



# Angular Components



# Contents

Creating an  
Angular  
Component

What are  
Angular  
Components?

Component  
Decorator  
metadata

Importing what  
we Needed



# Significance and Role of Components



UI Building Blocks

Encapsulation and Reusability

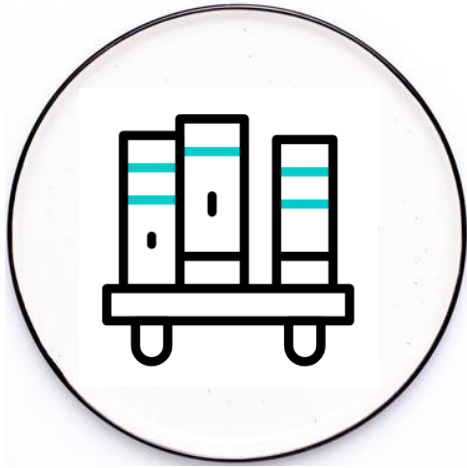
Component-Based Architecture

Component Lifecycle Hooks

Hierarchical Structure



# What Are Angular Components?



In Angular ,  
"everything is a  
components."



Component is a primary  
building block of an  
angular application.



Components are  
reusable, modular, and  
encapsulated

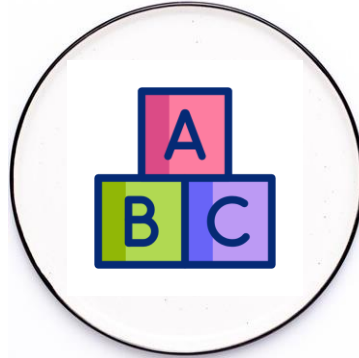


Only one component can be instantiated per element in a template.

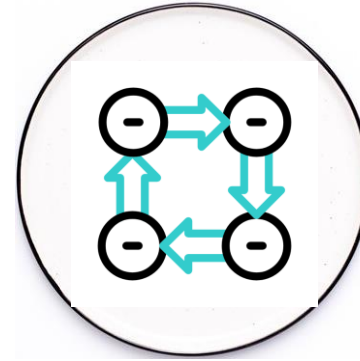
The component



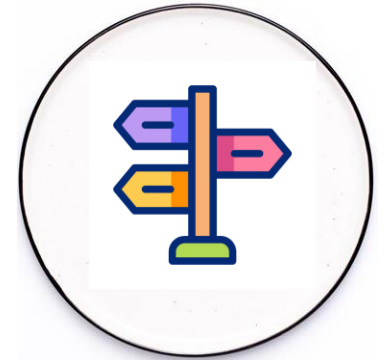
Component must belong to the ng module.



@component-decorator provides additional metadata.



