Creating a Component



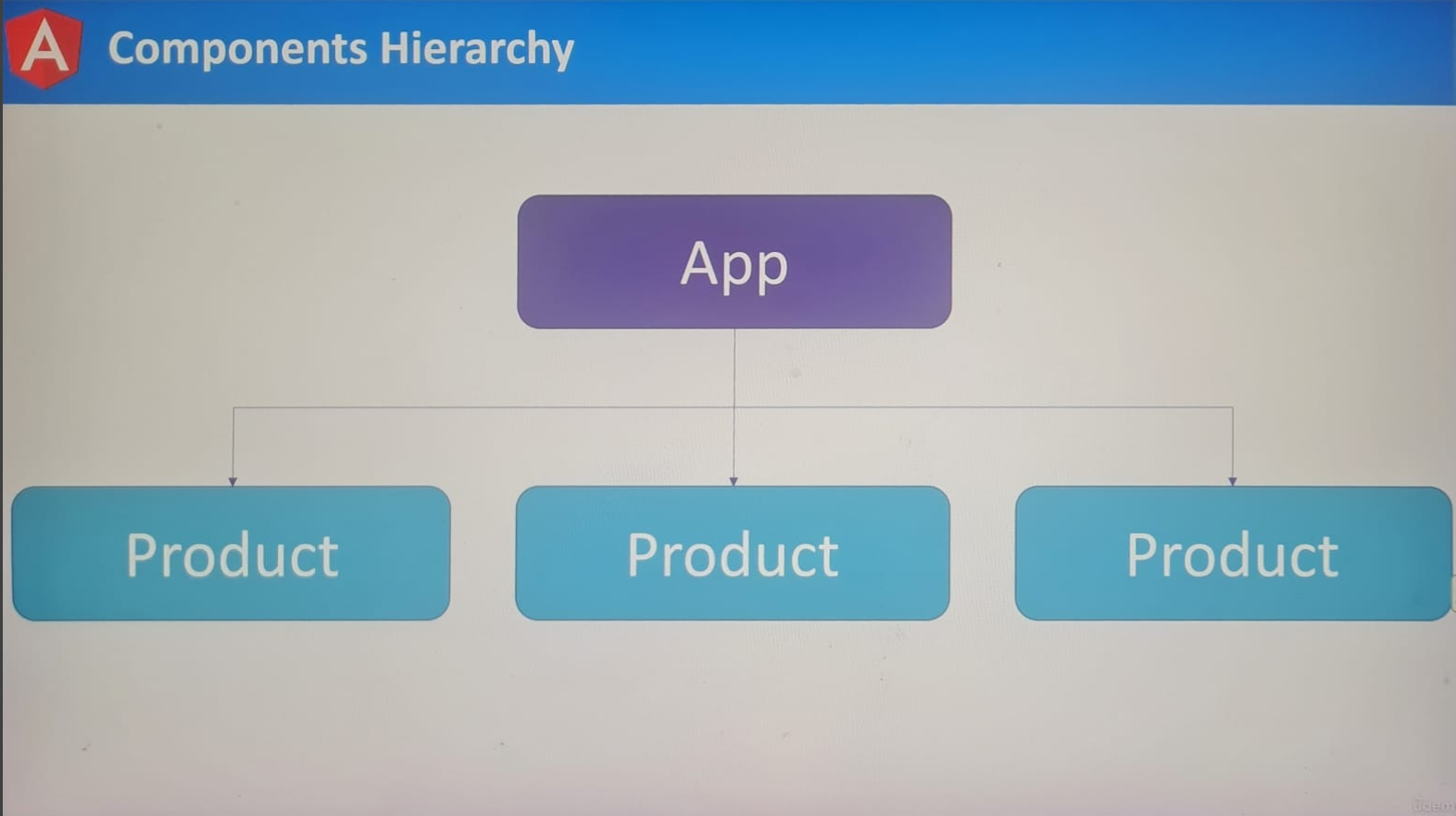
It’s recommended to break and application into several components.

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.

