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.

**Template Syntax**

Displaying data

Data interpolation : {{}}

**Prohibited expressions:**

**=,;,new,++,--,**

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>

**Template Statements:**

(event)="statement".

Cannot refer to global context, document.

**HTML has attributes while DOM has properties. And angularJs works with prpoperties and not attributes.**

**Source and target:**

**Target = source**

**Target:** Element property or event (rarely attribute).

|  |  |  |
| --- | --- | --- |
| **Binding type** | **Target** | **Examples** |
| Property | Element property Component property Directive property | <img [src] = "heroImageUrl">  <hero-detail [hero]="currentHero"></hero-detail>  <div [ngClass] = "{selected: isSelected}"></div> |
| Event | Element event Component event Directive event | <button (click) = "onSave()">Save</button>  <hero-detail (deleteRequest)="deleteHero()"></hero-detail>  <div (myClick)="clicked=$event">click me</div> |
| Two-way | Event and property | <input [(ngModel)]="heroName"> |
| Attribute | Attribute (the exception) | <button [attr.aria-label]="help">help</button> |
| Class | class property | <div [class.special]="isSpecial">Special</div> |
| Style | style property | <button [style.color] = "isSpecial ? 'red' : 'green'"> |

|  |  |  |
| --- | --- | --- |
| **Data direction** | **Syntax** | **Binding type** |
| One-way from data source to view target | {{expression}}  [target] = "expression"  bind-target = "expression"  [hero]=”selected.hero” | Interpolation Property Attribute Class Style |
| One-way from view target to data source | (target) = "statement"  on-target = "statement"  (onclick) = onSelectHero(hero) | Event |
| Two-way | [(target)] = "expression"  bindon-target = "expression"  [(ngModel)] = “hero” | Two-way |

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 .

Activated Route:

* To pass param across routing.
* Instantiated as service in constructor.
* Hence this instance now give the param.
* Contains the information about a component loaded in an outlet. The information is provided through the params, urlSegments, and data observables.

**Forms:**

1. disableDeprecatedForms()
2. provideForms()  **For the new Form API**