**Angular**

**Observables (rxjs)**

**Angular life cycle hooks**

**project structure of Angular framework**

**Components communication (P-> C )**

**listeners**

1. **Angular CLI (Command Line Interface):-**

is a tool that helps you quickly create and manage Angular applications.

It simplifies tasks like:

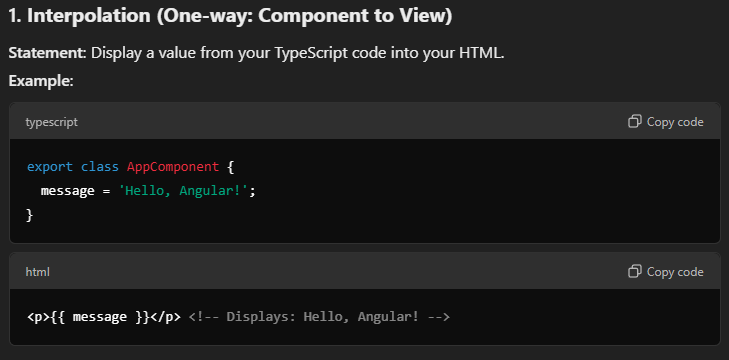
**Creating a new Angular project**:

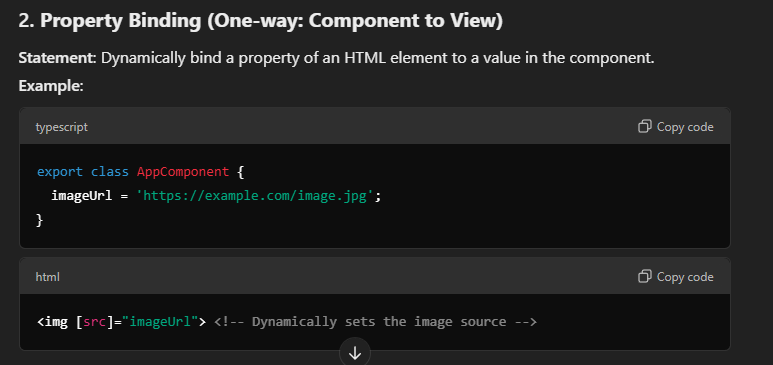
* + Command: ng new my-app
  + This sets up the entire project structure for you.

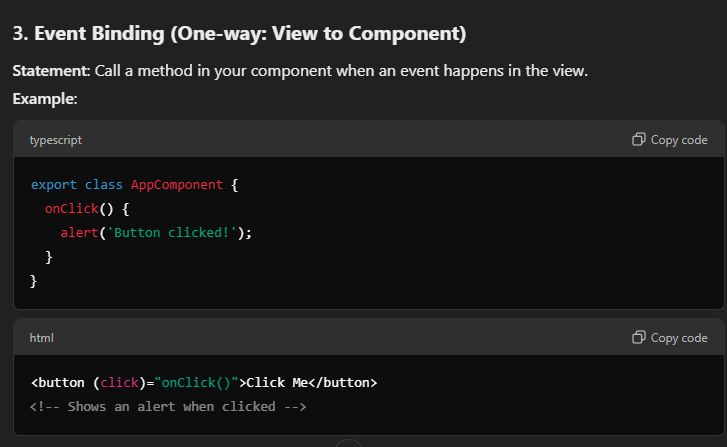
**Generating components, services, and more**:

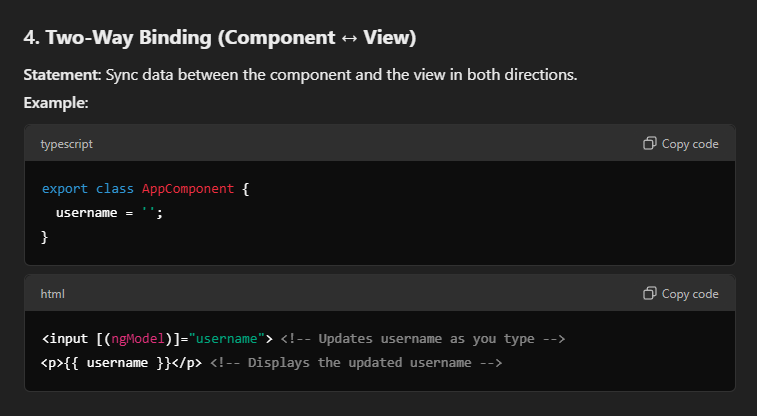
* + Command: ng generate component my-component
  + This creates files and boilerplate code automatically.

