HOA SEN UNIVERSITY

www.hoasen.edu.vn

Angular

Phạm Thị Kim Ngôn

Ngon.phamthikim@hoasen.edu.vn

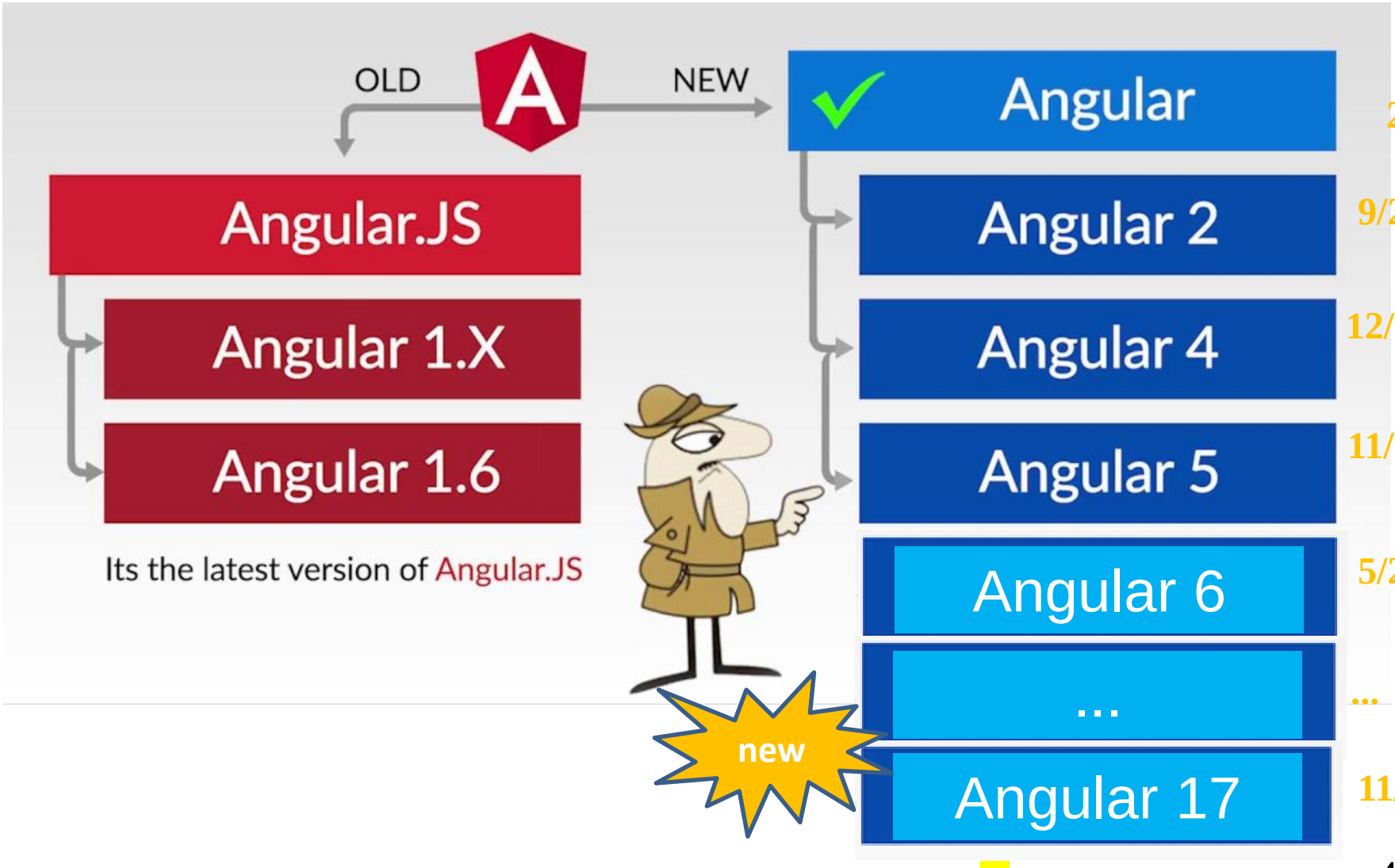
Contents

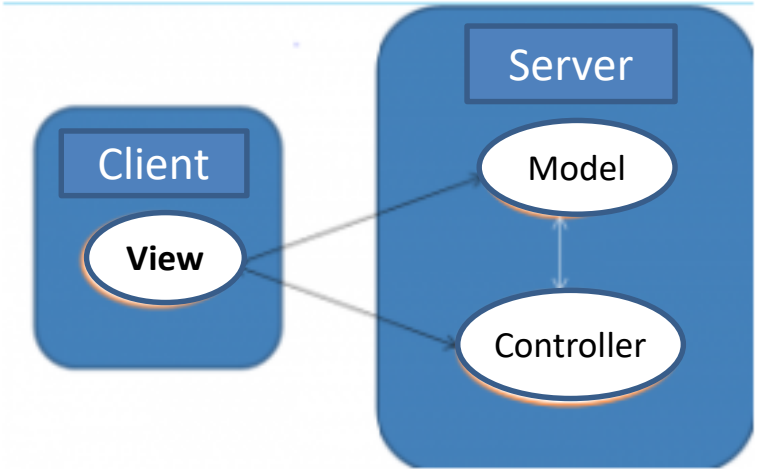
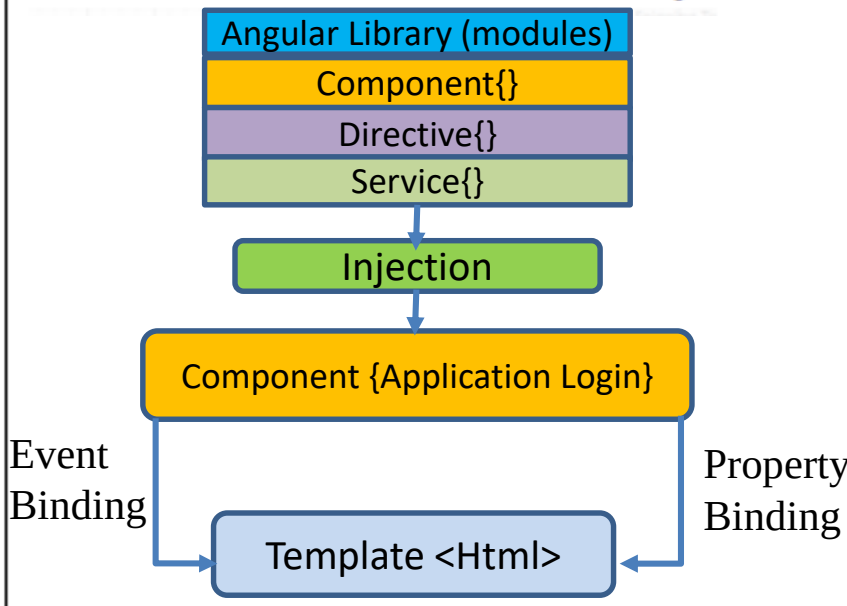
- Introduction
- Versions
- Architecture overview
- Install Angular 2+
- Components
- Data binding

Introduction

- Angular (commonly referred to as "Angular 2+" or "Angular v2 and above") is a TypeScript-based open-source front-end web application platform led by the Angular Team at Google and by a community of individuals and corporations.
- Angular is a complete rewrite from the same team that built AngularJS.
- Angular is an open source JavaScript framework that is used to build single page based web applications.

Versions


OLD
NEW
Angular.JS
Angular 1.X
Angular 1.6
Its the latest version of Angular.JS
Angular
2014
Angular 2
9/2016
Angular 4
12/2016
Angular 5
11/2017
Angular 6
5/2018
...
...
new
Angular 17
11/2023

	Angular 1	Angular 2
Released year	2010	Angular 2 - 2016
Architecture	Controllers and views Based on the Model View Controller. 	Angular 2 is based on a **Components structure**, like what we see in React.js. 
File Type	built using JavaScript.	built on Typescript, which is a superset of JavaScript
Mobile Support	Angular 1 was made for **responsive UI and two-way binding of applications**. But it didn't support mobile. AngularJS was not built for mobile devices	Angular 2 was made with a **mobile-oriented architecture.** NativeScript helped make Angular 2 mobile development faster and more effective. Starting with Angular 2, developers **could create cross-platform applications** with this framework

