

Creating a Component



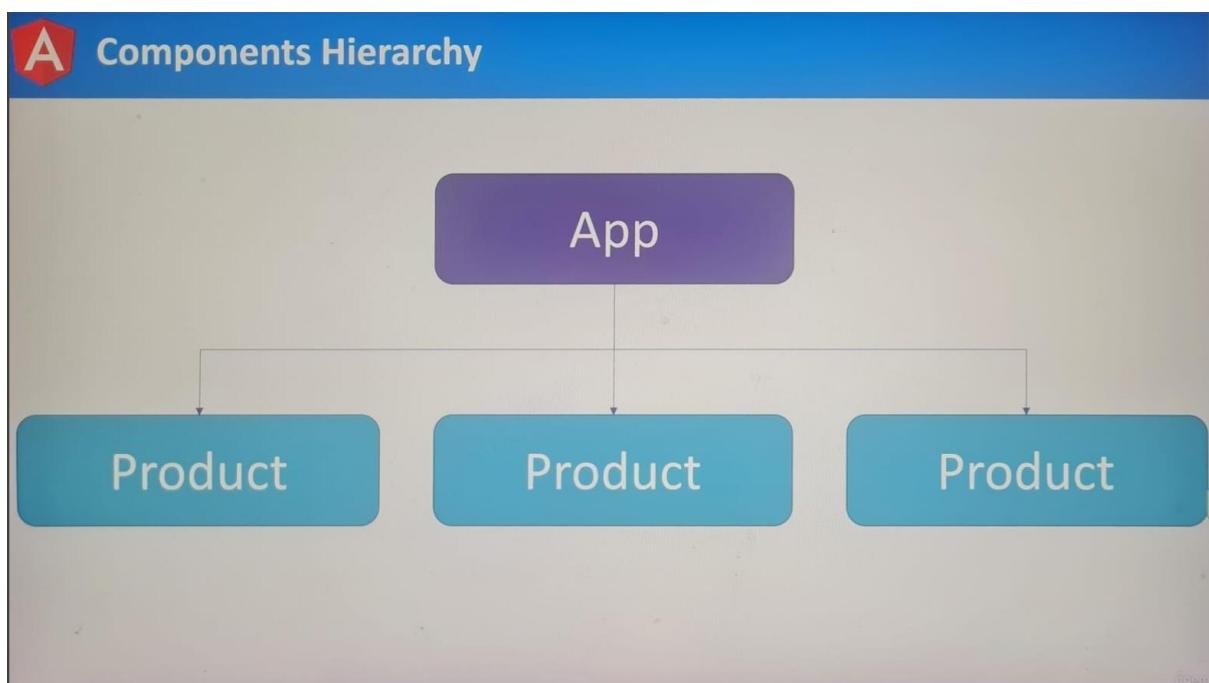
It's recommended to break an application into several components.

Components are a feature for teaching the browser new HTML tags. Therefore, Components can be structured like a tree, it's similar to the DOM tree.

The app component is typically the root or parent component of all components.

The job of the app component is to load other components.

In this diagram:



We have a hierarchy of components where the app component is the parent.

The product components are the children of the app component. Children components can load more components, there isn't a limit on the depth of the tree, it's similar to HTML, or we can have a deeply nested structure of HTML tags.

Typically, components should focus on one feature.

In our example, the product component would never render information about users. It should focus strictly on products.

We have two options for creating components:

We can create them manually or use the CLI.

We've already seen how to create components manually.

We can create a component with the **ng generate component** command, and in the command line we have the app running we can open a new command line by pressing the plus button, I recommend having two command lines open for the rest of this course, one for running the server, another for installing packages or creating files.