1. **Data Bindings Types:-**





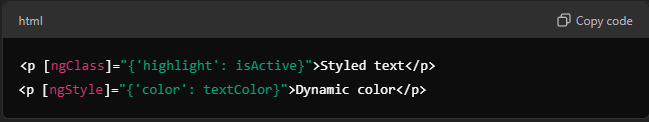




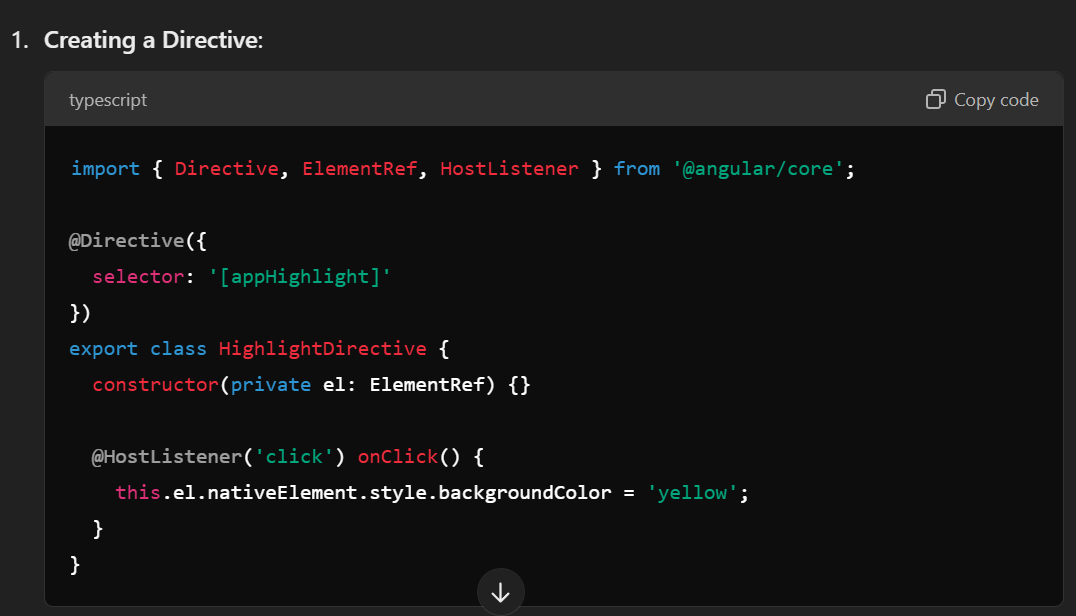
1. **Directives:-**

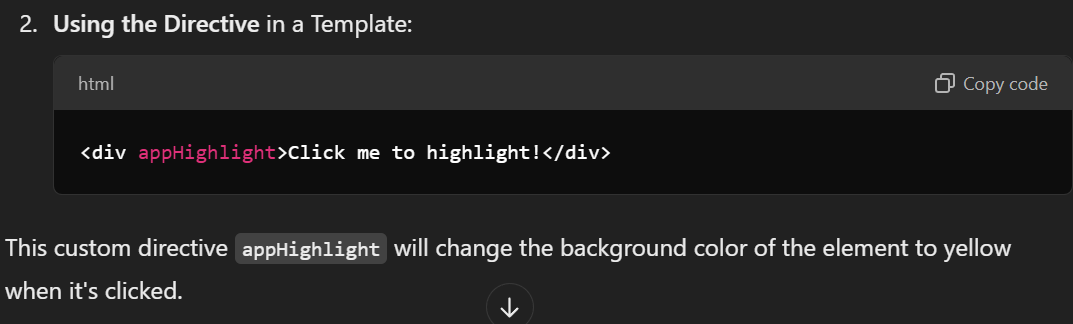
In Angular, **directives** are special instructions added to HTML elements to change their behavior, structure, or appearance. They are like custom attributes or elements that Angular processes to perform certain actions.

* **Structural Directives**: Change the DOM structure (\*ngIf, \*ngFor).
* **Attribute Directives**: Change the element's appearance/behavior (ngClass, ngStyle).



* **Custom Directives**: Your own reusable logic for elements.





1. **Module:-**

a module is like a container or box that groups related parts of an application together. These parts can include components, directives, pipes, and services.

The main purpose of a module is to organize your code and make it easier to manage.

The most important module in any Angular app is the **root module**, which is usually AppModule. It is the starting point for the app.  
decorator for module - @NgModule

Can create custom modules and create components, pipes, directives and services in them.

1. **Node.js:-**

**Node.js** is a tool that allows you to run **JavaScript** code outside of a web browser.

Node.js lets you use JavaScript to build powerful applications on your computer or server.

1. **NPM (Node Package Manager):-**

It’s a tool that comes with Node.js, and it helps you:

* **Download and install libraries/packages**: Ready-made code you can use in your project (e.g., tools for building websites, managing databases, etc.).
* **Share your own code**: If you create something useful, you can publish it on NPM for others to use.