MVC Pattern

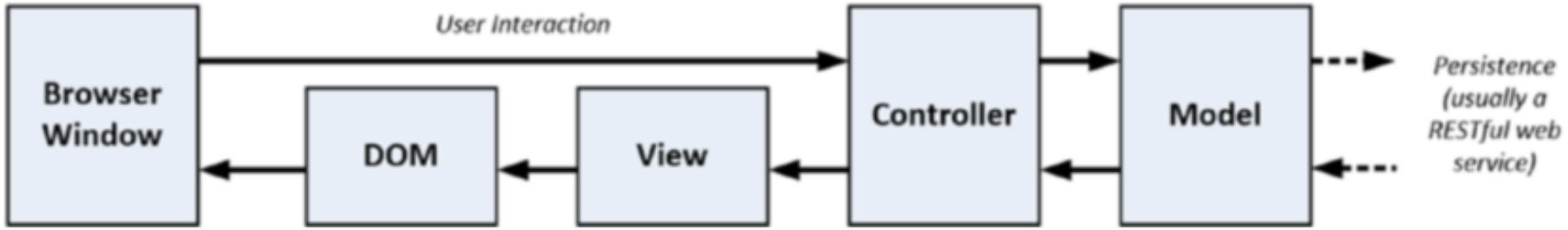


Figure 3-3. *A client-side implementation of the MVC pattern*

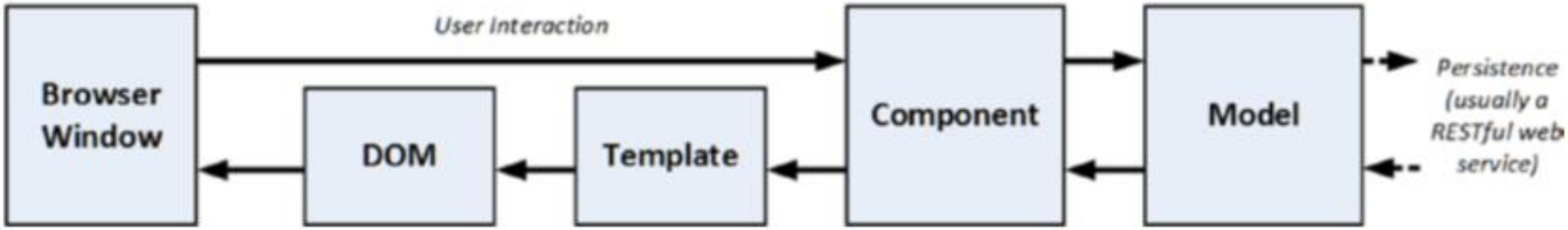
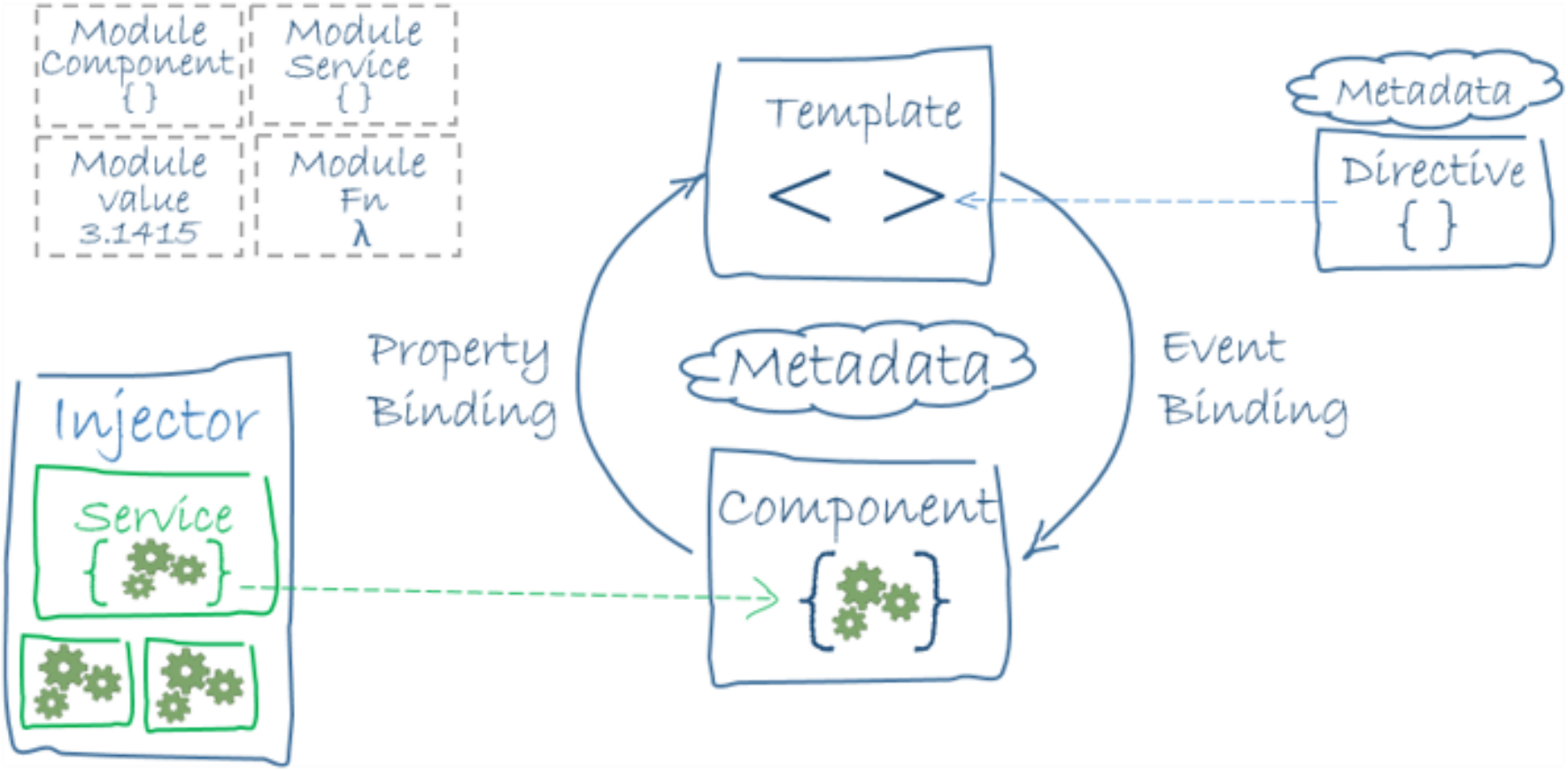


Figure 3-4. *The Angular implementation of the MVC pattern*

Architecture overview

- Angular is a platform and framework for building client applications in HTML and TypeScript.
- Angular is written in TypeScript. It implements core and optional functionality as a set of TypeScript libraries that you import into your apps.
- An app always has at least a *root module* that enables bootstrapping, and typically has many more *feature modules*.
  - **Components** define ***views***, which are sets of screen elements that Angular can choose among and **modify according to your program logic and data.**
  - **Components** use ***services***, which provide specific functionality not directly related to views. Service providers can be *injected* into components as *dependencies*, making your code modular, reusable, and efficient.