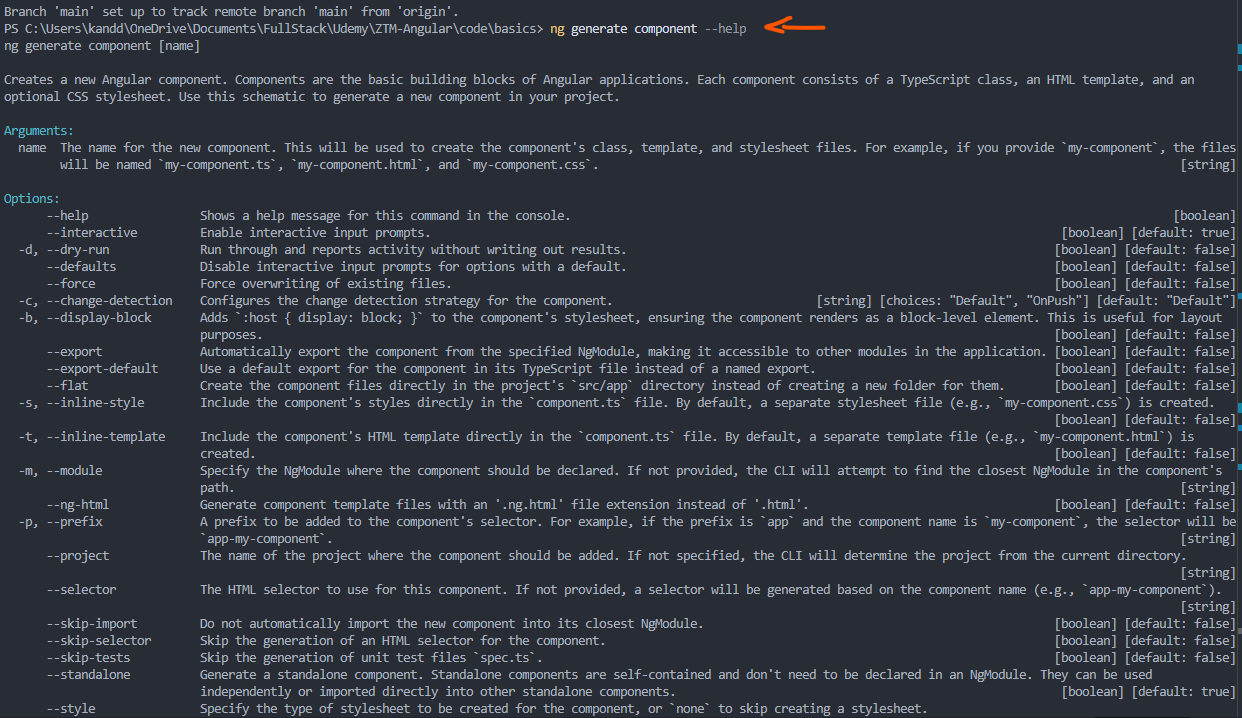
We have two options for creating components:

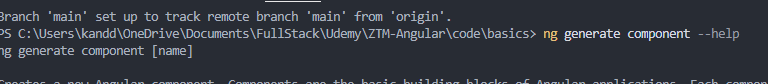
We can create them manually or use the CLI.

We’ve already seen how to created 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.

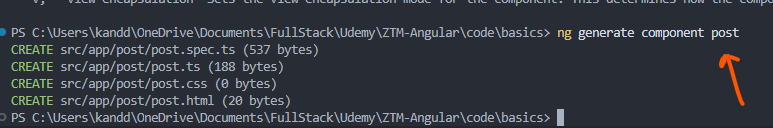
**ng generate component --help**

****

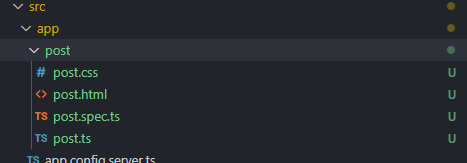
****

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.



