

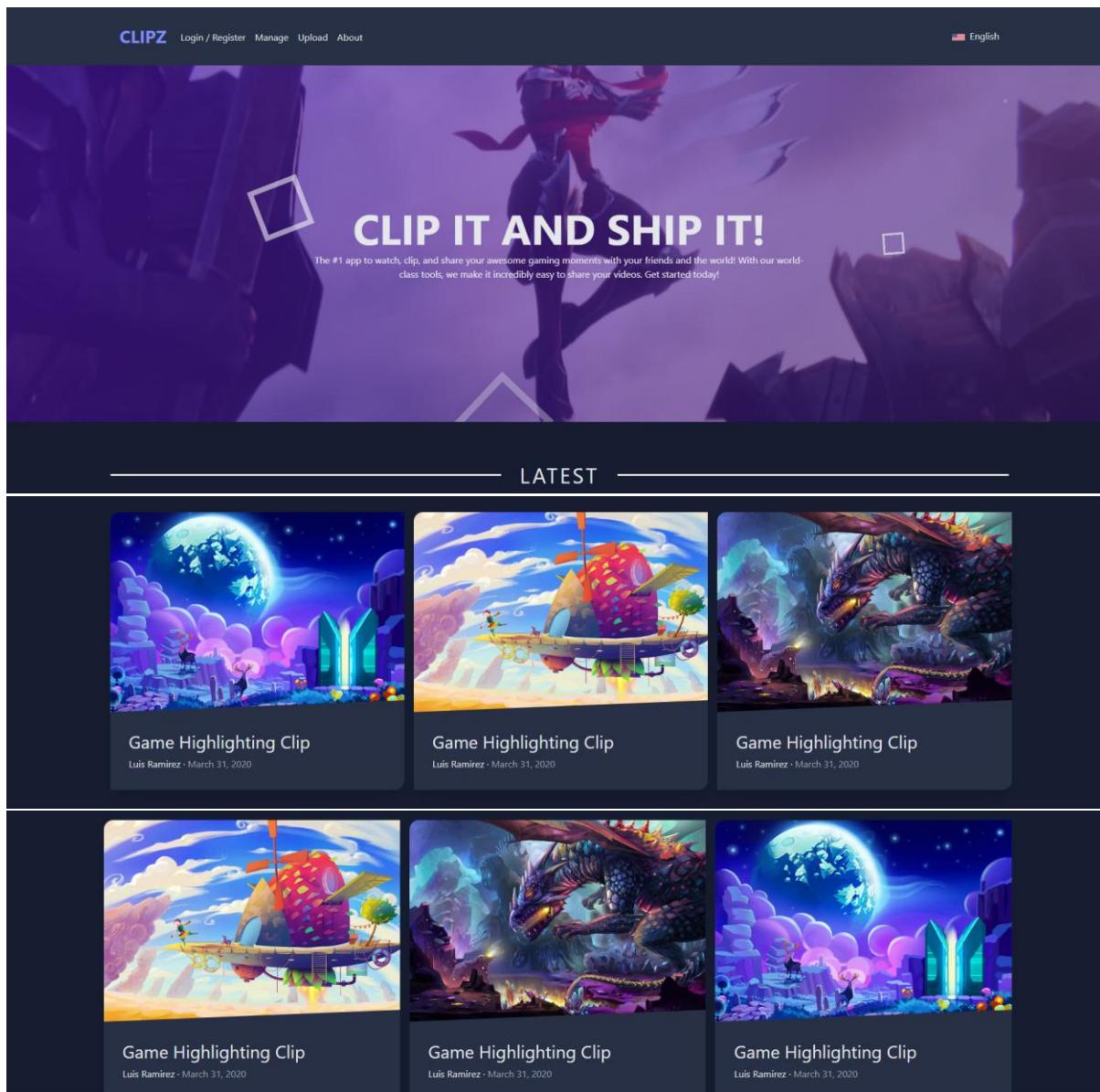
Introduction to Master Project

It's time for the most exciting part of the course, we haven't had the chance to explore every feature in Angular. However, we've covered most of the fundamentals. If we're going to continue further, we should work on a realistic project by working on a real project, we'll get a chance to learn the best practices for structuring a project. So, what are we going to build?

We will develop a project called **Clipz**, a platform for uploading and sharing gaming highlights.

Features include authentication, routing, video processing, state management and so much more.