Architecture overview



Component

src/app/hero-list.component.ts (class)

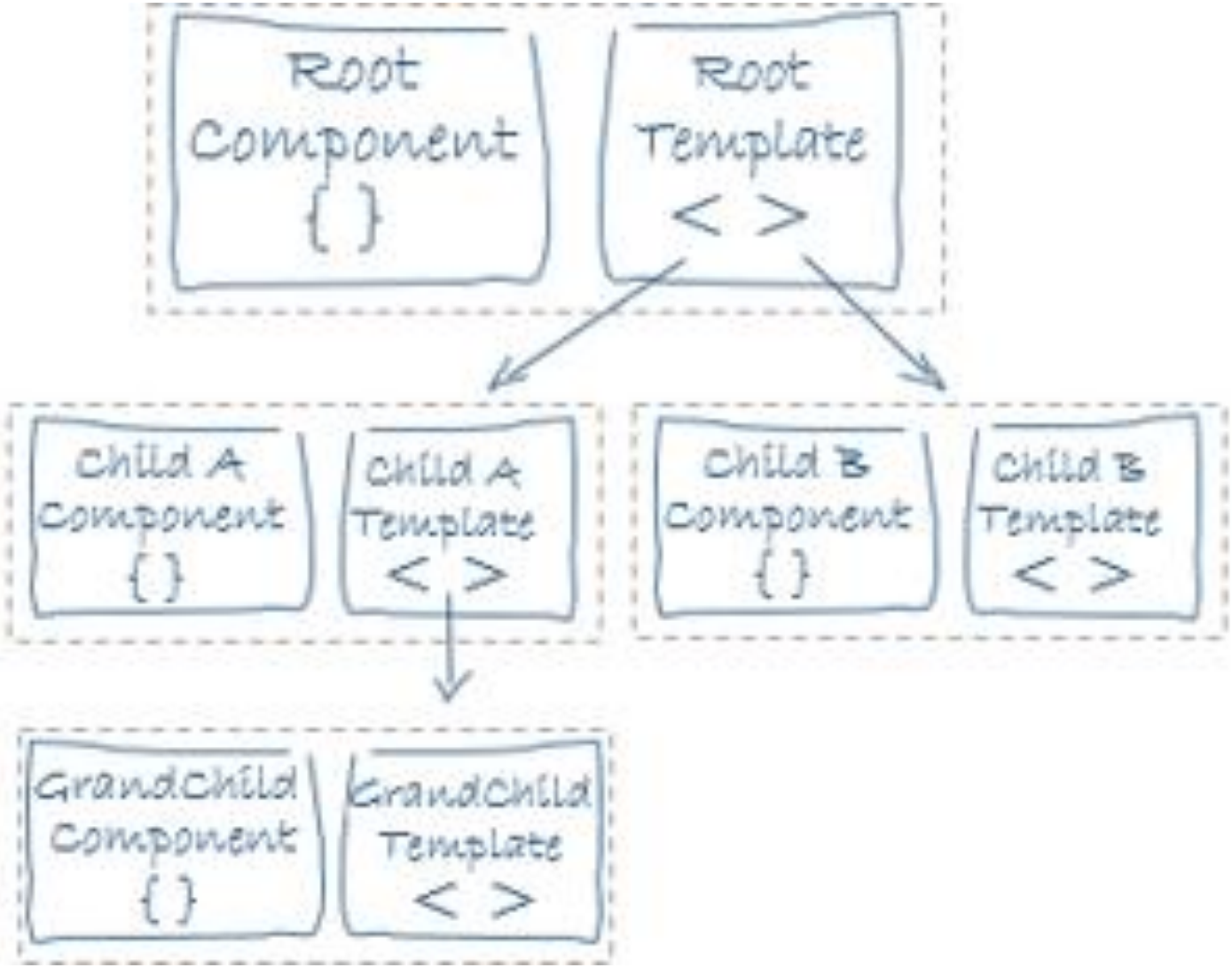
```
export class HeroListComponent implements OnInit {
  heroes: Hero[];
  selectedHero: Hero;

  constructor(private service: HeroService) { }

  ngOnInit() {
    this.heroes = this.service.getHeroes();
  }

  selectHero(hero: Hero) { this.selectedHero = hero; }
}
```

Component metadata

src/app/hero-list.component.ts (metadata)

```
@Component({
  selector:    'app-hero-list',
  templateUrl: './hero-list.component.html',
  providers:  [ HeroService ]
})
export class HeroListComponent implements OnInit {
/* . . . */
}
```

Views and Templates



Dependency injection (DI)

- DI is wired into the Angular framework and used everywhere to provide new components with the services or other things they need. **Components consume services; that is, you can *inject* a service into a component, giving the component access to that service class.**
- To define a class as a service in Angular, use the **@Injectable()** decorator to provide the metadata that allows Angular to inject it into a component **as a dependency**.