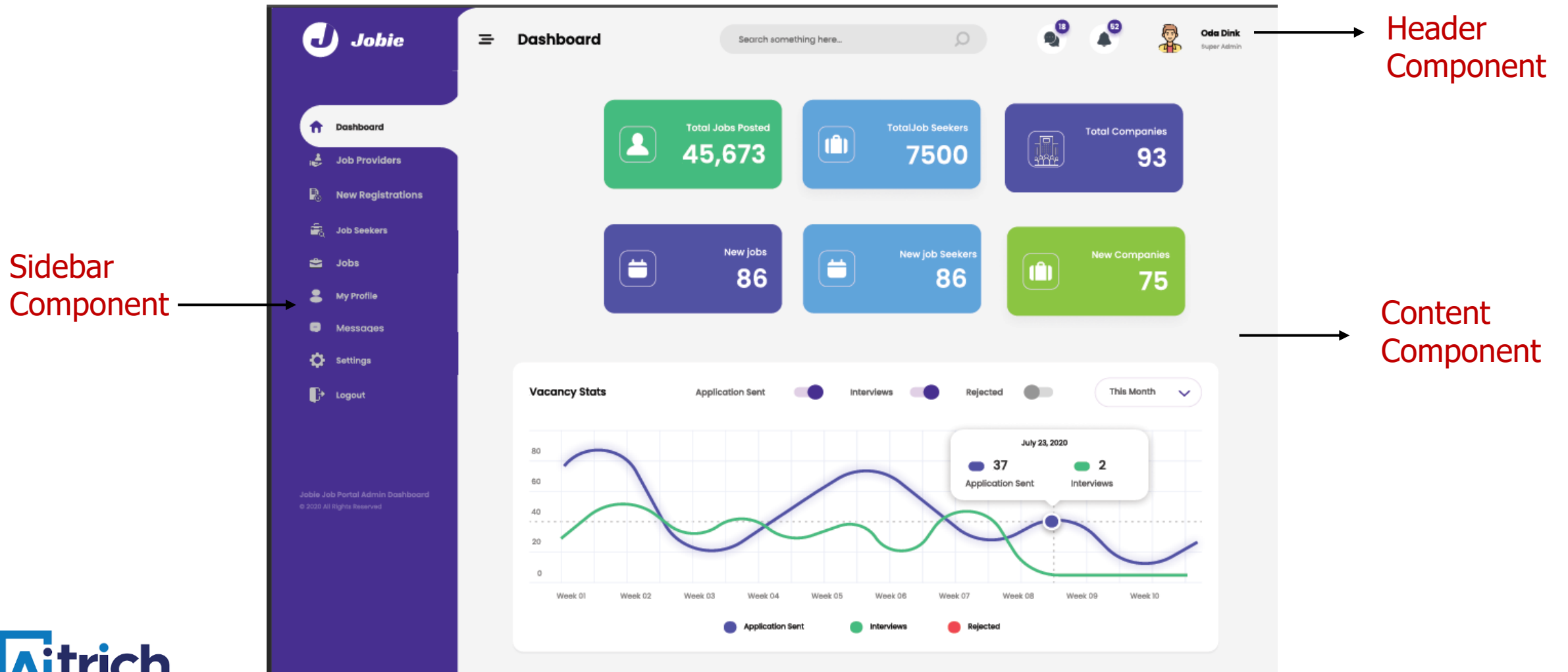
Components implements life cycle hooks.



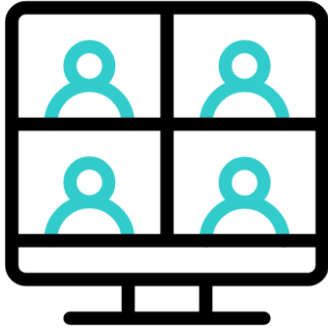
Angular components are a subset of directives.



# Example



# Component Structure



## Template

HTML code that defines the component's UI



## Class

TypeScript code that handles the component's logic and data.



## Metadata

Information about the component, such as selector, styles, etc.



# Creating Angular Component



To create a component, use the Angular CLI (Command Line Interface) or create the files manually.

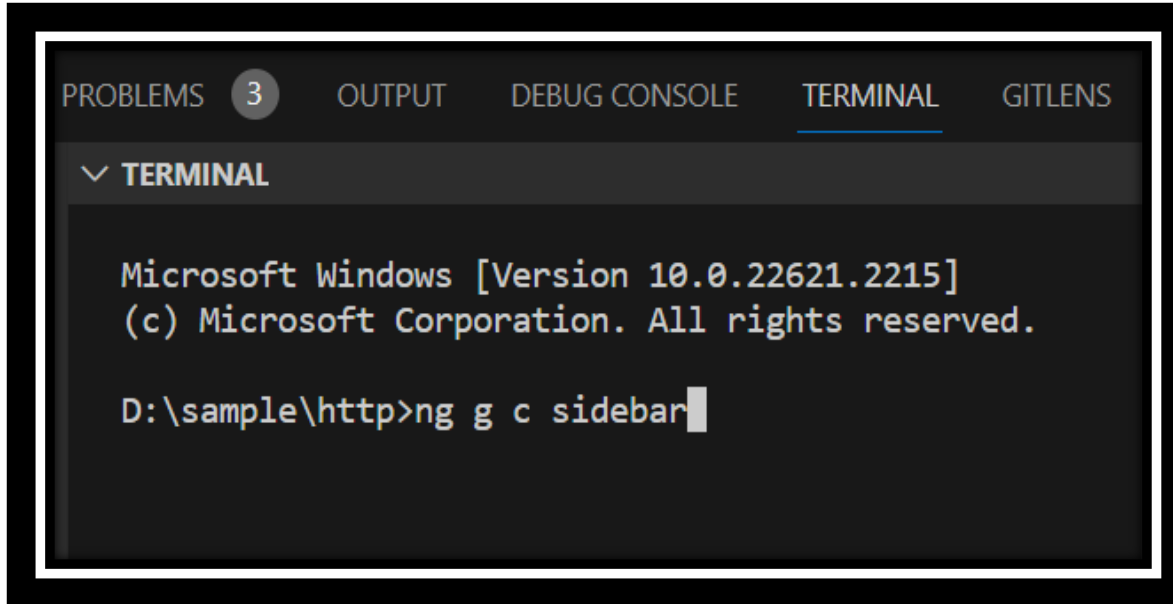
This generates the necessary files for the component: HTML, CSS, TypeScript, and a spec file.

