Dependency injection (DI) is a design pattern where objects are passed to another object to complete the tasks. ... In Angular we specify providers for services using @Injectable() , @NgModule() and @Component() decorators

In angular a service or component may require other dependent services to complete a task. Angular uses dependency injection design pattern to fulfill these dependencies. The advantage of dependency injection design pattern is to divide the task among deferent services. The client service will not create the dependent object itself rather it will be created and injected by an Angular injector. The responsibility of Angular injector is creating service instances and injecting them into classes like components and services. Angular creates root injector during bootstrap process and then creates other injectors. Angular injectors do not know automatically how to create service instances, so we need to specify providers for every service otherwise service instance will not be injected. Injector creates singleton object of a service and hence same object is injected in components and services.   
In Angular we specify providers for services using @Injectable(), @NgModule() and @Component() decorators. Dependency injection in the Angular components and services can be achieved using constructor or Injector. Now find the complete example of Angular dependency injection step by step.

Model:

export class Tech {

id: number;

name: string;

isSecret = false;

}

Model-data

import { Tech } from './Tech';

export const TECHES: Tech[] = [

{ id: 11, isSecret: false, name: 'Dr Nice' },

{ id: 12, isSecret: false, name: 'Narco' },

{ id: 13, isSecret: false, name: 'Bombasto' },

{ id: 14, isSecret: false, name: 'Celeritas' },

{ id: 15, isSecret: false, name: 'Magneta' },

{ id: 16, isSecret: false, name: 'RubberMan' },

{ id: 17, isSecret: false, name: 'Dynama' },

{ id: 18, isSecret: true, name: 'Dr IQ' },

{ id: 19, isSecret: true, name: 'Magma' },

{ id: 20, isSecret: true, name: 'Tornado' }

];

Parent Component

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

@Component({

selector: 'app-Teches',

template: `

<h2>Teches</h2>

<app-Tech-list></app-Tech-list>

`

})

export class TechesComponent { }

Child component

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

import { TECHES } from './mock-Teches';

@Component({

selector: 'app-Tech-list',

template: `

<div \*ngFor="let Tech of Teches">

{{Tech.id}} - {{Tech.name}}

</div>

`

})

export class TechListComponent {

Teches = TECHES;

}

Create an injectable service class

ng generate service Teches/Tech

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

import { TECHES } from './mock-Teches';

@Injectable({

// we declare that this service should be created

// by the root application injector.

providedIn: 'root',

})

export class TechService {

getTeches() { return TECHES; }

}

## **Injecting services**

In order for TechListComponent to get Teches from TechService, it needs to ask for TechService to be injected, rather than creating its own TechService instance with new.

You can tell Angular to inject a dependency in a component's constructor by specifying a **constructor parameter with the dependency type**. Here's the TechListComponent constructor, asking for the TechService to be injected.

Child component with DI

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

import { Tech } from './Tech';

import { TechService } from './Tech.service';

@Component({

selector: 'app-Tech-list',

template: `

<div \*ngFor="let Tech of Teches">

{{Tech.id}} - {{Tech.name}}

</div>

`

})

export class TechListComponent {

Teches: Tech[];

constructor(TechService: TechService) {

this.Teches = TechService.getTeches();

}

}