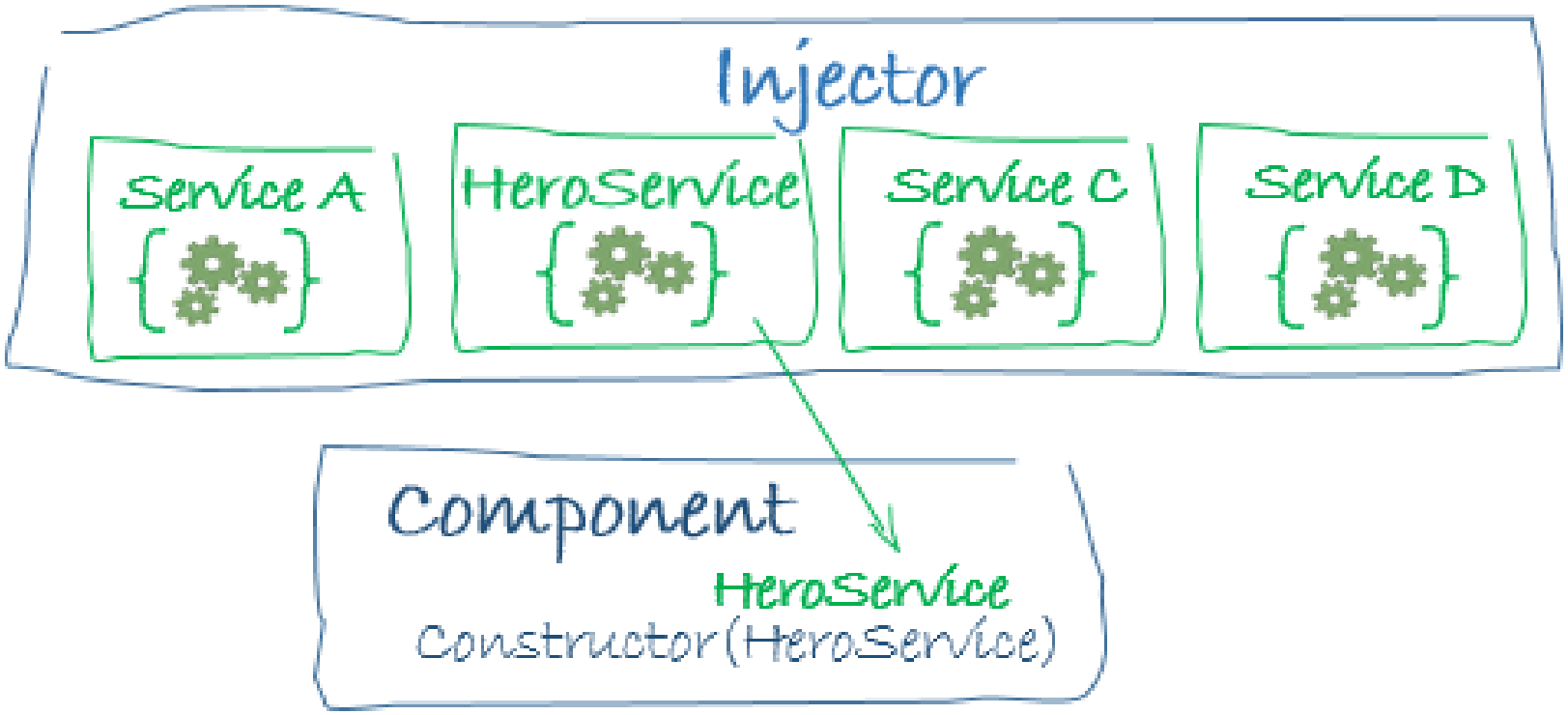
src/app/hero-list.component.ts (constructor)

```
constructor(private service: HeroService) { }
```


```
service: HeroService = inject(HeroService);
```

Version 17+

Dependency injection (DI)



Install Angular

- Install NodeJS
  - Check version Node: `node -v`
  - we will create an empty directory wherein, we will run the Angular CLI command.

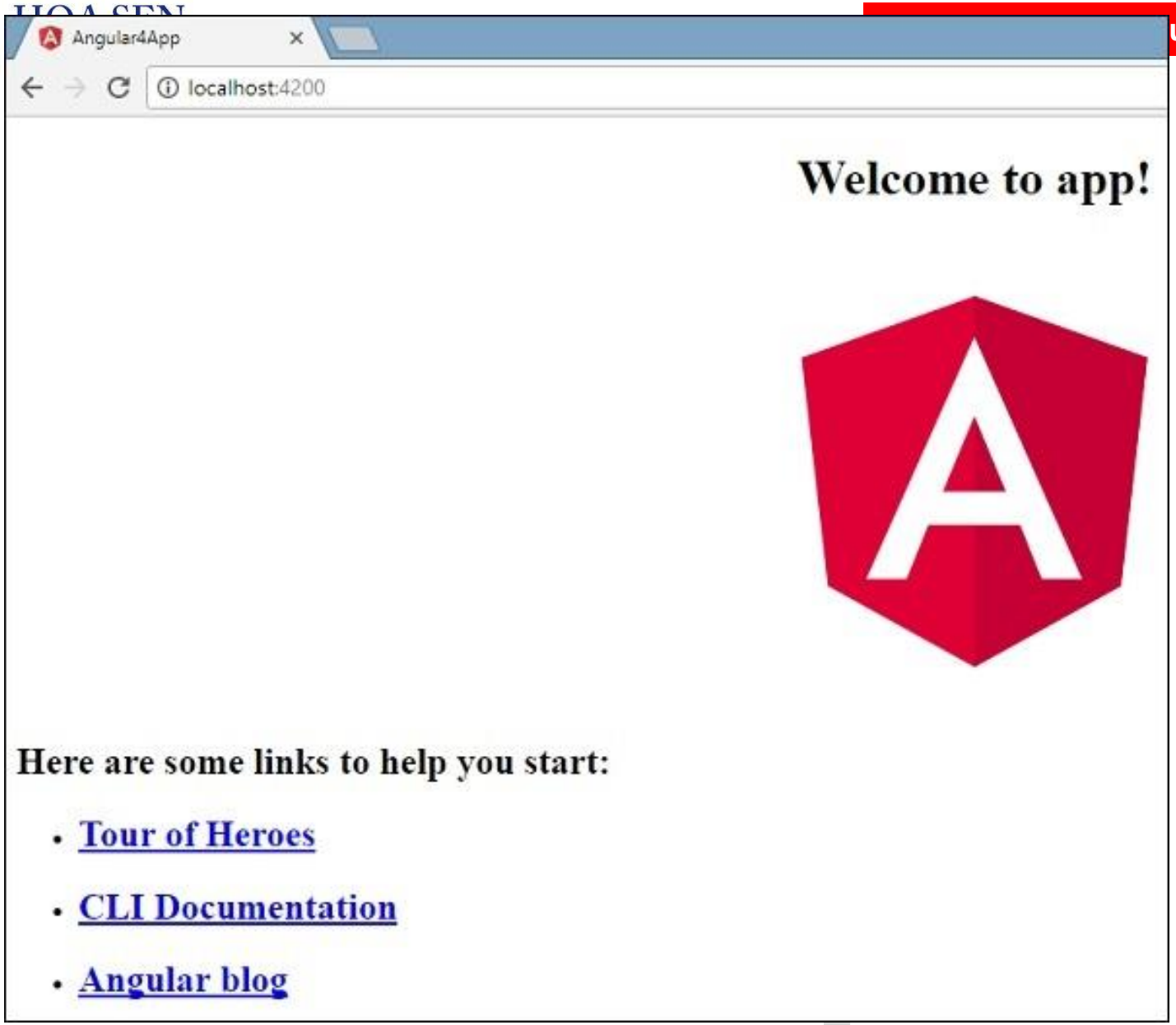
```
npm install -g @angular/cli
npm install -g @angular/cli@latest
```

