Angular Fundamentals

.NET

Angular is an application design framework and development platform for creating efficient and sophisticated single-page applications.

HTTPS://ANGULAR.IO/DOCS

What is Angular

https://hackr.io/blog/angular-interview-questions
https://angular.io/guide/aot-compiler

TODO – talk about Ahead-of-time compilation

Question: What is the AOT (Ahead-Of-Time) Compilation? What are its advantages?

Answer: An angular application consists of components and templates which a browser cannot understand. Therefore, every Angular application needs to be compiled before running inside the browser. The Angular compiler takes in the JS code, compiles it, and then produces some JS code. It is known as AOT compilation and happens only once per occasion per user.

There are two kinds of compilation that Angular provides:

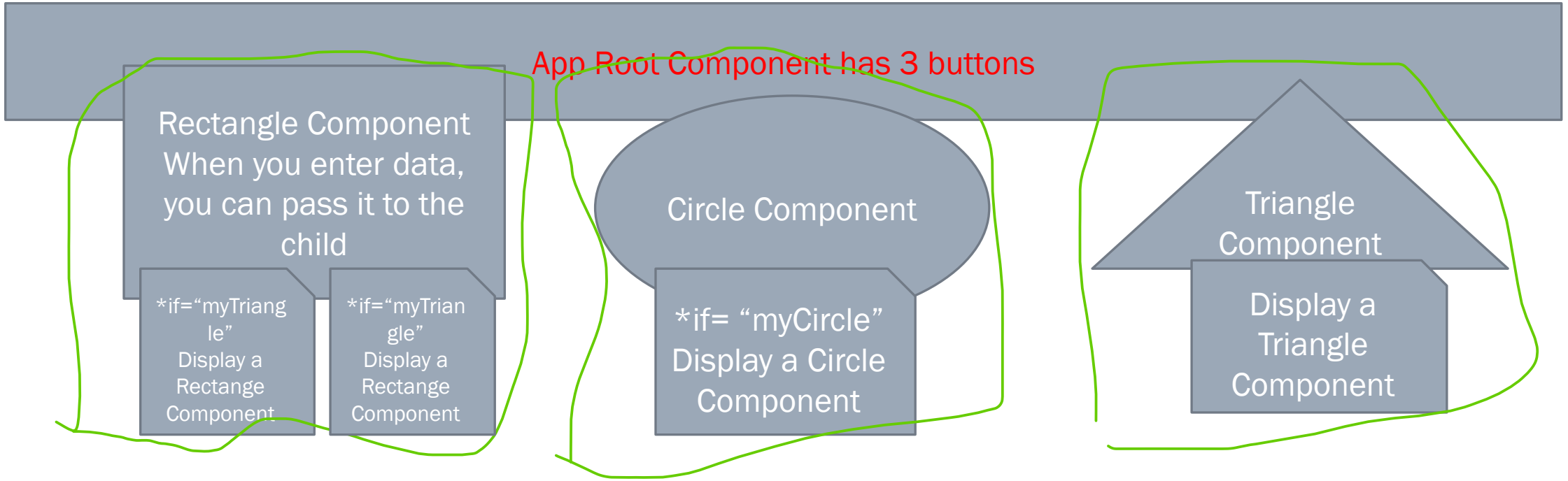
JIT(Just-in-Time) compilation: the application compiles inside the browser during runtime

AOT(Ahead-of-Time) compilation: the application compiles during the build time.

Advantages of AOT compilation:

- **Fast Rendering:** The browser loads the executable code and renders it immediately as the application is compiled before running inside the browser.
- **Fewer Ajax Requests:** The compiler sends the external HTML and CSS files along with the application, eliminating AJAX requests for those source files.
- **Minimizing Errors:** Easy to detect and handle errors during the building phase.
- **Better Security:** Before an application runs inside the browser, the AOT compiler adds HTML and templates into the JS files, so there are no extra HTML files to be read, thus providing better security for the application.

Angular Structure Diagram use shapes for example.