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

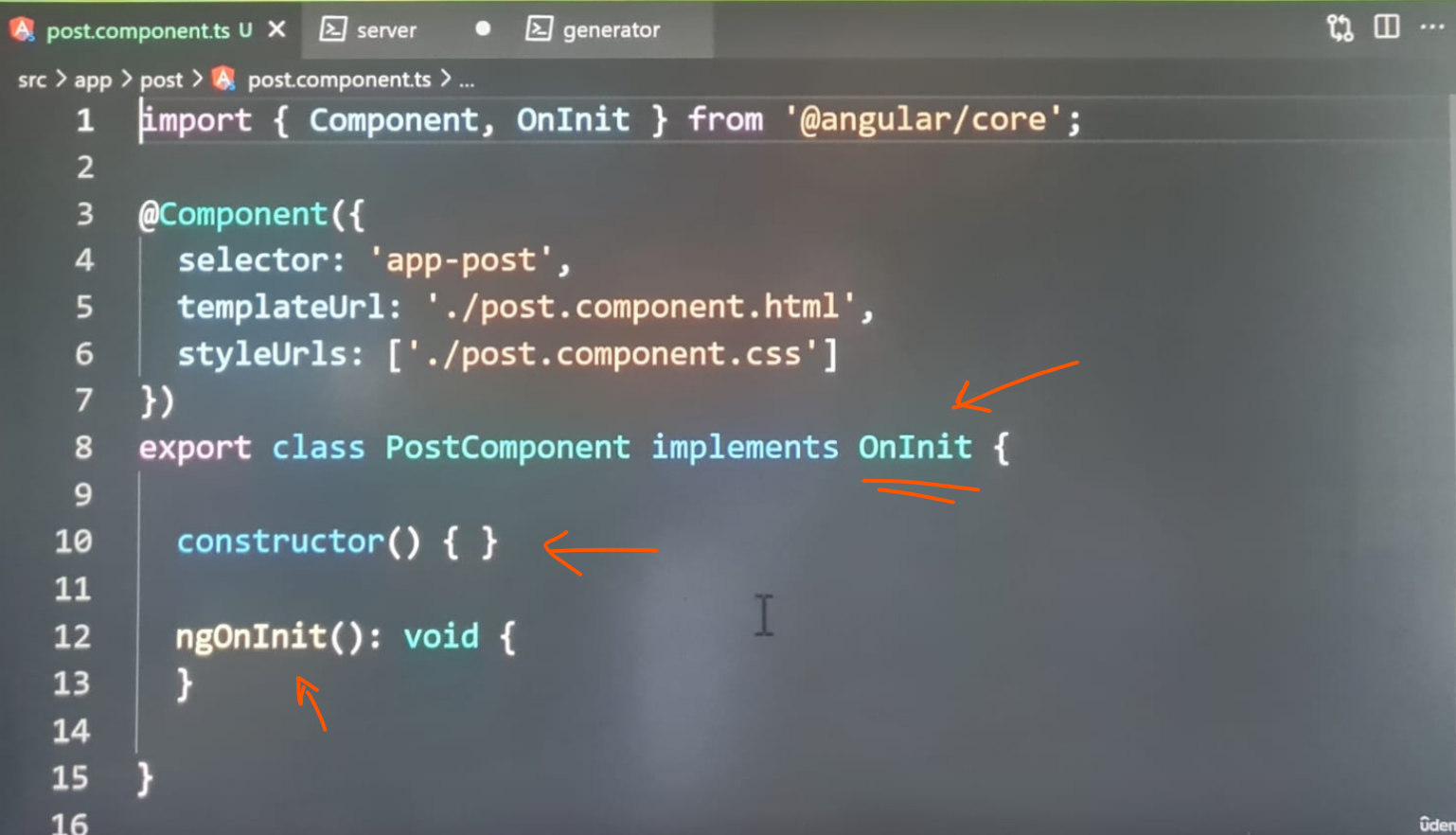


Components should be placed inside a separate directory; it’s a common practice we adopt.

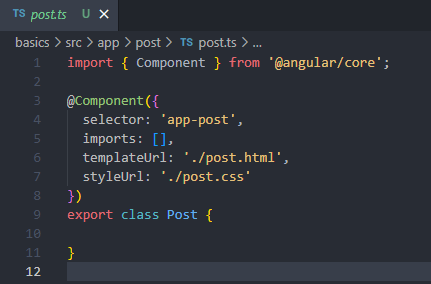
Angular will generate a class, template, stylesheet and test files.