  - **command to install angular**

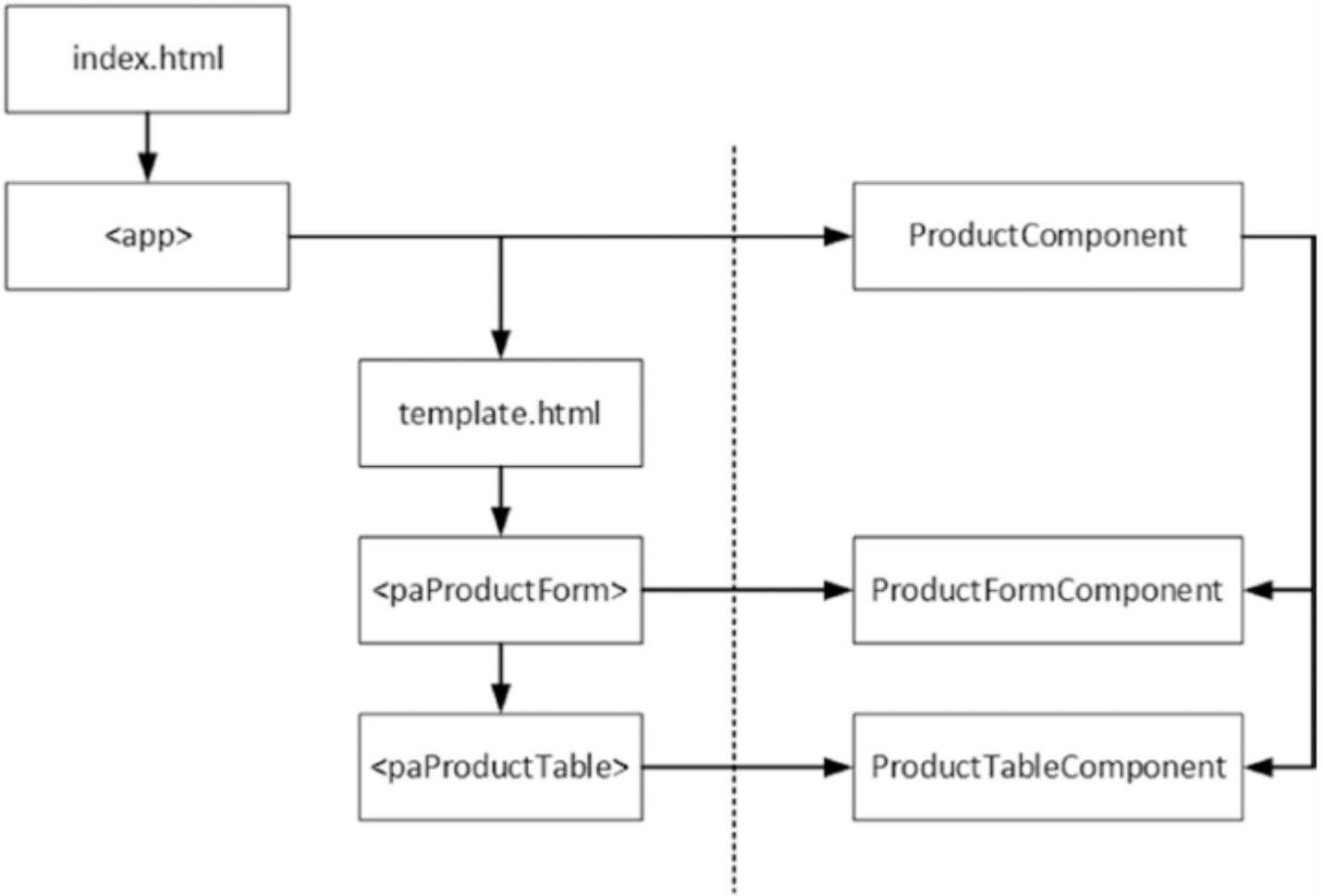
```
ng new Angular-app // name of the project
```