ng generate component --help

```
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code\basics> ng generate component --help ←
ng generate component [name]

Creates a new Angular component. Components are the basic building blocks of Angular applications. Each component consists of a TypeScript class, an HTML template, and an optional CSS stylesheet. Use this schematic to generate a new component in your project.

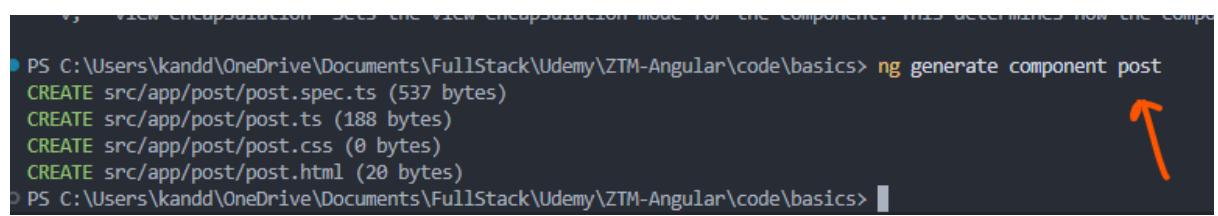
Arguments:
  name  The name for the new component. This will be used to create the component's class, template, and stylesheet files. For example, if you provide `my-component`, the files
        will be named `my-component.ts`, `my-component.html`, and `my-component.css`. [string]

Options:
  --help          Shows a help message for this command in the console. [boolean]
  --interactive   Enable interactive input prompts. [boolean] [default: true]
  -d, --dry-run    Run through and reports activity without writing out results. [boolean] [default: false]
  --defaults     Disable interactive input prompts for options with a default. [boolean] [default: false]
  --force         Force overwriting of existing files. [boolean] [default: false]
  -c, --change-detection Configures the change detection strategy for the component. [string] [choices: "Default", "OnPush"] [default: "Default"]
  -b, --display-block Adds ':host { display: block; }' to the component's stylesheet, ensuring the component renders as a block-level element. This is useful for layout
        purposes. [boolean] [default: false]
  --export        Automatically export the component from the specified NgModule, making it accessible to other modules in the application. [boolean] [default: false]
  --export-default Use a default export for the component in its TypeScript file instead of a named export. [boolean] [default: false]
  --flat          Create the component files directly in the project's `src/app` directory instead of creating a new folder for them. [boolean] [default: false]
  -s, --inline-style Include the component's styles directly in the `component.ts` file. By default, a separate stylesheet file (e.g., `my-component.css`) is created.
        [boolean] [default: false]
  -t, --inline-template Include the component's HTML template directly in the `component.ts` file. By default, a separate template file (e.g., `my-component.html`) is
        created. [boolean] [default: false]
  -m, --module      Specify the NgModule where the component should be declared. If not provided, the CLI will attempt to find the closest NgModule in the component's
        path. [string]
  --ng-html        Generate component template files with an '.ng.html' file extension instead of '.html'. [boolean] [default: false]
  -p, --prefix      A prefix to be added to the component's selector. For example, if the prefix is `app` and the component name is `my-component`, the selector will be
        `app-my-component`. [string]
  --project        The name of the project where the component should be added. If not specified, the CLI will determine the project from the current directory.
        [string]
  --selector       The HTML selector to use for this component. If not provided, a selector will be generated based on the component name (e.g., `app-my-component`). [string]
  --skip-import    Do not automatically import the new component into its closest NgModule. [boolean] [default: false]
  --skip-selector  Skip the generation of an HTML selector for the component. [boolean] [default: false]
  --skip-tests     Skip the generation of unit test files `spec.ts` . [boolean] [default: false]
  --standalone     Generate a standalone component. Standalone components are self-contained and don't need to be declared in an NgModule. They can be used
        independently or imported directly into other standalone components. [boolean] [default: true]
  --style          Specify the type of stylesheet to be created for the component, or `none` to skip creating a stylesheet. [string]
```

```
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code\basics> ng generate component --help
ng generate component [name]
```

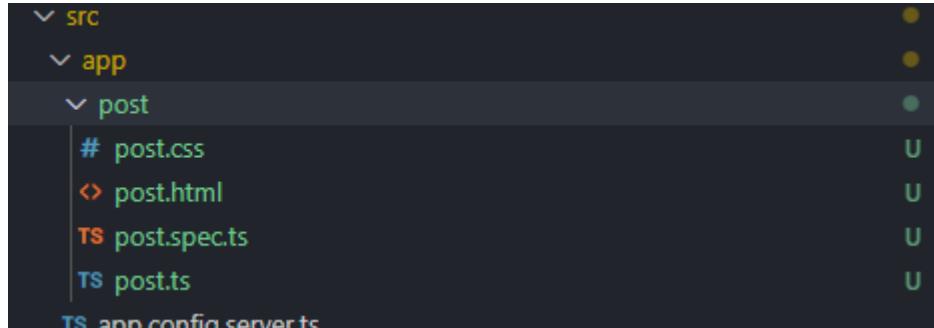
We will run the following command: **ng generate component post**

After a few moments the CLI will give us a list of files that were created or updated.



```
v, --viewEncapsulation=sets the view encapsulation mode for the component. This determines how the compo
● PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code\basics> ng generate component post
CREATE src/app/post/post.spec.ts (537 bytes)
CREATE src/app/post/post.ts (188 bytes)
CREATE src/app/post/post.css (0 bytes)
CREATE src/app/post/post.html (20 bytes)
● PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code\basics>
```