TS/Angular Workspace SetUp

https://angular.io/guide/setup-local
https://code.visualstudio.com/docs/typescript/typescript-compiling
https://angular.io/tutorial/toh-pt0#create-a-new-workspace-and-an-initial-application

---

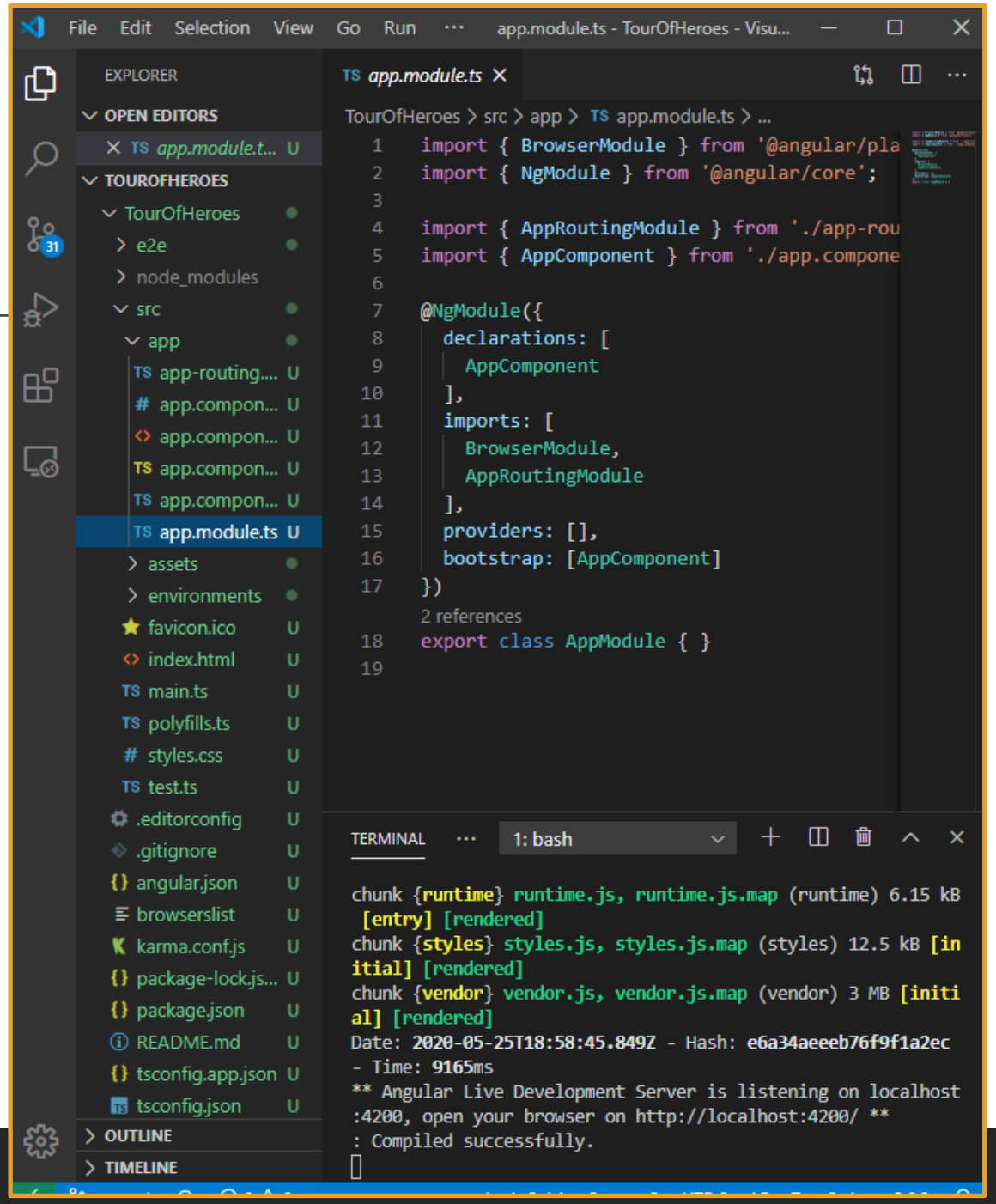
Following the steps from here to create your first Angular App.

