Angular cli : it’s a best tool ar a tool to go to , we can perform our all angular commands on it like ng serve to run , ng g c name : to create component

Ng g s sername : to create service in it. And all other.

Module : it’s a powerful unit of angular used to declare several thing that we want to use like ,

Declearation , import , export , and provider

Declearation : used to declare component , directives and pipes .

The main is root module to boostrap the app.

Routing module : used to handle routing functionalities

But in standalone application, use or module reduced lil bit

Directives : it is the special markers that defines the how dom elements behave

Like ex  
  
1. **Component Directives**

* These are the most common type—every Angular component is actually a directive with a template.
* Defined using the @Component decorator.
* Example:

@Component({ selector: 'app-hello', template: `

**Hello World**

` }) export class HelloComponent {}

🎨 2. **Attribute Directives**

* Modify the appearance or behavior of an element.
* Applied as attributes in HTML.
* Common built-in examples:
  + ngClass: Adds/removes CSS classes dynamically.
  + ngStyle: Applies inline styles conditionally.
  + ngModel: Enables two-way data binding in forms.

🧩 3. **Structural Directives**

* Change the DOM layout by adding or removing elements.
* Identified by an asterisk (\*) prefix.
* Common built-in examples:
  + \*ngIf: Conditionally includes a template.
  + \*ngFor: Loops over a collection.
  + \*ngSwitch: Switches between views.
* Example:

Welcome back!

🛠️ Custom Directives

You can also create your own directives using the @Directive decorator to encapsulate reusable logic.

Want help writing a custom directive or exploring how to use one in your app? I’d be happy to walk you through it.

Service in angular

In Angular, a **service** is a reusable class that encapsulates business logic, data access, or utility functions—essentially anything that doesn’t directly deal with the UI

One way and two way binding

One-Way Data Binding

Data flows in a single direction—from the component (TypeScript) to the view (HTML).

* + .

🔄 Two-Way Data Binding

Data flows both ways—changes in the component update the view, and changes in the view update the component.

Pipes in angular

In Angular, **pipes** are elegant tools that transform data directly in your templates—without changing the underlying model.

Ex : datePipe , uppercase pipe , lowercase pipe, currency pipe.

Lifecycle hooks:

Constructor: when the component is just intantiated , or just started

Ngonit: once after the first ngOnChanges

ngDestroy: just before the angular destroy the angular component

ngOnChange: when input property changes , then this life cycle returns / triggered.

String interpolation

You use double curly braces {{ }} to embed expressions:

What You Can Interpolate

- Component properties: {{ username }}

- Method calls: {{ getGreeting() }}

- Arithmetic expressions: {{ price \* quantity }}

- String concatenation: {{ 'Hello ' + name }}

in angular 12 , how to pass data from parent tot child and child to parent

parent to child.

in parent

<app-child [name]=”Amol”></app-child>

In child

Export class child {

@Input() name: String =””;

}

To child to parent

In child

Export class child {

@Output emitter = new Emmiitter<String>();

In one method like

sendData(){

this.emmitter.emit(“hi amol”);

}

}

In parent html

<app-child (dataEmmitter)=”getData($event)”></app-child>

In ts

getData(data: String){

console.log(data);

}

how to store data into localhost or cookies in angular 12

localstorage.set

localstorage.get

localstorage.remove

localstorage.clear

lazy loading in angular.

how inject methods works in angular 12

its used for the dependencies injections , its basically done in constructor also using @Inject