**Angular**

**Packages.json:** stores npm packages

**Component decorator:** is used to connect things.

**Data Binding:** There are two ways to data binding:

* One way data binding, component to template or template to component

**Summary of Angular Data Binding Types**

| **Binding Type** | **Direction** | **Syntax** | **Use Case** |
| --- | --- | --- | --- |
| **Interpolation** | Component → Template | {{ expression }} | Displaying values in the UI |
| **Property Binding** | Component → DOM Property | [property]="value" | Dynamically updating properties |
| **Event Binding** | Template → Component | (event)="handler" | Handling user interactions |
| **Two-Way Binding** | Component ↔ Template | [(ngModel)]="prop" | Synchronizing form input and data |

**Directives : It changes the behavior or modify the tags or dom.** In angular there are different types of directive

**Import CommonModule** in component, using this we can use ng modules. Like ngif etc.

**Route:** used to linked and names the components.

Routeoutlet is general tag or directive that show the active component

**Pipes :**  used to transform data before presenting on the page. Or we can say it decorate to upper

Two types of forms :

Template form: Normally create an object and bind with input boxes using ngModel,

Reactive Form: in which we have to use a formGroup in which every property have form control, and this name assigned to the form [formGroup] = name of form group,

APi Call: create object of http with httpClinet and create a function like get post delete etc,  
syntax of calling api is   
http.get(link).subscribe((result:any)=>{

Array = result;})

OnInIt is used to triggers things that are called in this function.

Input output is a way of communication between parent and child.

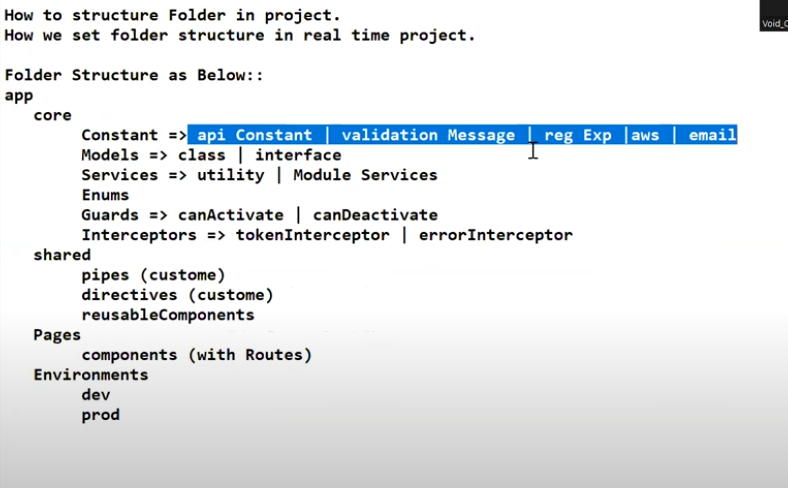
Input -> p-c

Output -> c-p

**Explanation of Folders:**

1. **core:**
   * Contains application-wide single-use functionalities like constants, models, services, enums, guards, and interceptors.
2. **shared:**
   * Holds reusable components, pipes, and directives that can be used across multiple features/modules.
3. **pages:**
   * Contains specific feature modules with their respective components.
4. **environments:**
   * Manages different configurations for development (dev) and production (prod).

app

│

├── core

│ ├── constants

│ │ ├── api.constants.ts

│ │ ├── validation-messages.constants.ts

│ │ ├── regex.constants.ts

│ │ ├── aws.constants.ts

│ │ ├── email.constants.ts

│ │

│ ├── models

│ │ ├── class

│ │ ├── interface

│ │

│ ├── services

│ │ ├── utility

│ │ ├── module-services

│ │

│ ├── enums

│ │

│ ├── guards

│ │ ├── can-activate.guard.ts

│ │ ├── can-deactivate.guard.ts

│ │

│ ├── interceptors

│ ├── token.interceptor.ts

│ ├── error.interceptor.ts

│

├── shared

│ ├── pipes

│ │ ├── custom.pipe.ts

│ │

│ ├── directives

│ │ ├── custom.directive.ts

│ │

│ ├── reusable-components

│ ├── component-name/

│

├── pages

│ ├── feature-module

│ │ ├── components

│ │ │ ├── component-name/

│ │ ├── feature.module.ts

│

├── environments

├── environment.dev.ts

├── environment.prod.ts