1. Make sure you have Node.js with node –v in Command Line. If not, go to nodejs.org to get it.
2. (This is required only once ever) Install Angular CLI globally with: npm install -g @angular/cli
3. Create a ***WorkSpace*** (accept all the default settings) for your app and install the default starter app with: ng new <my-app-name>
4. App name must start with a letter and only contain numbers, letters, and dashes.
5. Install the Angular ***npm*** packages needed with: ng new
6. Navigate in the CLI to your app folder with: cd <my-app-name>
7. Launch the server and open the browser with the default sample project with: ng serve –open (2 dashes)
8. In VS Code, you can install the ***Angular Extension Pack*** to get additional useful tools.
9. VS Code extensions recommendations: C#, C# Extensions, Bracket Pair Colorizer 2, Nuget Gallery, Material Icon Theme,
10. Use this Angular Cheat Sheet for quick reference!

Angular WorkSpace

https://angular.io/tutorial/toh-pt0#set-up-your-environment

A workspace contains all the files for one or more projects. A project is the set of files that comprise an app, a library, or end-to-end (e2e) tests.



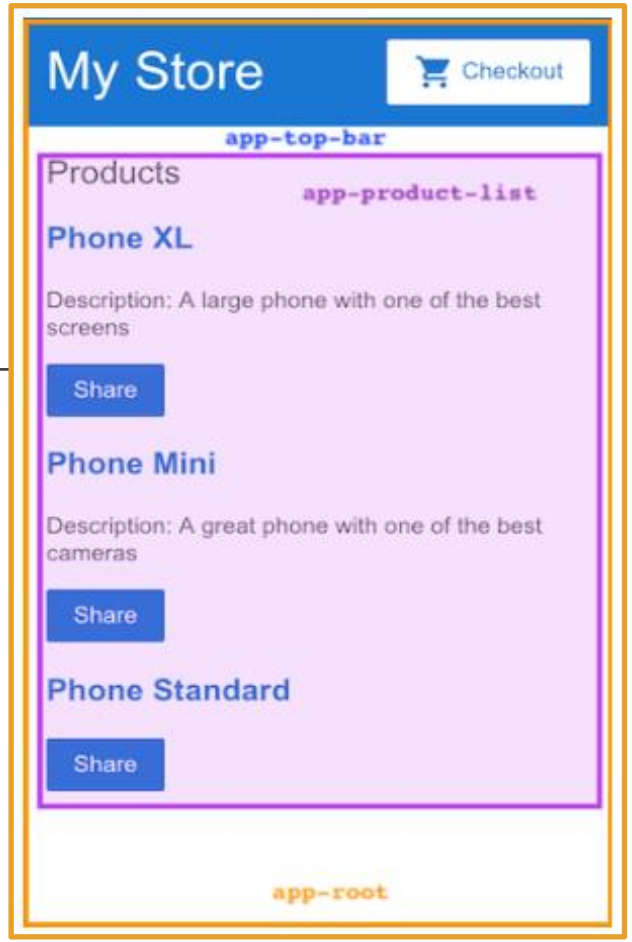
Angular Component

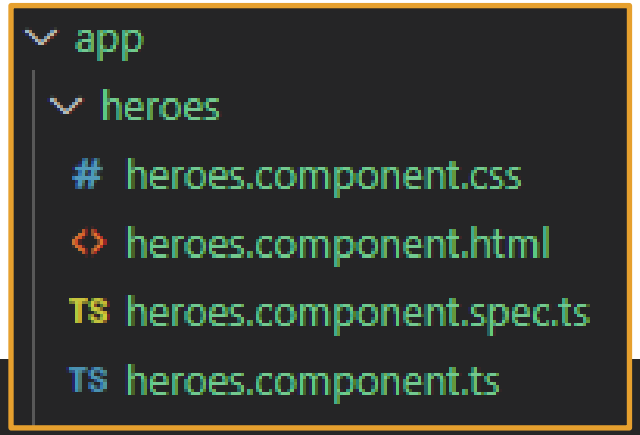
https://angular.io/tutorial/toh-pt0#set-up-your-environment
https://angular.io/guide/component-interaction

Components are the fundamental building blocks of ***Angular*** applications. They display data on the screen, listen for user input, and act based on that input.

