Bootstrapping :

When we bootstrap with the AppComponent class (see main.ts), Angular looks for a <my-app> in theindex.html, finds it, instantiates an instance of AppComponent, and renders it inside the <my-app> tag.

Displaying data

Data interpolation : {{}}

ngFor new Syntax :

<li \*ngFor="let hero of heroes">

{{ hero .me}}

</li>

ngIf new syntax:

<p \*ngIf="heroes.length > 3">There are many heroes!</p>

**Source and target:**

**Target = source**

Property is declared in **[]** if it’s a target.

**<my-hero-detail [hero]="selectedHero"></my-hero-detail>**

Here, hero property is the target as we will be passing the selected hero from app.component to hero details component’s property **hero.**

**Services:**

**Injectable()** is the required component.

Import gives us the concrete service, while to access the function we need an instance.

We cannot instantiate the service.

Reasons:

* If service constructor is change, you hava to change wherever you created the instance.
* Use your java brain and you know this, that if a property is cached by the service, the new instance will not be able to use it.

Solution:

Class MyComponent{

constructor(private heroServiceProp: **HeroService**) { }

}

providers: [HeroService];

Here, param is the hero, understands that this(heroServiceProp) is the injection site of the service. Angular is now ready to give an instance of HeroService whenever **MyComponent** is instantiated.

The providers array tells Angular to create a fresh instance of the HeroService when it creates a new AppComponent.

**ROUTER**

1. Base href=/
2. Router\_Provider in bootstrap
3. Router Directive in main shell comp

Main.ts –

* Imports a **route** from app.route.ts
* this route is provided in the bootstrap function array.

App.route.ts –

* Imports **provideRouter** and **routerConfig.**
* An object of routerConfig type is created.
* **provideRouter** takes the above and returns the route .