Inside the src/app/ directory, we have a new folder called post:

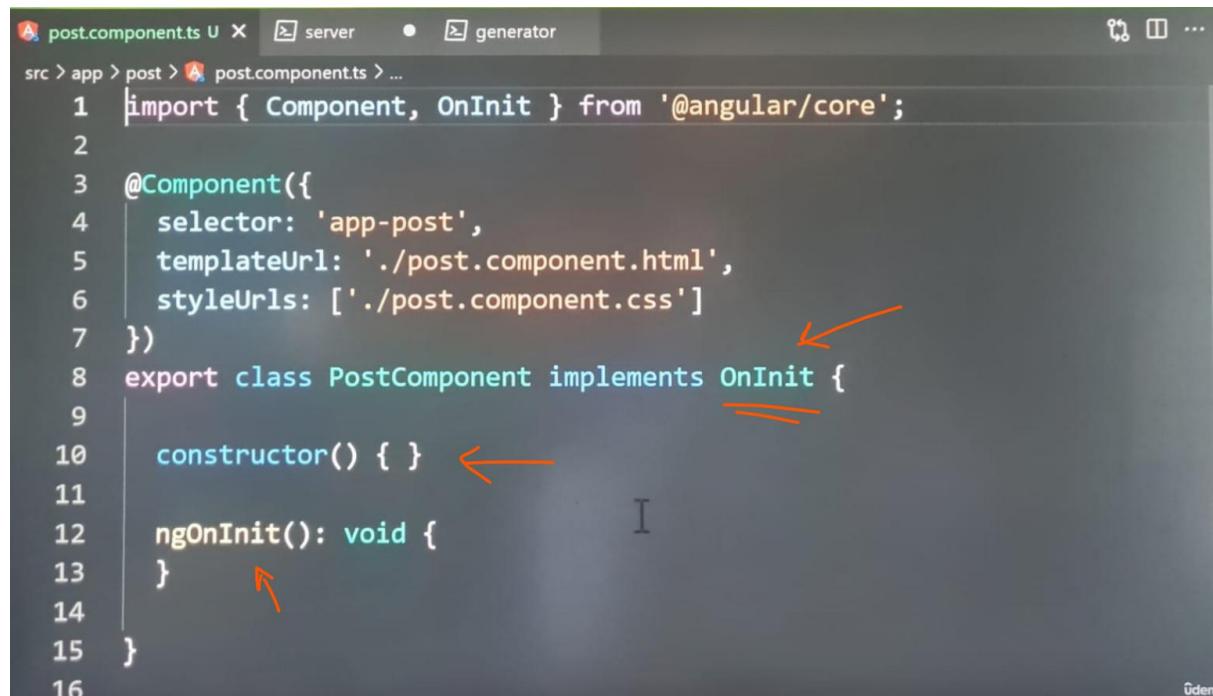


Components should be placed inside a separate directory; it's a common practice we adopt.

Angular will generate a class, template, stylesheet and test files.

The stylesheet and template will be very basic; there's nothing interesting happening in these files. As for the test file, we're going to ignore it. Testing is not a topic covered in this course.

Let's check out the class file:



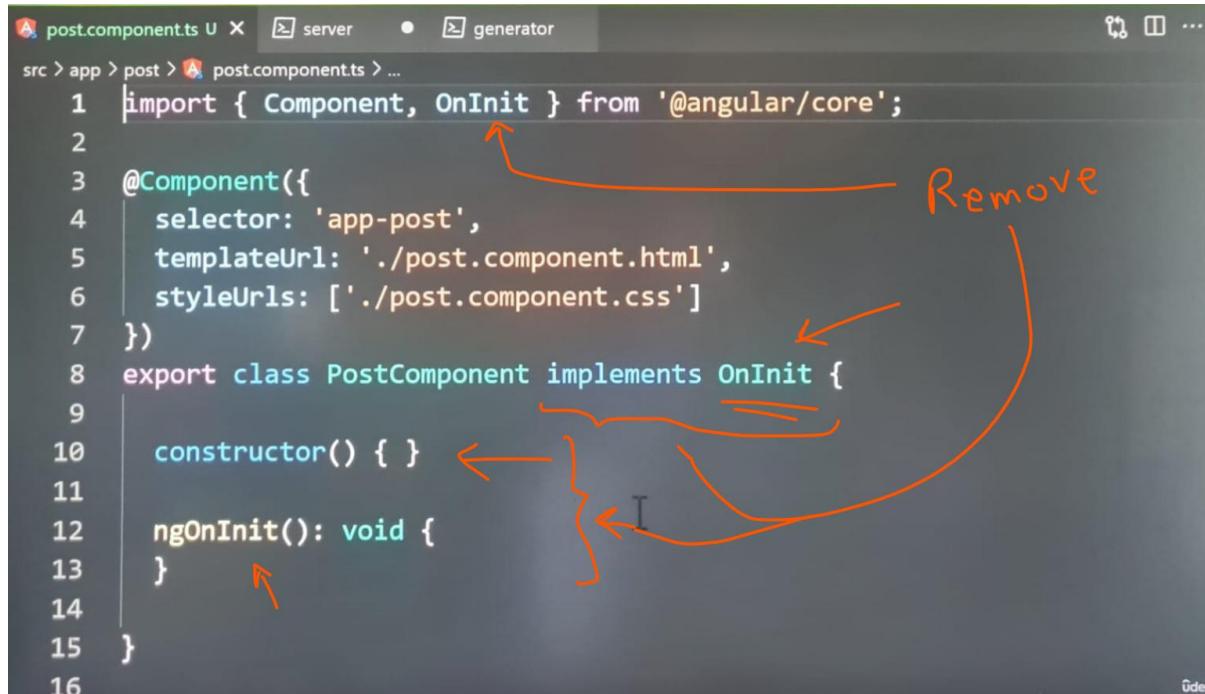
```
post.component.ts U X server generator ...
src > app > post > post.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-post',
5   templateUrl: './post.component.html',
6   styleUrls: ['./post.component.css']
7 })
8 export class PostComponent implements OnInit {
9
10   constructor() {} ←
11
12   ngOnInit(): void { ←
13   }
14
15 }
```

On my PC running latest version of Angular 20.0.5 – post.ts:

```
TS post.ts U X
basics > src > app > post > TS post.ts > ...
1 import { Component } from '@angular/core';
2
3 @Component({
4   selector: 'app-post',
5   imports: [],
6   templateUrl: './post.html',
7   styleUrls: ['./post.css'
8 })
9 export class Post {
10
11 }
12
```

It has the same settings as app component with some exceptions, firstly, the class has a **constructor** function and **ngOnInit** method.

The **constructor** function is commonly used for initializing properties, for this demonstration we're going to remove it from the class, it's not a function we will be using, as for the **ngOnInit** method, we're going to remove it as well as the **implements** keyword and **ngOnInit** object from the import statement:



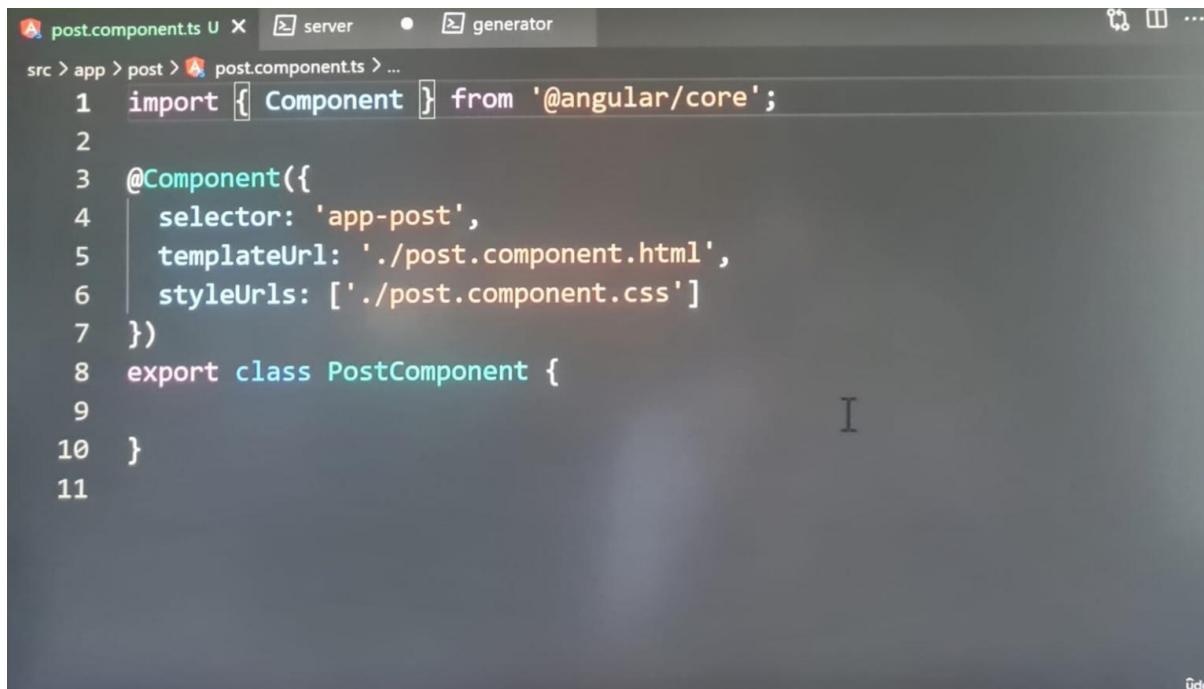
The screenshot shows a code editor with a file named `post.component.ts`. The code is as follows:

```
src > app > post > post.component.ts > ...
1 import { Component, OnInit } from '@angular/core';
2
3 @Component({
4   selector: 'app-post',
5   templateUrl: './post.component.html',
6   styleUrls: ['./post.component.css']
7 })
8 export class PostComponent implements OnInit {
9   constructor() { }
10  ngOnInit(): void {
11    }
12  }
13
14 }
```

