Angular | Lecture 2

Marina Magdy

Agenda

- Recap last lecture topics
- UI Libraries
- Directives
- Sharing data between components
- Pipes

UI Libraries

Angular Material

https://material.anghttps://material.angular.io/ular.io/

Ng Bootstrap

https://ng-bootstrap.github.io/#/getting-started

Primeng

https://primeng.org/installation

Directives

Directives

Directives are classes that add additional behavior to elements in your Angular applications.

https://angular.dev/guide/directives

Directives are kind of instructions to the DOM , Directives are components without a view. They are components without a template. Or to put it another way, components are directives with a view.

There’s different types of directives :

- Component Directives
- Structural Directives
- Attribute Directives

Directives

Directives are kind of instructions to the DOM.

There’s different types of directives :

- Component Directives - directives with a template.
- Structural Directives - change the DOM layout by adding and removing DOM elements.
- Attribute Directives - change the appearance or behavior of an element, component, or another directive.

Directives

Structural directives :

Structural directives are responsible for HTML layout. They shape or reshape the DOM's structure, typically by adding, removing, or manipulating elements.

- NgIf
- NgFor
- NgSwitch [Self-Study]

Directives

NgFor (Old versions before v17)

If using standalone components you need to import commonModule in the imports array of component.

```
<li *ngFor="let person of people; let i = index"> (1)
   {{ i + 1 }} - {{ person.name }} (2)
 </li>
```

Directives

With built in @for

The @for block renders its content in response to changes in a collection. Collections can be any JavaScript iterable, You can optionally include an @empty section immediately after the @for block content. The content of the @empty block displays when there are no items.

```
@for (user of users; track user.id) {
  {{ user.name }}
} @empty {
  Empty list of users
}
```

Directives

Track key

- When Angular renders a list of elements with @for, those items can later change or move. Angular needs to track each element through any reordering, usually by treating a property of the item as a unique identifier or key.

- This ensures any updates to the list are reflected correctly in the UI and tracked properly within Angular

- Loops over immutable data without trackBy as one of the most common causes for performance issues across Angular applications. Because of the potential for poor performance, the track expression is required for the @for loops. When in doubt, using track $index is a good default.

Directives

NgIf (Old versions before v17)

The NgIf directive is used when you want to display or remove an element based on a condition.

If the condition is false the element the directive is attached to will be removed from the DOM.

```
<div *ngIf="loggedIn; else anonymousUser">
  The user is logged in
</div>
<ng-template #anonymousUser>
  The user is not logged in
</ng-template>
```

Directives

With built in @if

```
@if (loggedIn) {
  The user is logged in
} @else {
  The user is not logged in
}
```

*ngIf then and else (legacy before v17)

<div *ngIf="condition; then thenBlock else elseBlock">

<ng-template #thenBlock>Content to render when condition is true.</ng-template>

<ng-template #elseBlock>Content to render when condition is false.</ng-template>

More info for *ngIf

https://angular.io/api/common/NgIf

Directives

Attribute directives

You should add CommonModule to imports array or import NgClass and NgStyle separated in imports.

- **ngClass :** It changes the class attribute that is bound to the component or element it's attached to.

  <div [ngClass]="isSpecial ? 'special' : ''">This div is special

- **ngStyle :** It's used to modify or change the element's style attribute. This attribute directive is quite similar to using style metadata in the component class.

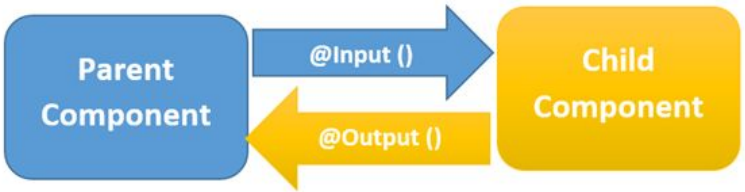
  <div [ngStyle]="{'background-color': isSpecial? 'green' : 'red' }">This div is special</

Sharing Data between Components

Sharing data between components

Parent to child :

This is probably the most common and straightforward method of sharing data. It works by using the @Input() decorator to allow data to be passed via the template.