  - builds the application and starts the web server.

```
cd Angular-app
ng serve
```

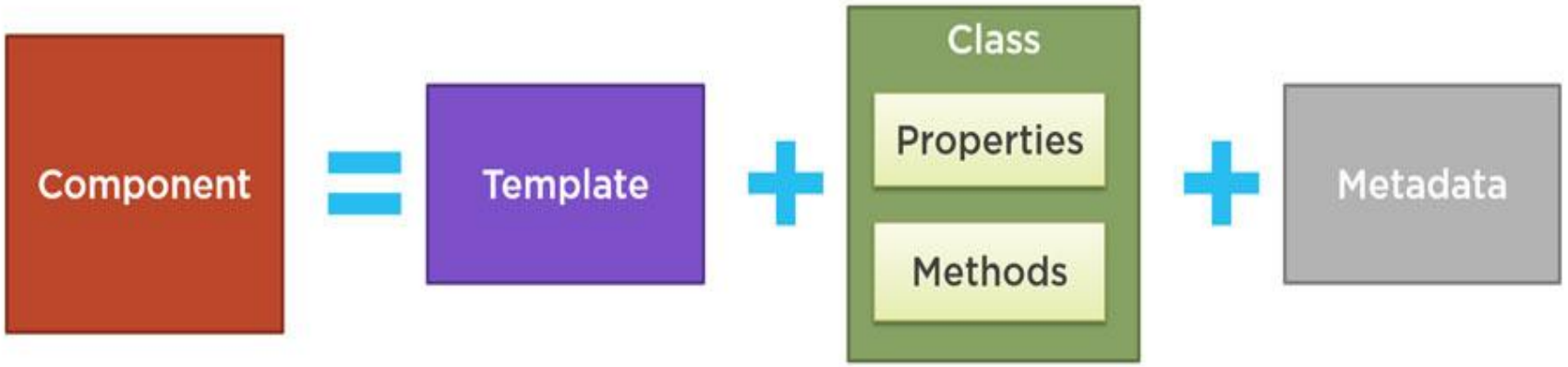

Angular4App
localhost:4200
Welcome to app!
Here are some links to help you start:
Tour of Heroes
CLI Documentation
Angular blog

Application Structure



COMPONENTS

What is component?