Annotations in red highlight parts of the code for removal:

- An arrow points to the `import { Component, OnInit } from '@angular/core';` line with the text "Remove".
- A curly brace groups the `constructor()` and `ngOnInit()` lines, with an arrow pointing to the brace and the text "Remove".
- A small red arrow points to the closing brace of the class definition at line 14.

I want to start with a minimal component:



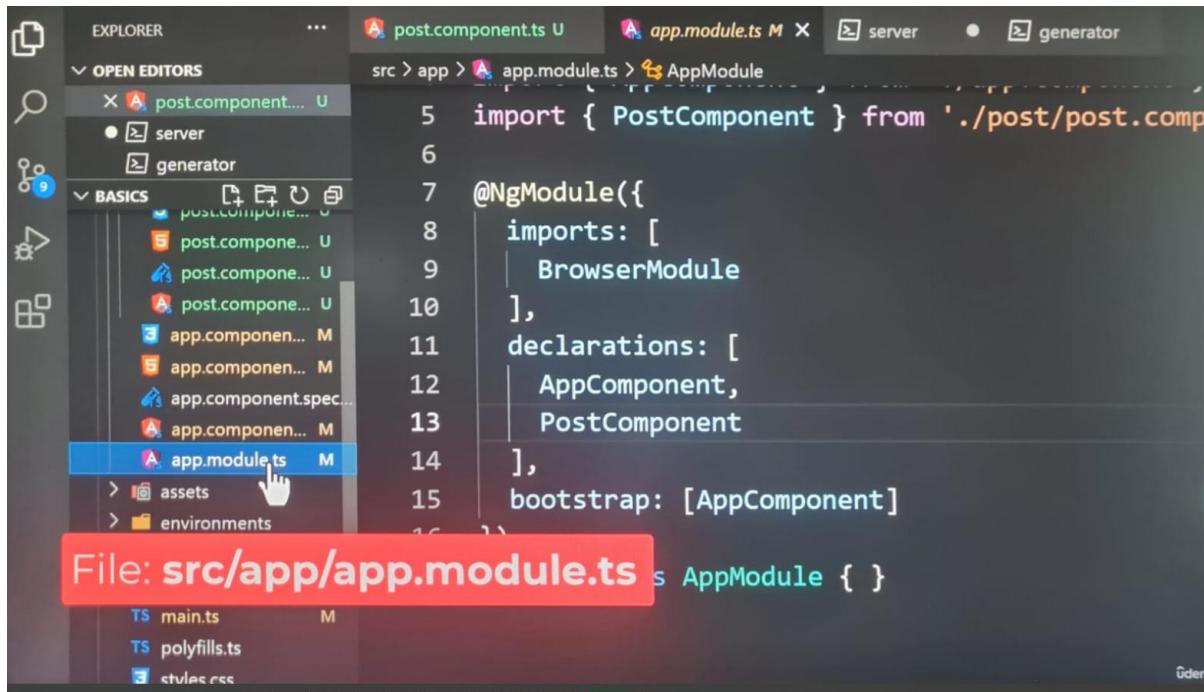
```

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

```

These extra pieces of code will be revisited in a future lecture. Other than that, everything in this file should be familiar to you. Typically, we would need to register the component. Luckily, we won't have to. The CLI has done so for us.