The command for creating a component using the Angular CLI: **ng generate component component-name**





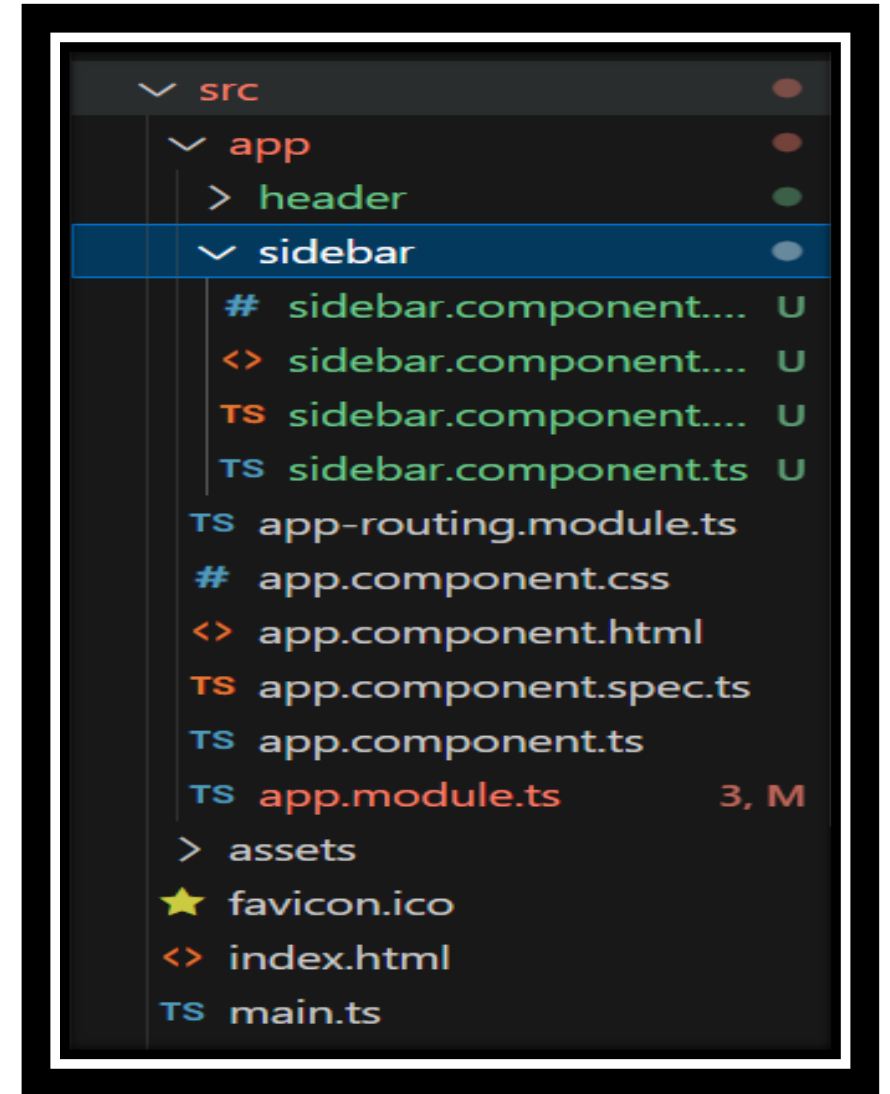
# Example



A screenshot of a Windows terminal window. The title bar shows 'PROBLEMS 3 OUTPUT DEBUG CONSOLE TERMINAL GITLENS'. The 'TERMINAL' tab is active. The terminal text reads: 'Microsoft Windows [Version 10.0.22621.2215] (c) Microsoft Corporation. All rights reserved. D:\sample\http>ng g c sidebar'.

```
Microsoft Windows [Version 10.0.22621.2215]
(c) Microsoft Corporation. All rights reserved.

D:\sample\http>ng g c sidebar
```



