Module 6

What to learn

Passing Data from Parent to Child Component
Passing Data from Child to Parent Component using EventEmiter
Custom Pipes
Custom Directives

Practice Exercise

Practice 1
Do the hands on from the following link to understand the concept @input and @output decorator https://angular.io/start
Practice 2
Create a custom pipe named capitalize that capitalizes the first letter of a given string. Use it in a template to transform "hello world" to "Hello world".

Practice 3
Write a custom pipe called reverse to reverse a given string. For example, "Angular" should become "ralugnA".

Practice 4
Build a custom pipe named truncate to shorten a string to a specified length and append "..." if truncated. Pass the length as an argument.

Practice 5
Create a pipe named filter that filters an array of strings based on a search keyword. Demonstrate its use in an *ngFor loop.

Practice 6
Write a custom pipe called square that takes a number and returns its square. Use it in a template.

Practice 7
Develop a custom pipe named currencyConvert to convert an amount from one currency to another. Pass the conversion rate as an argument.

Practice 8
Create an orderBy pipe to sort an array of objects by a given property (e.g., sort by name or age).

Practice 9
Write a custom pipe evenFilter that filters an array of numbers and returns only even numbers. Use it in a component.

Practice 10
Implement a pipe named toDateString that converts a timestamp to a human-readable date format. Use it with a timestamp like 1633024800.

Practice 11
Build a pipe called mask that masks a credit card number, showing only the last 4 digits. For example, 1234567812345678 should become ************5678.

Practice 12
Create a Basic Directive

Create a directive named highlight that changes the background color of an element to yellow when it is applied.

Practice 13
Add Event Listener

Write a directive that listens for a click event on an element and logs "Element clicked!" to the console.

Practice 14
Change Text Color

Build a directive that changes the text color of an element to red.

Practice 15
Accept Input Property

Create a directive appBorder that accepts an input property for border color and applies it to the element.

Practice 16
Toggle CSS Class

Implement a directive that toggles a CSS class active on an element when clicked.

Practice 17
Show/Hide on Hover

Write a directive that shows an element on hover and hides it otherwise.

Practice 18
Conditional Style

Develop a directive that applies a green background if a boolean input property isValid is true, and red if false.

Practice 19
Log Mouse Coordinates

Write a directive that logs the x and y coordinates of the mouse whenever it moves over an element.

Practice 20
Dynamic Font Size

Create a directive that adjusts the font size of an element based on an input property fontSize.

Practice 21
Disable Button

Build a directive that disables a button if the input property isDisabled is true.

Practice 22
Change Content Dynamically

Create a directive that changes the text content of an element to "Hello World" on double-click.

Practice 23
Listen to Focus Event

Write a directive that listens for the focus event on an input field and changes its border to blue.

Practice 24
Style Multiple Elements

Implement a directive that applies a bold font weight to all child elements of a container.

Practice 25
Apply Gradient Background

Build a directive that applies a gradient background with two input colors to an element.

Practice 26
Restrict Input

Create a directive that restricts input to numeric characters only.

Practice 27
Highlight Active Element

Develop a directive that highlights the currently focused input field with a green border.

Practice 28
Scroll Listener

Write a directive that logs a message to the console when a user scrolls within a div.

Practice 29
Rotate Element

Create a directive that rotates an element by 45 degrees when clicked.

Practice 30
Animate Width

Build a directive that gradually increases the width of an element when hovered over.

Practice 31
Detect Outside Click

Develop a directive that detects clicks outside an element and logs a message to the console.

Assignment 1

Reference to the day10 Task. In the app component call Student Component and StudentList Component. Use Event Emitter to pass data from student component to the app component. From app component send data to studentList component using @Input decorator.

Assignment 2

Parent-to-Child Communication (@Input)

Use case: A parent component manages a list of products. The child component displays product details.

Question: Implement a parent component that passes a list of products to a child component using @Input. The child displays the details of the selected product.

Assignment 3

Two-Way Binding

Use case: A user updates their profile information using a form component.

Question: Create a form component with two-way binding to allow users to update their profile details (e.g., name, email). Reflect the changes in the parent component when the form is submitted.

Assignment 4

Service-Based Data Sharing

Use case: Two sibling components need to share a selected product's details.

Question: Use a shared service to enable data communication between two sibling components. Simulate the selection of a product in one component and display its details in the other.

Assignment 5

Dynamic Component Rendering

Use case: Display different types of notifications (success, error, info) in a notification container component.

Question: Implement a notification service and container that dynamically renders notification components based on type.

Assignment 6

Dynamic Component Interaction with @ViewChild

Use case: A parent component dynamically adds and interacts with multiple child components.

Question: Create a parent component that dynamically adds child components to a list. Use @ViewChild to call a method (e.g., highlight()) on a selected child component.

Assignment 7

Parent Controlling Child Visibility

Use case: A parent component shows or hides a child component based on user actions.

Question: Create a child component that displays a message. Use @Input in the child to control its visibility from the parent.

Assignment 8

Child Passing Events to Parent

Use case: A child component contains a button to mark a task as completed.
Question: Implement a task child component that uses @Output to notify the parent component when a task is marked as completed.

Assignment 9

Synchronizing Input and Output

Use case: A parent component controls the state of a child toggle component.
Question: Create a toggle component that uses @Input for the initial state and @Output to notify the parent of state changes when toggled.

Online Reference

https://angular.io/api/core/EventEmitter

https://dzone.com/articles/understanding-output-and-eventemitter-in-angular

https://dzone.com/articles/understanding-output-and-eventemitter-in-angular

https://angular.io/api/core/ViewChild