*[](https://angular.io/)*

***ANGULAR 6***

***Agenda:->***

1. *Architecture overview*
2. *Getting started*
3. *Prerequisites*
4. *Tutorial*
5. *Fundamentals*

***Architecture overview:->***

*A close up of a logo

Description automatically generated*

*1) Modules 2) Components (Templates, directives, and data binding) 3) Services and dependency injection 4) Routing.*

***Getting started :->***

***Prerequisites:->***

***CLI commands:->***

* ***npm install,*** *to install NPM.*
* ***ng new demo(ng new app-name),*** *to Create a new angular project.*
* ***ng g c demo1****, to create a new component.*
* ***ng serve****, to serve the app.*
* ***ng serve –-port 9001****, to change the port.*
* ***ng g m demo1****, to generate module.*
* ***ng g I demo1****, to create an interface.*
* ***ng g s demo1****, to create a service.*
* ***ng g p demo1****, to generate pipe.*
* ***ng g d demo1****, to generate directive.*
* ***ng build –prod****, for production build.*
* ***ng update****, to update the application and etc.*

***Demo project:->***

***OUTPUT:->***

* *Once if you open application you will find will below structure:>*
* *E2e*
* *Node modules*
* *Src*
  + *App*
    - *App.component.html*
    - *App.component.css*
    - *App.component.spec.ts*
    - *App.module.ts*
  + *Assets*
  + *Etc*

**

***Tutorial:->***

* *Let's learn angular practically, run below command to create new project. cmd:* ***ng new angular-tour-of-heroes,*** *after this run cmd:* ***ng serve*** *command to get output****.*** *The page you see is the application shell. The shell is controlled by an Angular component named AppComponent****.***
* *o/p:-> http://localhost:4200/ (default path to see output).*
* ***What is component?***
* *Components are the fundamental building blocks of Angular applications. They display data on the screen, listen for user input, and take action based on the input. component contains 3 types of files which are listed as below.*
* ***app.component.ts****— the component class code, written in TypeScript.*
* ***app.component.html****— the component template, written in HTML.*
* ***app.component.css****— the component's private CSS styles.*

***Basics chapter 1:->***

* *You always import the* ***Component*** *symbol from the* ***Angular core library*** *and annotate the component class with* ***@Component decorator****. @Component is a decorator function that specifies the Angular metadata for the component.*
* ***selector****— the component's CSS element selector. The CSS element selector, 'app-heroes', matches the name of the HTML element that identifies this component within a parent component's template. <app-heroes></app-heroes>(calling in app.component.html). A CSS selector that tells Angular to create and insert an instance of this component wherever it finds the corresponding tag in template HTML.*
* ***templateUrl****— the location of the component's template file.*
* ***styleUrls****— the location of the component's private CSS styles.*
* *The* ***ngOnInit*** *is a lifecycle hook. Angular calls ngOnInit shortly after creating a component. It's a good place to put initialization logic.*

***Modules:->***

* *Angular apps are* ***modular****(module by module) and Angular has its own modularity system called****NgModules.*** *NgModules are containers for a cohesive block of code dedicated to an application domain. They can contain* ***components, service providers****, and other code files whose scope is defined by the containing NgModule.*
* *Every Angular app has at least one NgModule class, the root module, which is conventionally named AppModule and resides in a file named app.module.ts. You launch your app by bootstrapping the root NgModule.*
* *An NgModule is defined by a* ***class decorated with @NgModule().*** *The @NgModule() decorator is a function that takes a single metadata object, whose properties describe the module. The most important properties are as follows.*

***declarations:*** *The components, directives, and pipes that belong to this NgModule.*

***Exports:*** *The subset of declarations that should be visible and usable in the component templates of other NgModules.*

***Imports:*** *Other modules whose exported classes are needed by component templates declared in this NgModule.*

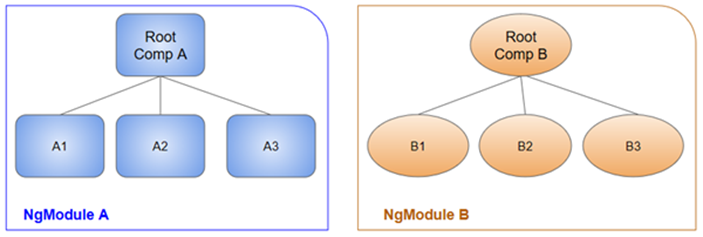
***Providers:*** *Creators of services that this NgModule contributes to the global collection of services; they become accessible in all parts of the app.*

***Bootstrap:*** *The main application view, called the root component, which hosts all other app views. Only the root NgModule should set the bootstrap property.*

*A root NgModule has no reason to export anything because other modules don't need to import the root NgModule. any NgModule can include any number of additional components, which can be loaded through the router or created through the template.*

*Angular loads as a collection of JavaScript modules. Each Angular library name begins with the @angular prefix. We can import parts of them with JavaScript import statements.*

*Eg:->import {* [*BrowserModule*](https://angular.io/api/platform-browser/BrowserModule) *} from '@angular/platform-browser’;*

**

***Components:->***

* ***Components are the fundamental building blocks of Angular applications.*** *A component controls a patch of screen called a view. The service is provided to the component through the dependency injection system.*
* ***@***[***Component***](https://angular.io/api/core/Component)*(As earlier explanation, it’s a decorator, contains metadata about components like how the component can be referenced in HTML and what services it requires).*

*({*

*selector: 'app-hero-list’,*

*templateUrl: './hero-list.component.html’,*

*providers: [ HeroService ]*

*})*

*providers: An array of providers for services that the component requires.*

***Template syntax:->***

*It is a .html file and looks like regular html file. Our template can use data binding to coordinate the app and DOM data, pipes to transform data before it is displayed, and directives to apply app logic to what gets displayed.*