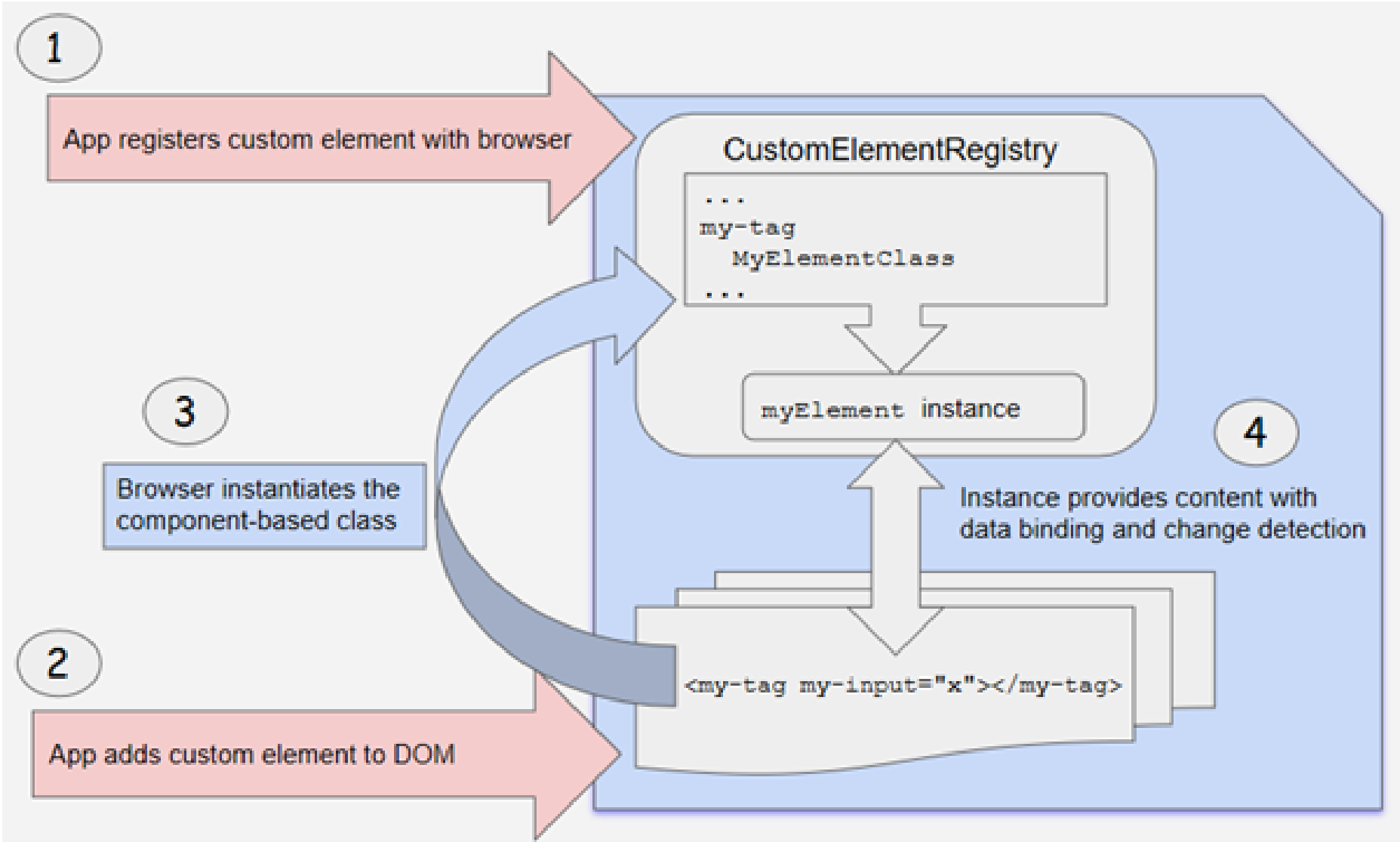


- View layout
- Created with HTML
- Includes binding and directives

- Code supporting the view
- Created with TypeScript
- Properties: data
- Methods: logic

- Extra data for Angular
- Defined with a decorator

Angular Elements Overview



Component Structure

- Components are basically classes that interact with the .html file of the component, which gets displayed on the browser.
- An Angular *component* is responsible for managing a template and providing it with the data and logic it needs
- The file structure has the **app component** and it consists of the following files:
  - app.component.css
  - app.component.html
  - app.component.spec.ts
  - app.component.ts
  - app.module.ts

app.module.ts

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


import { AppComponent } from './app.component';

@NgModule({
    declarations: [
        AppComponent
    ],
    imports: [
        BrowserModule
    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

Create Component

- `ng g component new-cmp`
- The following files are created in the **new-cmp folder**
  - **new-cmp.component.css** – css file for the new component is created.
  - **new-cmp.component.html** – html file is created.
  - **new-cmp.component.spec.ts** – this can be used for unit testing.
  - **new-cmp.component.ts** – here, we can define the module, properties, etc.

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

import { AppComponent } from './app.component';
import { NewCmpComponent } from './new-cmp/new-cmp.component';

@NgModule({
    declarations: [
        AppComponent,
        NewCmpComponent
    ],
    imports: [
        BrowserModule
    ],
    providers: [],
    bootstrap: [AppComponent]
})
export class AppModule { }
```

new-cmp.component.ts

- The app component, which is created by default becomes the parent component. Any component added later becomes the child component.

```
import { Component, OnInit } from '@angular/core';
@Component({
    selector: 'app-new-cmp',
     imports: [CommonModule,AsyncPipe ],
    templateUrl: './new-cmp.component.html',
    styleUrls: ['./new-cmp.component.css']
})
export class NewCmpComponent implements OnInit {

    constructor() { }

    ngOnInit() {
    }
}
```

DATA BINDING OVERVIEW

Showing component properties with interpolation

src/app/app.component.ts

```
1.  import { Component } from '@angular/core';
2.
3.  @Component({
4.    selector: 'app-root',
5.    template: `
6.      <h1>{{title}}</h1>
7.      <h2>My favorite hero is: {{myHero}}
    </h2>
8.      `
9.  })
10. export class AppComponent {
11.   title = 'Tour of Heroes';
12.   myHero = 'Windstorm';
13. }
```

Template inline
Delete app.component.html

@for, *ngFor

src/app/app.component.ts (class)

```
export class AppComponent {
  title = 'Tour of Heroes';
  heroes = ['Windstorm', 'Bombasto', 'Magneta', 'Tornado'];
  myHero = this.heroes[0];
}
```

```
@for (item of items; track item.id) {
  {{ item.name }}
}
```

17+

src/app/app.component.ts (template)

```
template: `
  <h1>{{title}}</h1>
  <h2>My favorite hero is: {{myHero}}</h2>
  <p>Heroes:</p>
  <ul>
    <li *ngFor="let hero of heroes">
      {{ hero }}
    </li>
  </ul>
```

Creating a class for the data

ng generate class hero

```
export class Hero {
    constructor(
        public id: number,
        public name: string) { }
}
```

```
import { Component } from '@angular/core';
import { Hero } from './hero';
@Component({
    selector: 'app-root',
    templateUrl: './app.component.html',
    styleUrls: ['./app.component.css']
})
export class AppComponent {
    title = 'app';
    heroes = [
        new Hero(1, 'Windstorm'),
        new Hero(13, 'Bombasto'),
        new Hero(15, 'Magneta'),
        new Hero(20, 'Tornado')
        ];
        myHero = this.heroes[0];
}
```

*NgIf (version 17-)

```
<h1>{{title}}</h1>
<h2>My favorite hero is:  {{myHero.name}}</h2>
<p>Heroes:</p>
<ul>
    <li *ngFor="let hero of heroes">
    {{ hero.name }}
    </li>
</ul>
<p *ngIf="heroes.length > 3">There are many
heroes!</p>
```

@if (version 17+)

```
<h1>{{title}}</h1>
<h2>My favorite hero is:  {{myHero.name}}</h2>
<p>Heroes:</p>
<ul>
   @for(hero of heroes; track  hero.id){

       <li>{{ hero.name }}</li>
    }
</ul>
@if(heroes.length>3){
       <pThere are many heroes!</p>
}
```

References

- Angular documents: https://angular.io/docs