Sharing data between components

Child to parent

Using **output and event emitter** is a way to share data is to emit data from the child, which can be listed to by the parent. This approach is ideal when you want to share data changes that occur on things like button clicks, form entires, and other user events.

Example :

- `<app-child (messageEvent)="receiveMessage($event)"></app-child> - parent`
- `receiveMessage($event) {this.message = $event}- parent.ts`
- `@Output() messageEvent = new EventEmitter<string>();- Child.ts`
- `sendMessage() {this.messageEvent.emit(this.message)} - Child.ts`
- `<button (click)="sendMessage()">Send Message</button> - Child.html`

Sharing data between components

Child to parent @ViewChild [Self Study]

Another way to share from child to parent using **ViewChild** that allows a one component to be injected into another, giving the parent access to its attributes and functions. One caveat, however, is that child won't be available **until after the view has been initialized.** This means we need to implement the AfterViewInit lifecycle hook to receive the data from the child.

```
@ViewChild(ChildComponent) child;
```

Sharing data between components

Unrelated Components

- When passing data between components that lack a direct connection, such as siblings, grandchildren, etc, you should you a shared service.
- And you can also create a service to set and get values across unrelated components.

Will be covered in details later with services

Pipes

Pipes

Pipes are simple functions you can use in template expressions to accept an input value and return a transformed value.

You should import `DatePipe` or any used pipe in imports array : Link

Angular provides built-in pipes for typical data transformations like :

- DatePipe: Formats a date value according to locale rules.
  - `{{ today | date : 'MMMM YYYY' }}`
- UpperCasePipe: Transforms text to all upper case.
  - `{{ name | uppercase }}`
- LowerCasePipe: Transforms text to all lower case.

For More Info : https://v17.angular.io/guide/pipes-overview

Custom Pipes [Extra]

- To generate custom pipe you need to run :
  - **`ng generate pipe pipeName`**

For example you can generate custom pipe that transform file size to MB :

```
transform(value: any, ...args: any[]): unknown {
   return (value / (1024 * 1024)).toFixed(2) + 'MB';
 }
```