Let's check out the app module file:



The screenshot shows the VS Code interface with the Explorer sidebar open, displaying files like post.component.ts, app.module.ts, and main.ts. The app.module.ts file is the active editor, showing the following code:

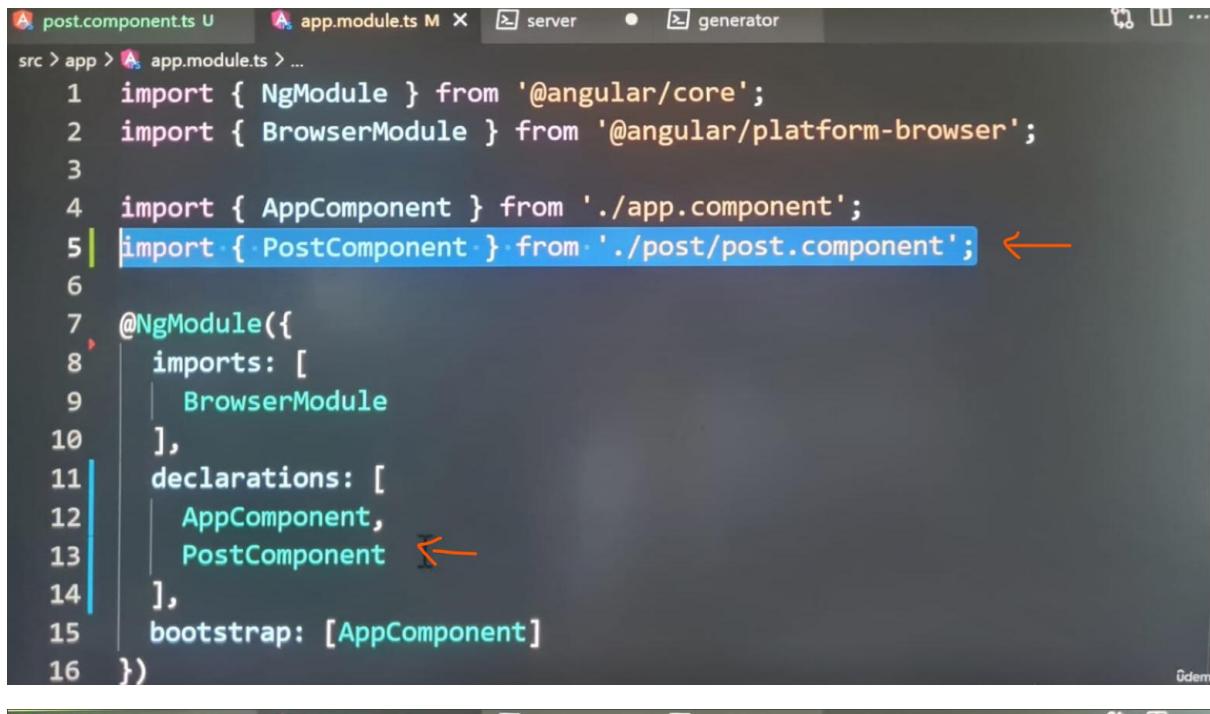
```

src > app > app.module.ts > AppModule
5 import { PostComponent } from './post/post.comp...
6
7 @NgModule({
8   imports: [
9     BrowserModule
10 ],
11 declarations: [
12   AppComponent,
13   PostComponent
14 ],
15 bootstrap: [AppComponent]
16 })

```

A red box highlights the file path "File: src/app/app.module.ts" at the bottom of the editor area.

