Project 1: EASYTASK

# Concepts Used in Angular Project

## Angular Component Concepts

### Component Declaration

Components are fundamental building blocks of Angular applications. They consist of three main parts: the template (HTML), the style (CSS), and the logic (TypeScript). Examples: HeaderComponent, CardComponent, TaskComponent, NewTaskComponent, UserComponent, and TasksComponent.

### Component Selector

The selector property in the component decorator defines the custom HTML tag representing the component. For example, <app-header> for the HeaderComponent.

### Template and Style URLs

The templateUrl and styleUrls properties point to the HTML and CSS files associated with the component, respectively.

### Input and Output Properties

@Input is used to pass data from a parent component to a child component. @Output is used to send data from a child component to a parent component using the EventEmitter class.

### Event Binding

Event binding is used to listen to events such as clicks and form submissions. For example, (click)="onSelectUser()" binds a click event to the onSelectUser method.

### ngContent

ngContent is used to project content into a component. This allows a component to render content passed from the parent component.

## Angular Modules

### NgModule

Angular modules are used to organize an application into cohesive blocks of functionality. They contain declarations of components, services, and other modules. Example: SharedModule, TasksModule, AppModule.

### CommonModule

This module contains common directives like ngIf and ngFor. It is imported to use these directives in components.

### FormsModule

This module is required for template-driven forms in Angular. It provides directives like ngModel.

## Angular Services

### Dependency Injection

Angular services are singleton objects that can be injected into components or other services using Angular's dependency injection system. Example: TasksService is injected into components to manage tasks.

### Local Storage

The TasksService uses local storage to persist data across sessions. This is done using localStorage.getItem and localStorage.setItem.

## Angular Directives and Pipes

### ngFor Directive

This directive is used for iterating over a list of items and rendering them in the template.

### ngIf Directive

This directive is used for conditionally rendering elements in the template based on a boolean expression.

### Date Pipe

Angular provides built-in pipes for transforming data. The date pipe is used to format dates in the template.

## Forms and Form Handling

### Template-Driven Forms

Forms are defined in the template and are bound to data models using ngModel.

### Form Validation

Angular provides directives for form validation. Form elements can be marked as required, and custom validation can be added.

### Form Submission

The ngSubmit directive is used to handle form submissions.

## CSS and Styling

### Flexbox Layout

CSS Flexbox is used for creating flexible and responsive layouts. Properties like display: flex, flex-direction, and justify-content are used.

### Media Queries

Media queries are used to apply different styles based on screen size, making the application responsive.

### CSS Box Shadow and Border Radius

These properties are used to enhance the visual appearance of components by adding shadows and rounded corners.

## Application Structure

### Root Component

The root component (AppComponent) is the main entry point of the application. It includes the header and other main components.

### Shared Module

The SharedModule contains shared components like CardComponent that can be reused across the application.

### Tasks Module

The TasksModule handles task-related functionality, including components for displaying and managing tasks.

### User Management

Users are displayed in a list, and selecting a user displays their tasks. This is managed using UserComponent and TasksComponent.

## Concepts in Practice

### Data Binding

One-way data binding is used to display data in the template. Two-way data binding ([(ngModel)]) is used to bind form inputs to component properties.

### Event Emission

Custom events are emitted from child components to parent components to handle actions like closing a form or selecting a user.

### Component Interaction

Parent-child component interaction is achieved through @Input and @Output properties.

## Angular Concepts Notes

### Property Binding

Property Binding involves binding an HTML element property to a component property. It allows the application to dynamically update the values of HTML element properties.

### Event Binding

Event Binding allows the application to listen and respond to user actions such as clicks, key presses, and other events. It binds an event to a method in the component class.

### Two-way Binding with [(ngModel)]

Two-way binding combines property and event binding, allowing the application to share data between the component class and the template. The [(ngModel)] directive is used for two-way data binding in forms.

### Form Submission with (ngSubmit)

The (ngSubmit) directive binds a method in the component class to the form's submit event. It allows the application to handle form submissions using Angular's Reactive Forms or Template-driven Forms.

### Services for CRUD Operations

Services in Angular provide a way to share data and logic across multiple components. They are typically used for performing CRUD (Create, Read, Update, Delete) operations by interacting with a backend API.

### @Injectable for Data Sharing

The @Injectable decorator is used to define a service class that can be injected into other components or services. It allows the application to share data and functionality across different parts of the application.

## Binding Concepts

### Property Binding

This is used to communicate between the parent and child components. The @Input decorator is used in the child component to receive data from the parent component.

### Event Binding

This allows the child component to communicate events to the parent component using the @Output decorator and EventEmitter in the child component.

### Two-way Binding

The [(ngModel)] directive facilitates two-way data binding. It binds the form input to a variable in the component class. For example, an empty string in the TypeScript file can be bound to an input element using [(ngModel)], enabling automatic synchronization of the variable with the input value.

## Form Handling

### (ngSubmit) Directive

This directive is used to handle form submission. It is placed at the beginning of the form tag and binds a method in the component to the form's submit event, enabling form handling logic to be executed upon submission.

## Services

### Services

Services are essential for performing CRUD operations. They are standalone files that provide methods for data operations and can be shared across components.

### @Injectable({ providedIn: 'root' })

This decorator is used to provide a service at the root level, making it available for dependency injection throughout the application.

## Application Initialization

### Constructor

The constructor is used to initialize values and handle dependency injection when the component is instantiated, which typically occurs at the beginning of the webpage loading.

### main.ts

This is the entry point of the Angular application. It bootstraps the root module (AppModule) and is the first file to run when the application starts.

## Angular Modules (NgModule)

### Using Angular Modules

While standalone components are recommended, Angular applications can also be organized using modules. This involves defining components, services, and other elements within a module file (e.g., app.module.ts).

### app.module.ts

This file declares the AppComponent and imports other modules or components whose standalone property is set to TRUE.

### BrowserModule

Essential for running Angular applications in the browser. It includes built-in pipes like DatePipe, so there's no need for separate imports for date manipulation.

### Bootstrap in Modules

The root module (AppModule) integrates the bootstrap component, while child modules do not.

### Exporting Components

In tasks.module, only the TasksComponent is exported, indicating it is used in other components or modules.