An ***Angular*** application comprises a tree of ***components***. Each ***Angular component*** has a specific purpose and responsibility. In the example to the right, there are 3 components displayed:

- ***app-root*** (orange box) is the application shell. This is the first component to load and the parent of all other components. You can think of it as the base page.
- ***app-top-bar*** (blue banner) is the store name and checkout button.
- ***app-product-list*** (purple box) is the product list.





Angular Component

https://angular.io/tutorial/toh-pt1#create-the-heroes-component

Use either the Angular helper (R-click the app folder) or the command ng generate component [name] to create a new ***component***. The ***CLI*** creates a new folder for each ***component*** and generates a .css, .ts, and .html, inside it.

Always import { Component, OnInit } from @angular/core; library.

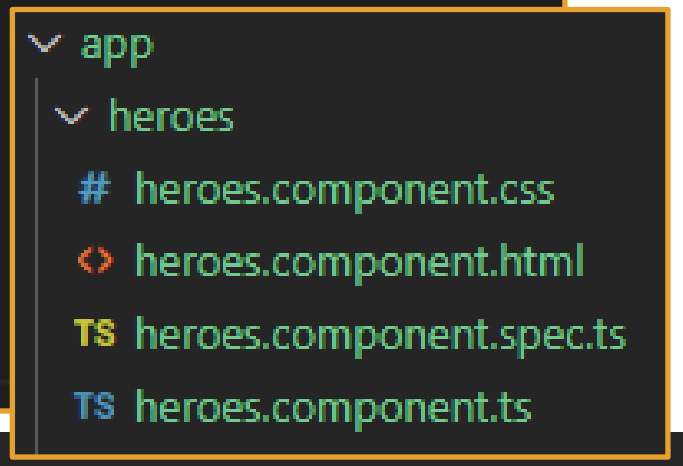
Annotate the ***component class*** with @Component(). @Component is a ***decorator*** function that specifies the Angular metadata for the ***component***:

1. The selector name to use for CSS and if importing this component into a .html page.
2. The relative .html location.
3. The relative .css location.

Use export to make the class available for import by other components.

ngOnInit() is a ***lifecycle hook***. It's the best place for @Component initialization logic, such as getting current data from a ***Service*** or initializing variables.

```
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-heroes',
5    templateUrl: './heroes.component.html',
6    styleUrls: ['./heroes.component.css']
7  })
   7 references
8  export class HeroesComponent implements OnInit {
9
       0 references
10     constructor() { }
11
       2 references
12     ngOnInit(): void {
13     }
14
15 }
16
```



Connect a new Component

https://angular.io/tutorial/toh-pt1#show-the-heroescomponent-view

Every ***component*** must be declared in @NgModule to function.

Angular CLI automatically imports the new component into app.module.ts and declares it under the @NgModule.declarations array.

```
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 { }
```

Angular TypeScript Interface

https://angular.io/tutorial/toh-pt1#create-a-hero-interface

Interfaces are useful for when you want to define a class or object (with its types), then import it into components where needed.

Create an ***interface*** with:

- ng generate interface <ComponentName>.

Then import that ***interface*** into the ***Component*** in which you want to use it from the relative file location.

src/app/hero.ts

```
export interface Hero {
  id: number;
  name: string;
}
```

```
1  import { Component, OnInit } from '@angular/core';
2  import { Hero } from '../hero';
```

