라고 하는 메인 어플리케이션의 뷰를 선언한다. bootstrap 항목은 루트 모듈에만 존재한다.

간단한 루트 모듈의 예는 아래와 같다.

import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

@NgModule({

imports: [ BrowserModule ],

providers: [ Logger ],

declarations: [ AppComponent ],

exports: [ AppComponent ],

bootstrap: [ AppComponent ]

})

export class AppModule { }

출처: <http://han41858.tistory.com/33> [한장현입니다.]