*<h2>Hero* [*List*](https://angular.io/api/common/NumberSymbol)*</h2>*

*<p><i>Pick* [*a*](https://angular.io/api/router/RouterLinkWithHref) *hero from the list</i></p>*

*<ul> <li \*[ngFor](https://angular.io/api/common/NgForOf)="let hero of heroes" (click)="selectHero(hero)“ > {{hero.name}} </li> </ul>*

*<app-hero-detail \*[ngIf](https://angular.io/api/common/NgIf)="selectedHero" [hero]="selectedHero"></app-hero-detail>*

*The \*ngFor is a directive tells Angular to iterate over a list.*

*The <app-hero-detail> tag in the example is an element that represents a new component, HeroDetailComponent.*

***Data binding:->***

* *Data-binding means* ***communication between your typescript code of your component and your template which user sees.*** *There are 4 types of data binding in angular.*

*<li>{{hero.name}}</li>* ***(>>interpolation).***

*<img [src]="heroImageUrl">* ***(>>template property binding).***

*<li (click)="selectHero(hero)"></li>* ***(>>event binding).***

*<input [(****[ngModel](https://angular.io/api/forms/NgModel)****)]="hero.name">* ***(>>2 way data binding).***

*1)* [*Interpolation*](https://angular.io/guide/displaying-data) ***{{hero.name}} :->***

*The easiest way to display a component property is to bind the property name through interpolation. With interpolation, you put the property name in the view template, enclosed in double curly braces {{hero.name}}.*

***2) Property binding:->***

* + *<button [disabled]="isUnchanged">Cancel is disabled</button>.*
  + *Write a template property binding to set a property of a view element. The binding sets the property to the value of a*[***template expression***](https://angular.io/guide/template-syntax)*. Eg: disabling a button when the component says that it isUnchanged.*

*3) E*[*vent binding*](https://angular.io/guide/user-input)*:->*

* + *<button (click)="onClickMe()">Click me!</button> (>>event binding).*
  + *You can use Angular event bindings to respond to any DOM event. Many DOM events are triggered by user input. Binding to these events provides a way to get input from the user. The (click) to the left of the equals sign identifies the button's click event as the target of the binding. The text in quotes to the right of the equals sign is the template statement, which responds to the click event by calling the component's onClickMe method.*

***4) Two-way data binding:->*** *(used mainly in*[***template-driven forms***](https://angular.io/guide/forms)*).*

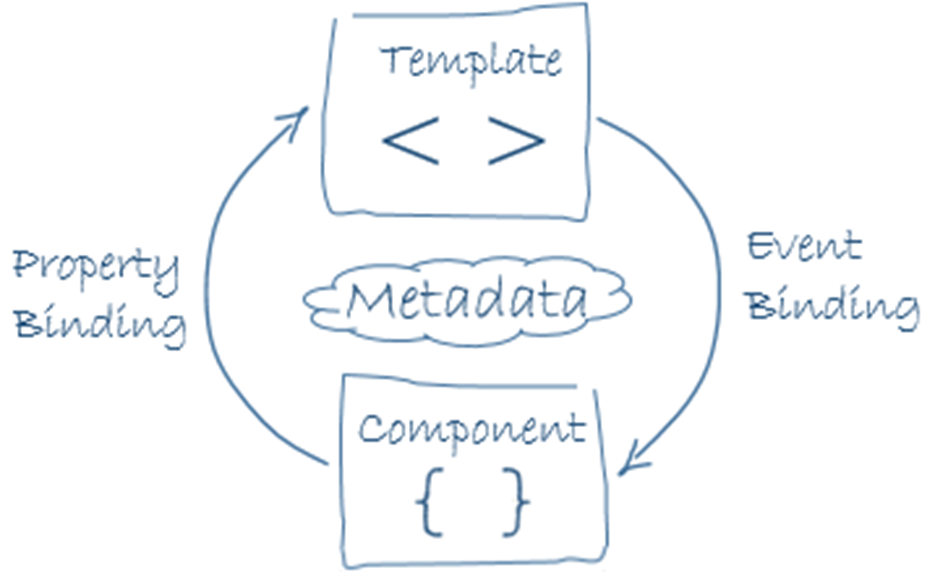
* + *This combines property and event binding in a single notation, Eg: <input [([ngModel](https://angular.io/api/forms/NgModel))]="hero.name"> {{hero.name}}In two-way binding, a data property value flows to the input box from the component as with property binding. The user's changes also flow back to the component, resetting the property to the latest value, as with event binding.*

***5) Class binding:->***

* + *You can add and remove CSS class names from an element's class attribute with a class binding. Class binding syntax resembles property binding. Instead of an element property between brackets, start with the prefix class.*
  + *Ex:-><div class="special“ [class.special]="!isSpecial">This one is not so special</div>*

***6) Style binding:->***

* + *Style binding syntax resembles property binding. Instead of an element property between brackets, start with the prefix style.*
  + *Ex:-><button [style.color]="isSpecial ? 'red': 'green'">Red</button>.*

**

***Assignments:->***

1. *Create button and change background colour, once if we click(apply any colour).*
2. *Create one input field, display the input value in html as well as in typescript file(.ts), it means if we change the value of input field, the same value should update in both file(in html as well as in typescript), example for two way data binding.*
3. *Create a button and disable it, once if we click.*
4. *Keep a image in folder called assets and try to display it in html page(property binding).*
5. *Bind a function for submit button and try to display value in console(any string value), once if we click(event binding).*