TypeScript Modules

https://www.typescriptlang.org/docs/handbook/modules.html

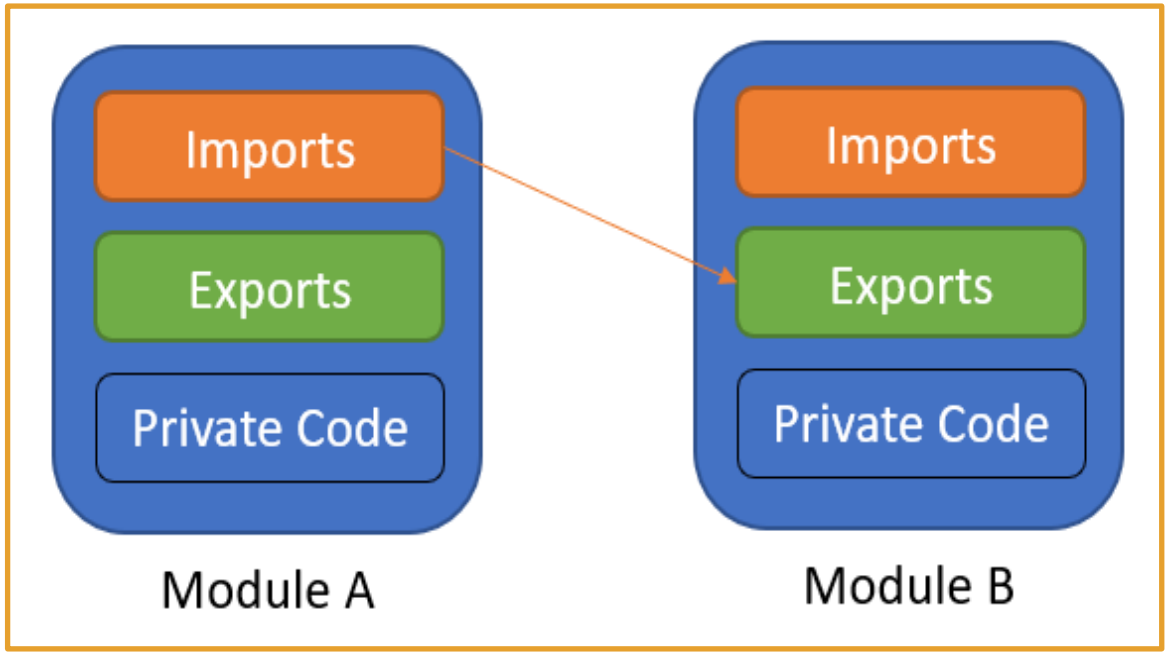
TS shares the *JS* concept of ***Modules***. ***Modules*** in *TS* have their own scope. A module must be explicitly exported to make its members visible.

To consume a property ***exported*** from a different ***module***, it must be ***imported*** using an ***import*** method.

The relationships between ***modules*** are specified in terms of ***imports*** and ***exports*** at the file level.

In *TS*, any file containing a top-level ***import*** or ***export*** is considered a ***module***.

A file without any top-level ***import*** or ***export*** declarations is treated as a script whose contents are available in the global scope (and therefore in ***modules*** as well).



TypeScript - Exporting a Declaration

https://www.typescriptlang.org/docs/handbook/modules.html#export

Any declaration (variable, function, class, type alias, interface) can be ***exported***.

1. Use the ***export*** keyword to make a class, function, or variable available to other ***modules*** from within the ***module*** (***component***).
2. ***Import*** the class, function, or variable into the ***module*** (***component***) where you want to implement it.

```
export interface StringValidator {
  isAcceptable(s: string): boolean;
}
```

```
import { StringValidator } from "./StringValidator";

export const numberRegexp = /^[0-9]+$/;

export class ZipCodeValidator implements StringValidator {
  isAcceptable(s: string) {
    return s.length === 5 && numberRegexp.test(s);
  }
}
```