In short: **NPM helps us easily manage and use code libraries in your Node.js projects.**

1. **Pipes:-**

 A **pipe** is a way to transform data in Angular templates before it’s displayed to the user.

 Pipes are used for **data formatting**, **transformations**, and **filtering** without modifying the underlying data.

 You use pipes in **template expressions** like {{ value | pipeName }} to format or change data.

**Types of Pipes**:

**Built**-in pipes:

**<p>{{ amount | currency:'USD' }}</p>**

**<p>{{ 'hello world' | uppercase }}</p>**

<pre>{{ user | json }}</pre>

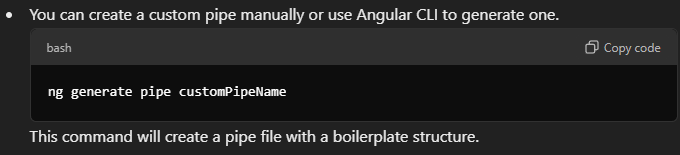
Custom pipes:

 **Custom pipes** in Angular allow you to define your own transformation logic for data before it is displayed in the template.

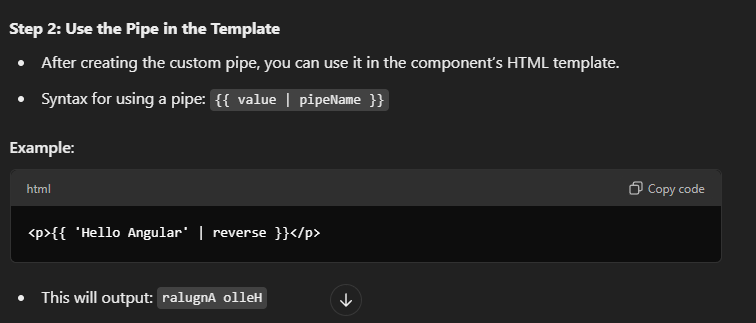
 You can create custom pipes to perform tasks that aren’t handled by built-in pipes (e.g., custom string manipulation, filtering, formatting specific data types, etc.).

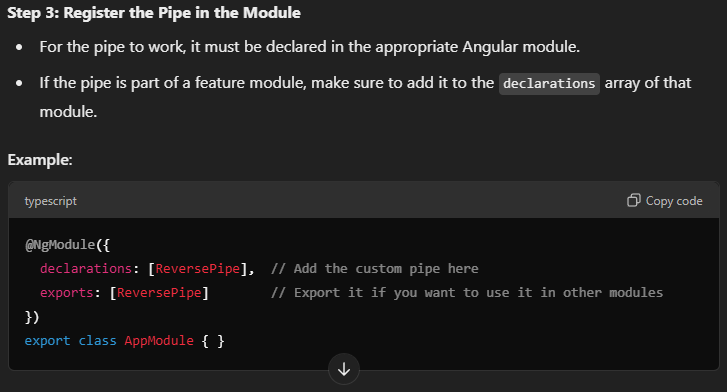
**Steps to Create a Custom Pipe:**

Example: A pipe that reverses a string.









**There are two types of pipes:**

**Pure Pipes:-**

* + A **pure pipe** is a pipe in Angular that only runs when its **input data** changes.
  + Angular **optimizes** pure pipes by executing them only when Angular detects a change in the **input value** or **dependencies**.
  + **t helps with performance optimization by avoiding unnecessary recalculations.**
  + **All pipes in Angular are pure by default unless specified otherwise (pure: false).**

**Impure pipes:-**

* **Impure pipes run every time Angular performs change detection, regardless of whether the input value changes or not.**
* **You create an impure pipe by setting pure: false in the @Pipe decorator.**
* **Use impure pipes when the transformation relies on external sources, side effects, or non-trackable changes.**
* **Be mindful of performance, as impure pipes can become costly if used in large or complex applications.**

1. **Stand Alone:-**

a standalone component, directive, or pipe is an Angular feature that allows you to define and use these building blocks without being part of an Angular module. This was introduced in Angular 14 to simplify application structure and reduce the need for explicitly managing modules.

 A **standalone component**, directive, or pipe can be used independently, without being declared inside a module (NgModule).

 It directly declares its dependencies (like other components, pipes, directives, or Angular libraries) in the @Component, @Directive, or @Pipe metadata.

**Why Use Standalone Components?**

1. Reduces the overhead of managing and importing everything in NgModule.
2. Simplifies smaller or modular applications.
3. Promotes tree-shakable and modular code.

**How to Make Something Standalone?**

ng generate component my-component –standalone

ng generate pipe my-pipe –standalone

ng generate directive my-directive –standalone

to make any class or component stand alone if it is already created(not stand alone) use “standalone: true,”   
 To use this just import that specific class or component only.