A screenshot of a file explorer showing the project structure. The 'src' folder is expanded, showing 'app', 'header', and 'sidebar'. The 'sidebar' folder is selected and expanded, showing files: 'sidebar.component...', 'sidebar.component...', 'sidebar.component...', 'sidebar.component.ts', 'app-routing.module.ts', 'app.component.css', 'app.component.html', 'app.component.spec.ts', 'app.component.ts', 'app.module.ts' (3, M), 'assets', 'favicon.ico', 'index.html', and 'main.ts'.

```
src
├── app
│   ├── header
│   └── sidebar
│       ├── sidebar.component... U
│       ├── sidebar.component... U
│       ├── sidebar.component... U
│       ├── sidebar.component.ts U
│       ├── app-routing.module.ts
│       ├── app.component.css
│       ├── app.component.html
│       ├── app.component.spec.ts
│       ├── app.component.ts
│       ├── app.module.ts 3, M
│       ├── assets
│       ├── favicon.ico
│       ├── index.html
│       └── main.ts
```



# Component Decorator : Metadata

- ❖ Metadata is defined using the @Component decorator.
- ❖ It provides information about the component such as selector, template URL, style URLs, etc.
- ❖ Decorator always start with @ symbol.
- ❖ Selector is used to create an instance of the component where it finds <my-app> tag in parent HTML, that is index.html file.
- ❖ Template tells Angular how to display the component.

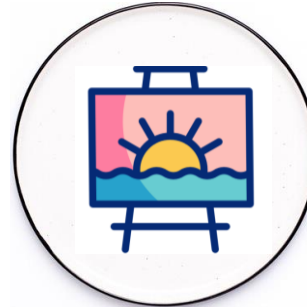




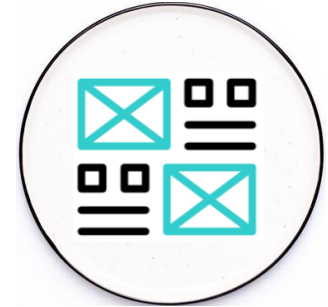
Selector



Template



Template URL



Styles



Providers



Animations



Style URLs



# Example

```
src > app > second-header > TS second-header.component.ts > ...
```

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-second-header',
5    templateUrl: './second-header.component.html',
6    styleUrls: ['./second-header.component.css']
7  })
8  export class SecondHeaderComponent {
9
10 }
11
```



# Template Syntax

Templates can be defined in two ways

## 1. Inline Template

Created within the component file itself using the `template` property. Suitable for small templates or when the HTML is simple.

```
typescript

@Component({
  selector: 'app-example',
  template: '<h1>Inline Template</h1>'
})
```

## 2. External Template

Defined in a separate HTML file. Ideal for larger templates or when you want to maintain a separation of concerns.

```
typescript

@Component({
  selector: 'app-example',
  templateUrl: './example.component.html'
})
```



# Component Class

The component's class is written in TypeScript.

It contains the logic and data for the component.

Properties and methods are defined in the class..



# Example

```
export class ExampleComponent {  
  title: string = 'Example Component';  
  count: number = 0;  
  
  increaseCount() {  
    this.count++;  
  }  
}
```



# Component Lifecycle Hooks

Angular provides lifecycle hooks to perform actions at different stages of a component's life.

**Examples:** `ngOnInit`, `ngOnDestroy`, `ngOnChanges`, etc.

These hooks allow you to initialize data, make API calls, handle subscriptions, and more.

To use a component, you can include its selector in another component's template.

```
<app-example> </app-example>
```





# Summary



- ❖ Angular components are essential for building Angular applications.
- ❖ They consist of a template, class, and metadata.
- ❖ Components are reusable, modular, and encapsulated.
- ❖ They handle the UI and user interactions.
- ❖ Component lifecycle hooks provide additional control over a component's behavior.





# Questions?