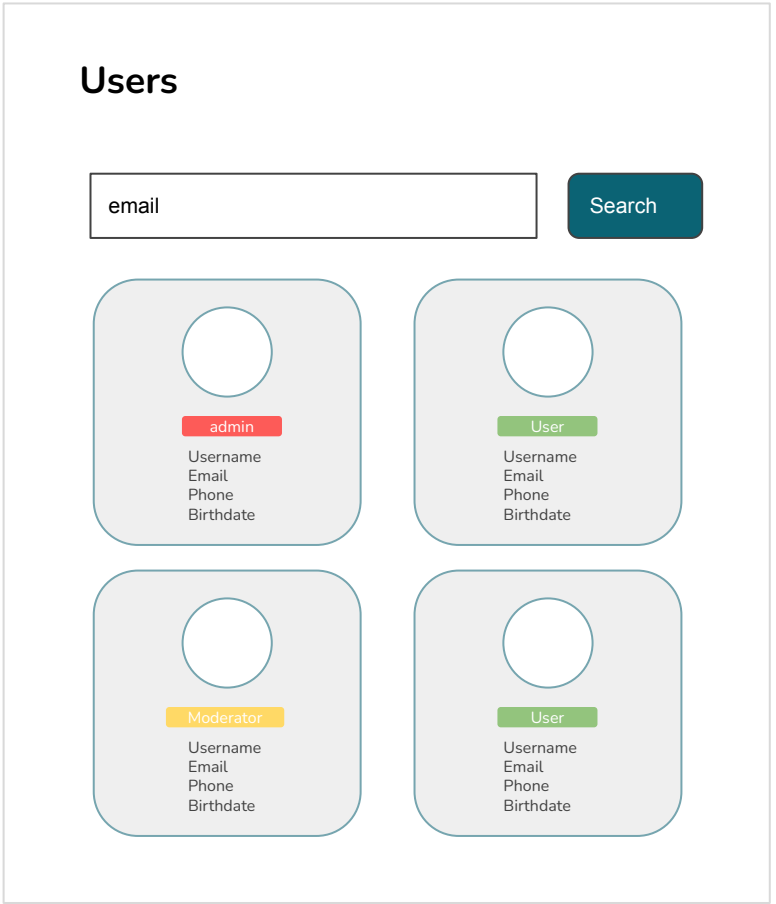
Thank you

Lap

Users list

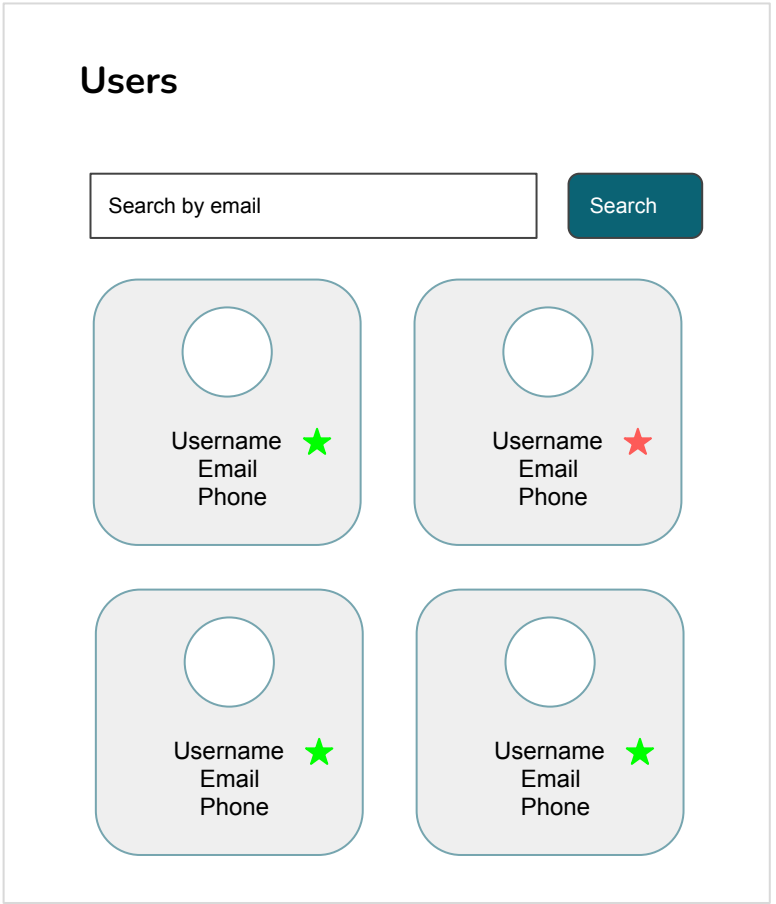
- Using the provided users list to render the users cards with the following:
  - Profile picture
  - Username
  - Email
  - Phone number
  - Birthdate (format Pipe)
  - Role chip

- Based on the user role show a chip with different color. If admin show chip with red, if user show with green, if moderator show chip with yellow



Users list

- [Bonus] Also you can search and in the users list by Email and After user search, reset button will appear when click will reset the fields and show all users again.
- [Bouns] How to read data from JSON file



To do app

Create a to do app to create self notes with the following features:

- Red is Parent Component for the To-Do (Array of todos)
- Yellow borders are child components one for the input with add button [which will emit the input value to the parent component]
- Other yellow border is for the other child component of the items list]
- User can add new task
- User can delete Task
- User can mark as completed and when mark as completed will be marked with linethrough.[Bonus]