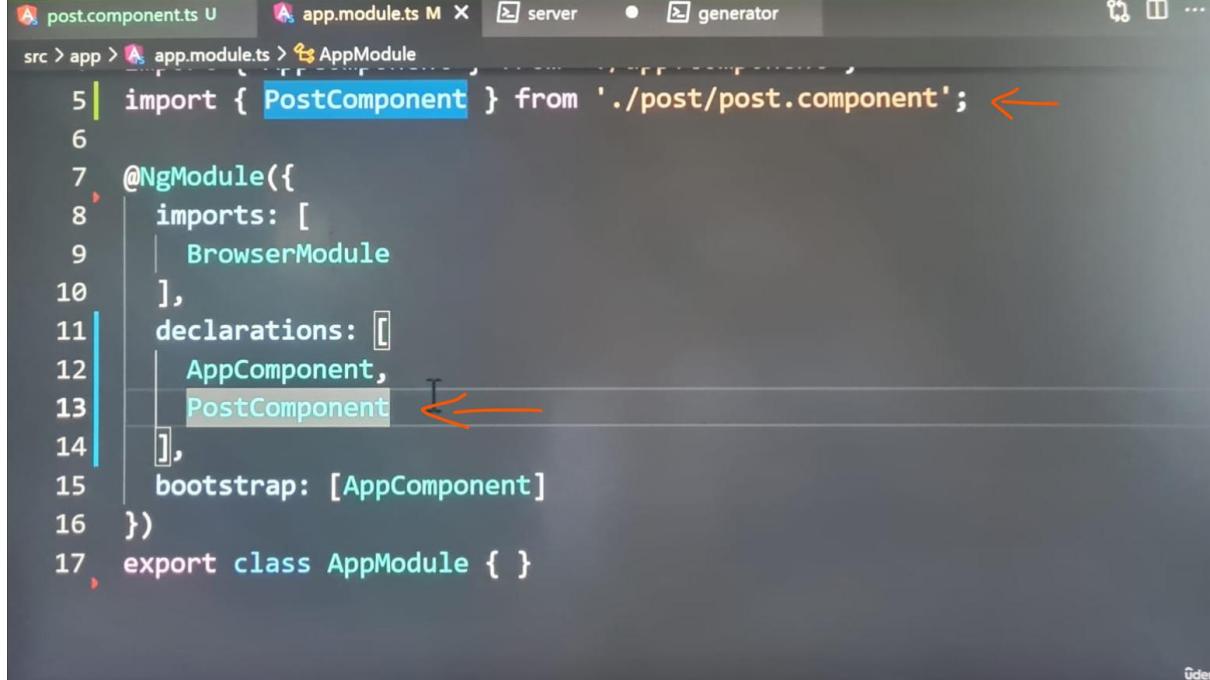
The file has been updated to register the component for us. At the top, the component is being imported into the module. Inside the declaration options, the component has been added to the array, which is awesome.



```

src > app > app.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppComponent } from './app.component';
5 import { PostComponent } from './post/post.component'; ←
6
7 @NgModule({
8   imports: [
9     BrowserModule
10 ],
11 declarations: [
12   AppComponent,
13   PostComponent ←
14 ],
15 bootstrap: [AppComponent]
16 })

```



```

src > app > app.module.ts > AppModule
1 import { PostComponent } from './post/post.component'; ←
2
3 @NgModule({
4   imports: [
5     BrowserModule
6   ],
7   declarations: [
8     AppComponent,
9     PostComponent ←
10   ],
11 bootstrap: [AppComponent]
12 })
13 export class AppModule { }

```

On my PC using latest version of Angular 20.05 -

In the next lecture, we're going to use our component for displaying the image.

In angular version 20.0.5 where can i find the app.modules.ts file ?

ChatGPT said:

In **Angular 20.0.5**, the location of the `app.module.ts` file follows the same standard Angular project structure as in earlier versions.

Default Location of `app.module.ts`

When you create an Angular project using the Angular CLI (e.g., `ng new my-app`), the `app.module.ts` file is located at:

```
arduino  
CopyEdit  
src/app/app.module.ts
```

Full Path:

```
arduino  
CopyEdit  
<root-project-folder>/src/app/app.module.ts
```

Where:

- `<root-project-folder>` → Your project folder (e.g., `my-app`)
 - `src` → Source folder
 - `app` → Main application folder
 - `app.module.ts` → Angular root module
-

How to check:

If you are inside your project folder, just run:

```
bash  
CopyEdit  
cd src/app
```

Then:

```
bash  
CopyEdit  
ls
```

You should see:

```
cpp  
CopyEdit
```

```
app.component.ts
app.component.html
app.component.css (or .scss)
app.module.ts
```

If it's missing:

If you **don't see app.module.ts**, possible reasons:

1. **You deleted it accidentally**
 2. **You're inside a library project or standalone Angular project using "standalone components"**
(Angular 15+ introduced standalone components that **don't require a root module**, but if you didn't choose that explicitly, you should still have app.module.ts.)
-

How to confirm project type:

Open your angular.json or check your main.ts file (src/main.ts).
If you see bootstrap like this:

```
ts
CopyEdit
platformBrowserDynamic().bootstrapModule(AppModule)
```

Then your project uses app.module.ts.

If you see this (Angular Standalone Bootstrapping):

```
ts
CopyEdit
bootstrapApplication(AppComponent, ...)
```

Then your project uses **standalone components** (no app.module.ts).

Summary:

Scenario	Action
Normal Angular app (non-standalone)	Check: src/app/app.module.ts
Standalone Angular app	No app.module.ts exists (by design)
Accidental deletion	Recreate manually or generate via CLI

If you want, send me your **src/main.ts content** and I can tell you definitively which case applies.