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:

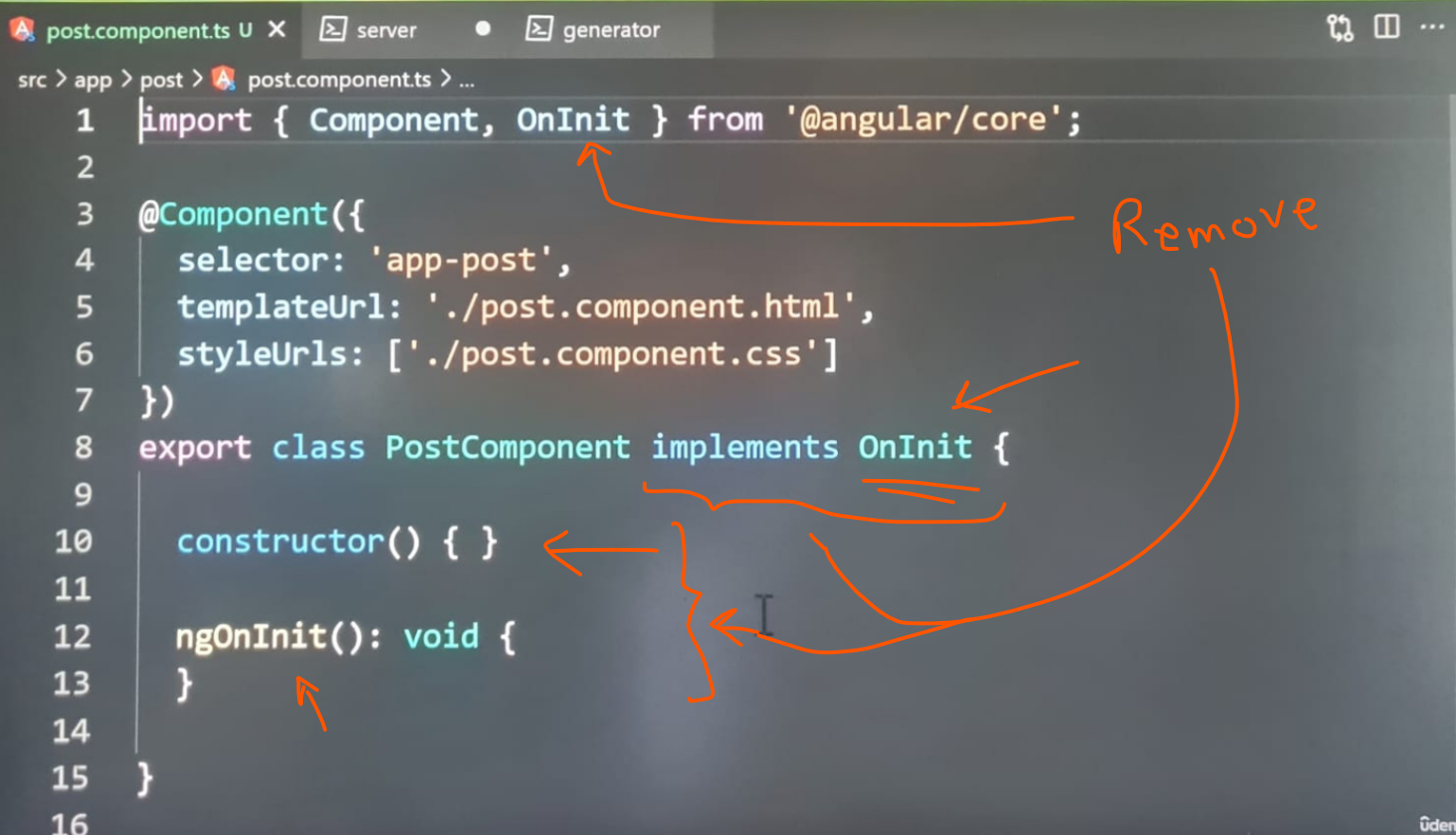


**On my PC running latest version of Angular 20.0.5 – post.ts:**

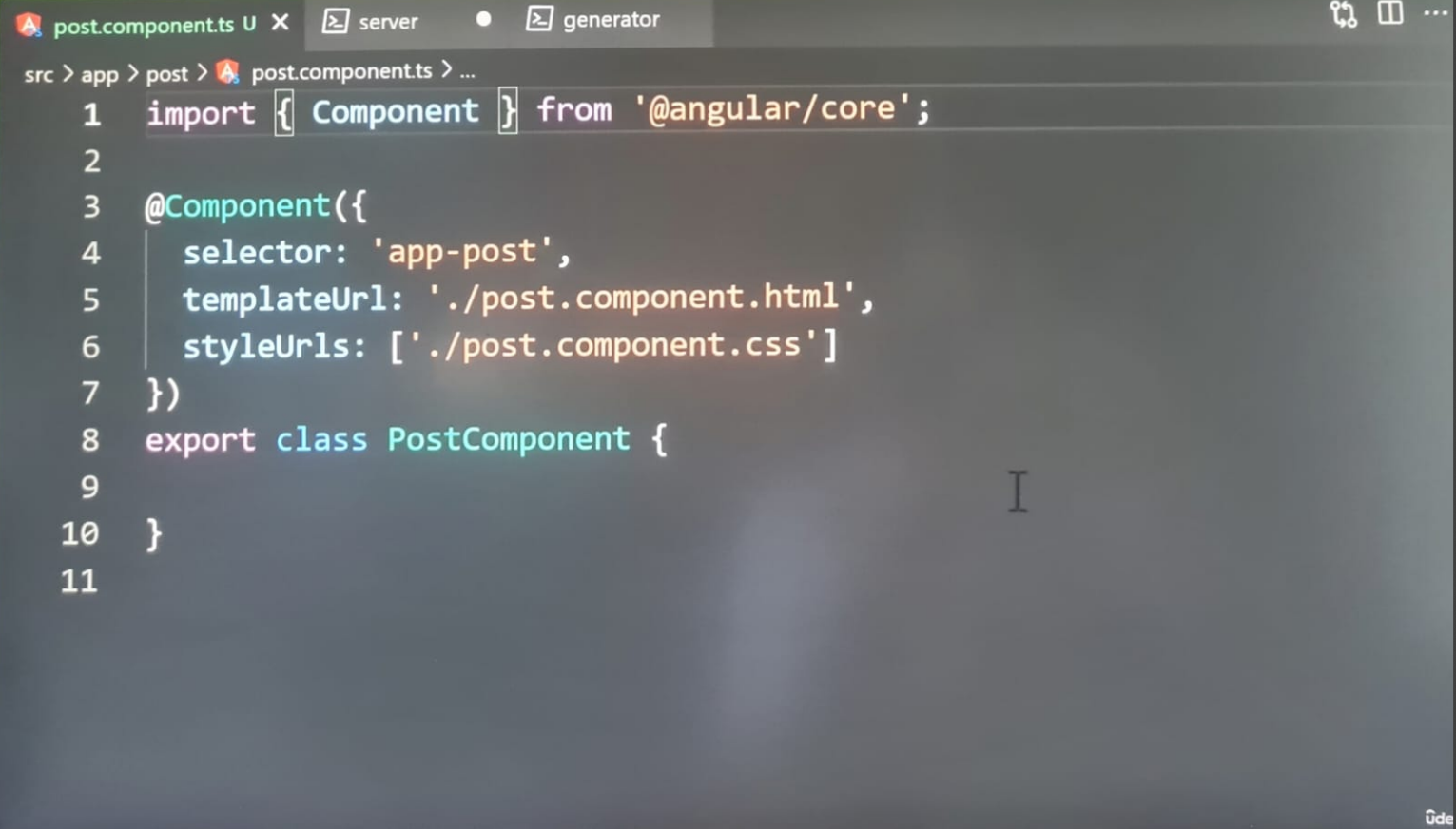


It has the same settings as app component with some exceptions, firstly, the class has a **constructor** function and **ngOnInit method.**

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:

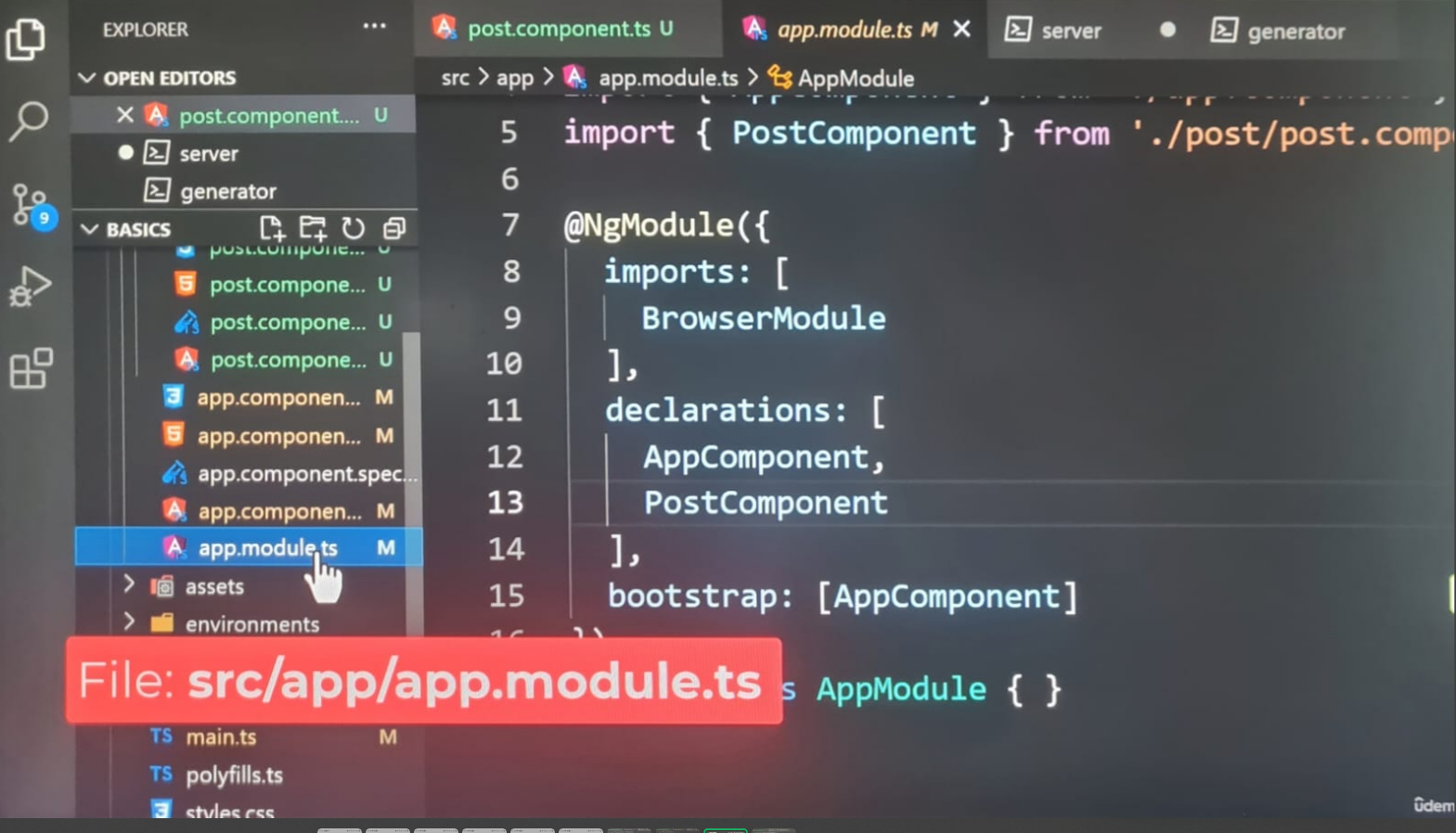


I want to start with a minimal component:

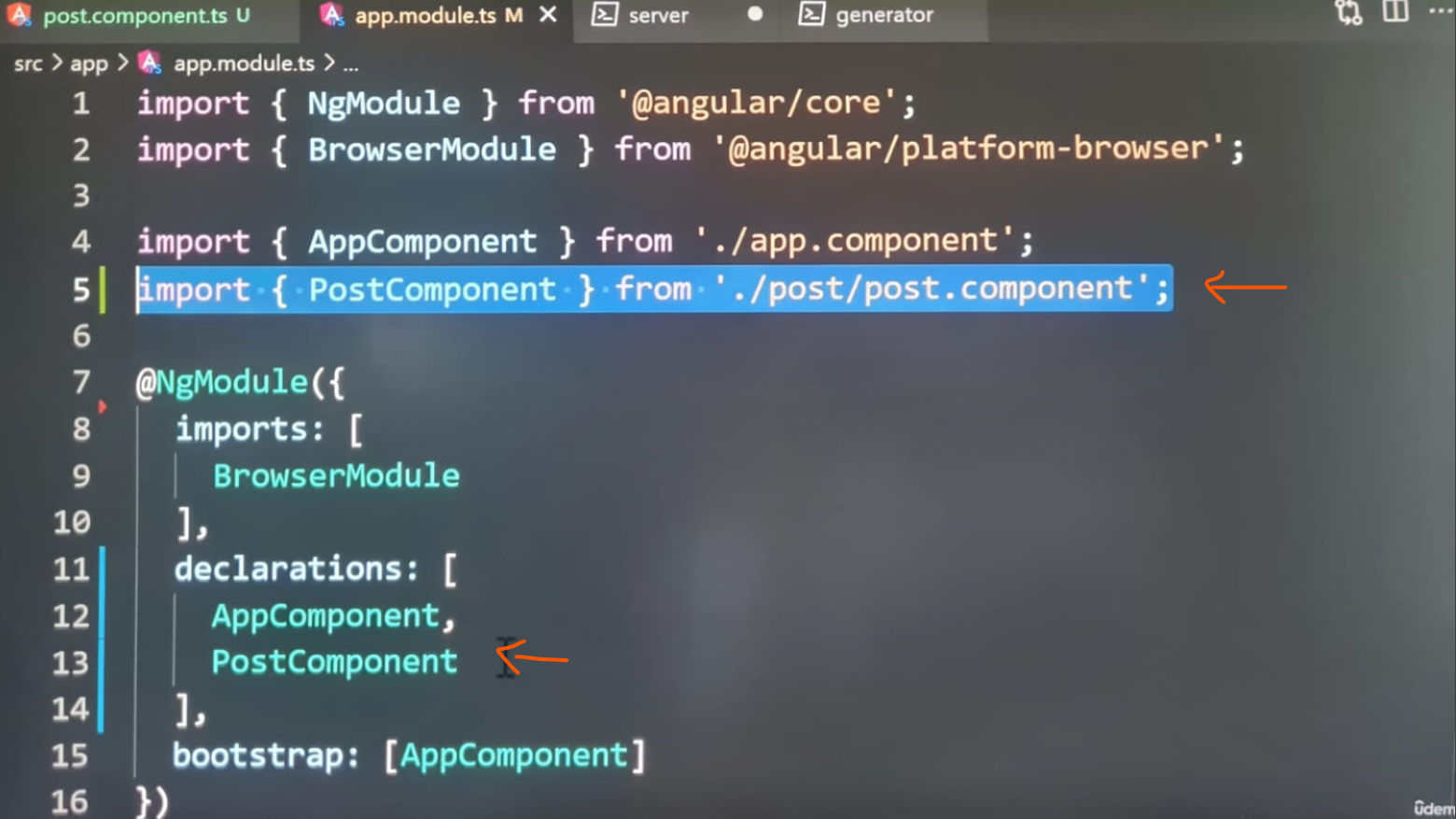


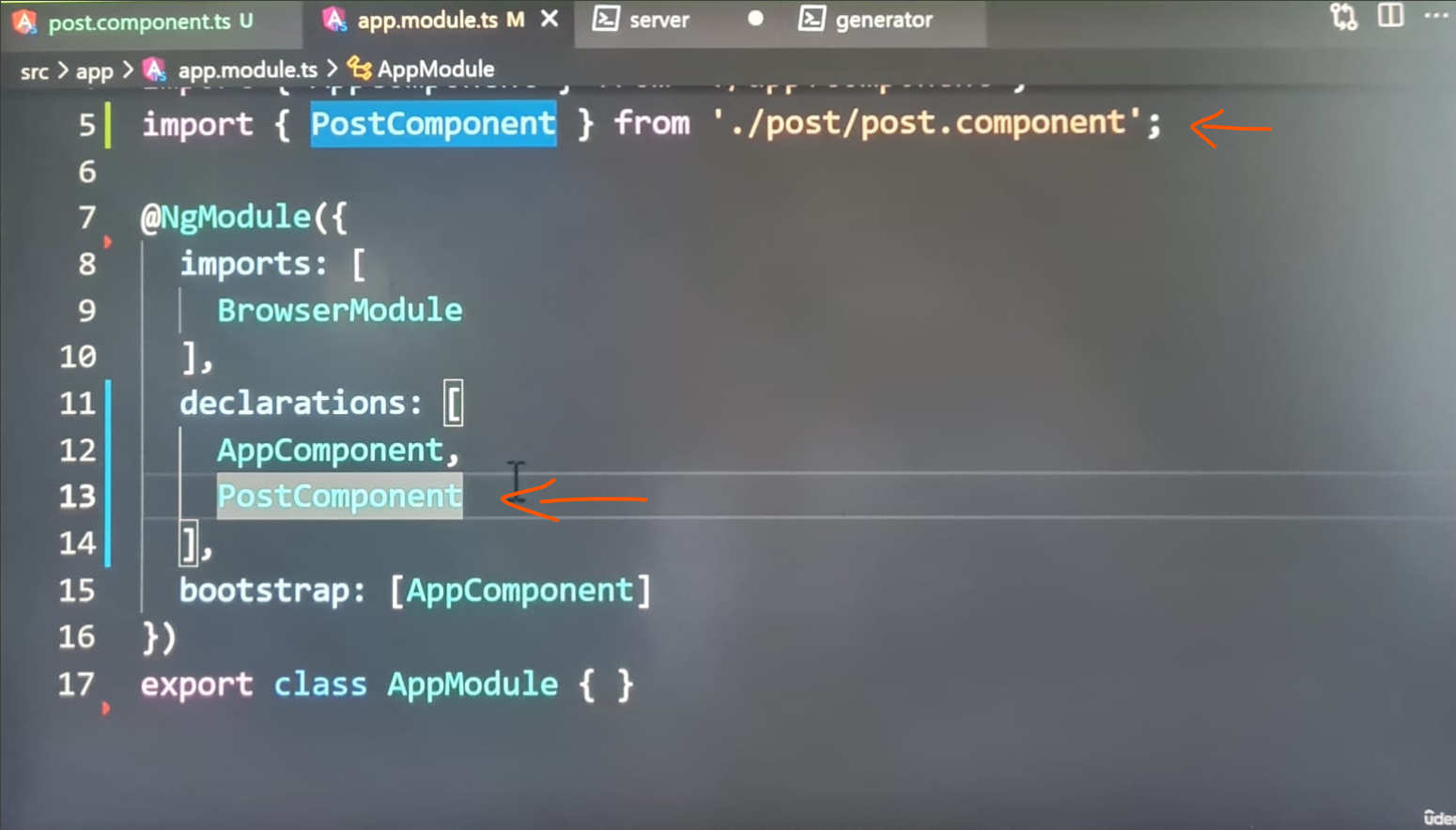
These extra pieces of code will be revisited in a future lecture. Other than that, everything in this file should be familiar to you. Typically, we would need to register the component. Luckily, we won’t have to. The CLI has done so for us.

Let’s check out the app module file:



The 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.





**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 provideRouter in app.config.ts |
| **Add global services** | Use providers array in app.config.ts |
| **Import Angular modules** | Import them directly inside your standalone components (imports: []) |
| **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 craeted a new component using the following CLI command: ng generate component post, where would the post component 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 standalone: true |
| Do I need a module for it? | ❌ No. No app.module.ts needed |
| How do I use it inside other components? | Import it into the imports: [] 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).