Dependency Injection – Services and Injectables

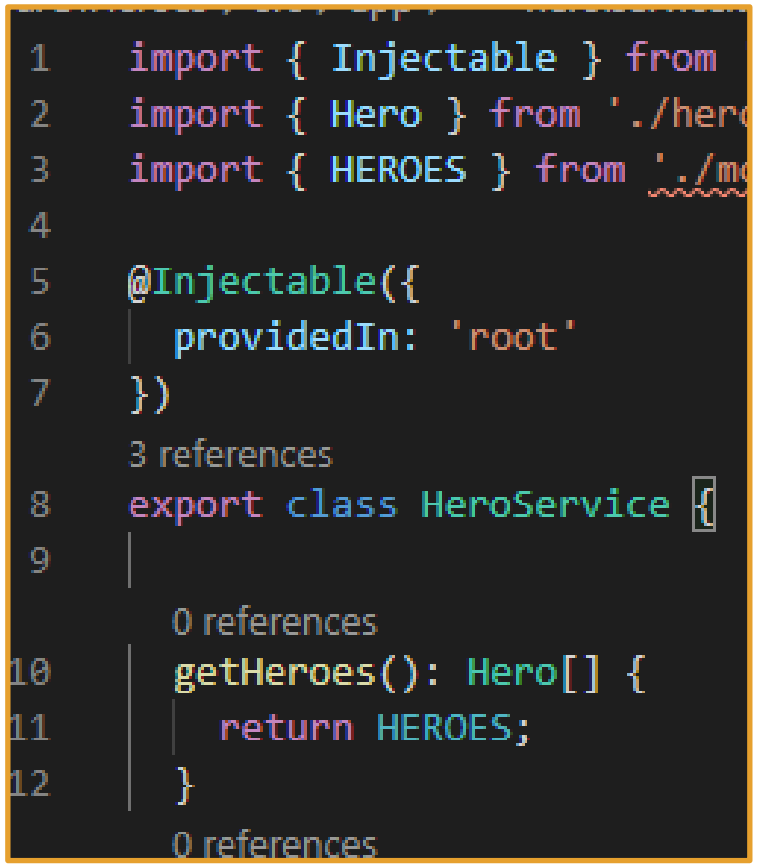
https://angular.io/guide/glossary#dependency-injection-di
https://angular.io/guide/dependency-injection

Components should always delegate data access to a ***Service***. A ***Service*** can get data from an API web service, local storage, or a mock data source, etc.

Services are integral to Angular. A ***service*** is an instance of a class that you can make available to any part of your application using ***Angular***'s ***Dependency Injection*** system.

A ***Service*** is your portal to persist data and have methods to access that data.

The @Injectable() decorator accepts a metadata object for the service, the same way the @Component() decorator does for component classes.



Dependency Injection – Services and Injectables

https://angular.io/tutorial/toh-pt4#provide-the-heroservice
https://angular.io/guide/dependency-injection
https://angular.io/guide/architecture-services

Services must be registered with Angular's Dependency Injection system before they can be injected into a ***Component***.

By default, the ***Angular CLI*** command ng generate service registers a ***provider*** with the ***root*** injector for your ***Service*** by including ***provider*** metadata that's providedIn: 'root' in the @Injectable() ***decorator*** of the ***Service Component***.

When a ***Service*** is provided at the root level, Angular creates a single, shared instance of the ***Service*** and injects it into any class that asks for it.

Angular will also remove any unused ***Services***.

```
1   import { Injectable } from
2   import { Hero } from './her
3   import { HEROES } from './m
4
5   @Injectable({
6     providedIn: 'root'
7   })
    3 references
8   export class HeroService {
9
      0 references
10    getHeroes(): Hero[] {
11      return HEROES;
12    }
      0 references
```

Angular – Use DI to Get a Service

https://angular.io/tutorial/toh-pt4

To create a service to access your stored data,

1. Create a ***Service***:
   - ng generate service <serviceName>.
2. Import the ***Injectable*** symbol into the ***Service Component*** To allow the ***Service*** to be injected into ***Components***:
   - import { Injectable } from '@angular/core';
3. Import the ***Service*** into the ***Component*** where it will be used:
   - import { ServiceName } from '../relative/location';.
4. Inject the ***Service*** into the constructor of the ***Component*** where it will be used:
   - constructor(private ServiceVariableName: ServiceName) {}.

Use ngOnInit() to access and retrieve data from a service on instantiation of the ***Component*** instead of using the constructor.

```
2   import { Hero } from '../hero';
3   import { HeroService } from '../hero.service';
```

```
0 references | 1 reference
constructor(private heroService: HeroService) {}

1 reference
getHeroes(): void {
  this.heroes = this.heroService.getHeroes();
}
6 references
ngOnInit(): void {
  this.getHeroes();
}
```

How to stop a running Angular Program

https://anthonygiretti.com/2018/03/26/how-to-avoid-port-4200-is-already-in-use-error-with-angular-cli/

---

1. In Command Line, use netstat -ano | findstr :yourPortNumber. (Usually it's 4200 with Angular.) to get your process number (PID). It's on the right or 'Listening'
2. In Command Line, use tskill [yourPID#].
3. In Command Line, use ng serve –open to recompile and reopen your app.