Want me to check?

You said:

my main.ts looks as follows: import { bootstrapApplication } from '@angular/platform-browser'; import { appConfig } from './app/app.config'; import { App } from './app/app'; bootstrapApplication(App, appConfig) .catch((err) => console.error(err));

ChatGPT said:

Thanks for sharing your `main.ts`. Based on this line:

```
typescript
CopyEdit
import { bootstrapApplication } from '@angular/platform-browser';
```

And this line:

```
typescript
CopyEdit
bootstrapApplication(App, appConfig)
```

What it means:

Your project is using **Angular Standalone Components**, introduced in Angular v15+ and commonly used in Angular v16, v17, v18, v19, and v20.

This means:

-  You **won't have an `app.module.ts` file**.
 -  Instead, **your app bootstraps directly from a standalone component (`App`)** defined in `src/app/app.ts`.
 -  Application-wide configuration is likely in: `src/app/app.config.ts`
-

Folder Structure for Standalone Angular App:

You should see something like this in your project:

```
lua
CopyEdit
src/
  └── app/
    ├── app.ts           <-- Your standalone root component (instead of
    │   └── app.component.ts)
    │   └── app.config.ts <-- Your global app configuration
    │       └── (other feature components, routes, etc.)
    └── main.ts          <-- Bootstrap entry point
      └── ...
```

How Angular Bootstraps in Your Project:

Instead of this (classic module-based):

```
typescript
CopyEdit
platformBrowserDynamic().bootstrapModule(AppModule);
```

You now have this (standalone API):

```
typescript
CopyEdit
bootstrapApplication(App, appConfig);
```

What to do instead of `app.module.ts`:

If you want to:

Task	Do this with standalone
Declare routes	Use <code>provideRouter</code> in <code>app.config.ts</code>
Add global services	Use <code>providers</code> array in <code>app.config.ts</code>
Import Angular modules	Import them directly inside your standalone components (<code>imports: []</code>)
Use dependency injection	Works the same as before (constructor injection)

Example of Standalone App Component (`app.ts`):

```
typescript
CopyEdit
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterModule } from '@angular/router';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [CommonModule, RouterModule],
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css'],
})
export class App {}
```

Summary:

You are **not supposed to have `app.module.ts`** in this Angular 20 project.
This is **normal and expected behavior for Standalone Angular projects**.

When using Angular 20.0.5 with standalone, and I have created a new component using the following CLI command: `ng generate component post`, where would the post component be registered if the `app.module.ts` file does not exist?

ChatGPT said:

Excellent and very important question! 

In **Angular 20.0.5 with standalone architecture**, when you run:

```
bash
CopyEdit
ng generate component post
```

Angular CLI generates a **standalone component by default**, meaning:

Where is the `PostComponent` registered?

It's NOT registered in `app.module.ts` (because you don't have one). Instead:

- The generated component will have:

```
typescript
CopyEdit
standalone: true
```

in its `@Component()` decorator.

That means:

- ✓ It registers itself as a standalone component.
 - ✓ It doesn't need to be added to any module.
-

Example of Angular 20 Standalone Component (`post.component.ts`)

```
typescript
CopyEdit
import { Component } from '@angular/core';

@Component({
  selector: 'app-post',
  standalone: true,
  templateUrl: './post.component.html',
  styleUrls: ['./post.component.css']
})
export class PostComponent {}
```

How do you use this new standalone component in your app?

If you want to use `PostComponent` **inside another standalone component (like your App component)**, you need to **import it directly** in the `imports` array of the parent component.

Example → In your `app.ts` (root App component):

```
typescript
CopyEdit
import { Component } from '@angular/core';
import { PostComponent } from './post/post.component';

@Component({
  selector: 'app-root',
  standalone: true,
  imports: [PostComponent],
  template: `
    <h1>Welcome</h1>
    <app-post></app-post>  <!-- Using your new component -->
  `
})
export class App {}
```

Summary:

Question	Answer
Where is the component registered?	It self-registers via <code>standalone: true</code>
Do I need a module for it?	 No. No <code>app.module.ts</code> needed
How do I use it inside other components?	Import it into the <code>imports: []</code> array of the parent standalone component

Optional Tip:

If you generate components often and want to **explicitly choose standalone or module-based**, you can use:

```
bash
CopyEdit
ng generate component post --standalone
```

or

```
bash
CopyEdit
ng generate component post --skip-import
```

But Angular CLI **by default makes new components standalone now** in v16+ (including v20).