One of the most interesting features is running a web assembly file to process video files with FFmpeg. We will be playing with cutting edge features, which will surely impress recruiters. On my screen, I'm showing the static design of the app we will be developing:



It doesn't look like much, but believe me, it's going to take a lot of work to convert this template

to a functioning app.

Once we're finished with our project, we will be deploying it. Throughout the development lifecycle of this project, you will be surprised by how helpful Angular can be.

First, we need to create a new project.

Open the command line in your editor. We'll be working in a completely new directory. We aren't going to be installing our project with the basics app we worked on previously.

Inside the command line run the following command:

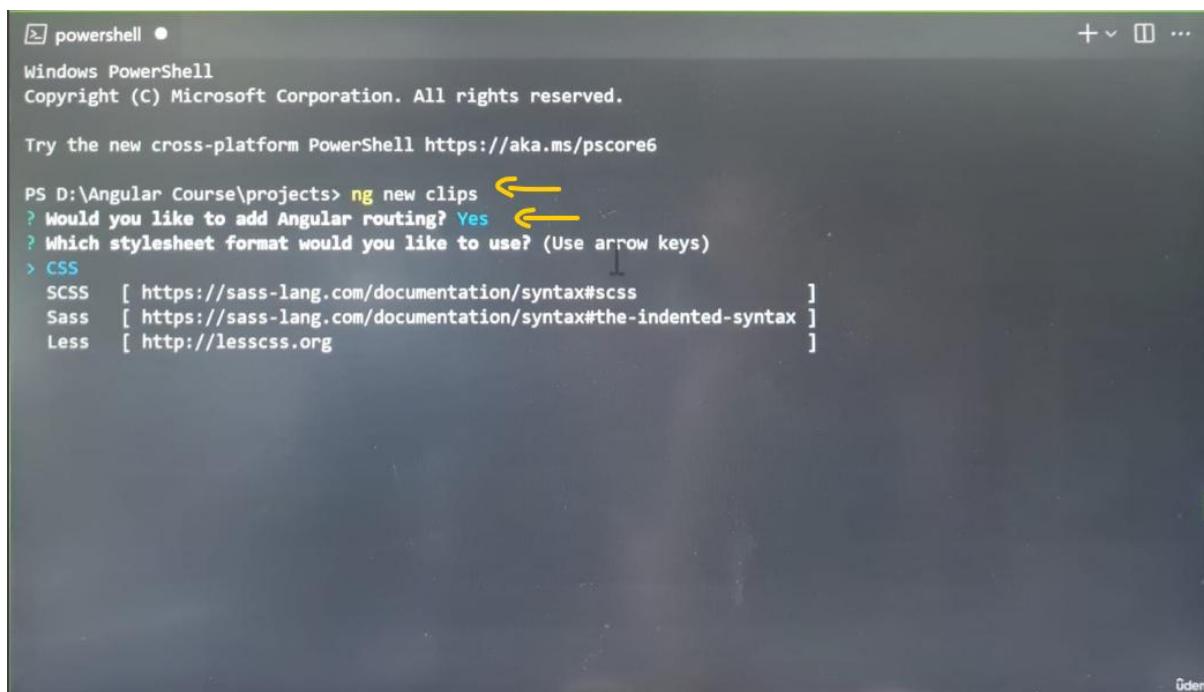
```
ng new clips
```

The CLI will ask us if we'd like to add **routing** for this project.

We will select Yes.

Routing is an optional feature. Larger apps typically have **routing**.

We will discuss how **routing** works in this course.



```
powershell •
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Angular Course\projects> ng new clips ↩
? Would you like to add Angular routing? Yes ↩
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
SCSS [ https://sass-lang.com/documentation/syntax#scss ]
Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
Less [ http://lesscss.org ]
```

The next option will ask us to select a **CSS Preprocessor**.

We're not going to be using a preprocessor.

However, feel free to choose or preprocessor if you are familiar with any of them from this list.

After selecting our options, the CLI will be again creating and installing the project.

This installation may take a while.

```

PS D:\Angular Course\projects> ng new clips
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS ←
CREATE clipes/angular.json (3033 bytes)
CREATE clipes/package.json (1067 bytes)
CREATE clipes/README.md (1051 bytes)
CREATE clipes/tsconfig.json (783 bytes)
CREATE clipes/.editorconfig (274 bytes)
CREATE clipes/.gitignore (604 bytes)
CREATE clipes/.browserslistrc (703 bytes)
CREATE clipes/karma.conf.js (1422 bytes)
CREATE clipes/tsconfig.app.json (287 bytes)
CREATE clipes/tsconfig.spec.json (333 bytes)
CREATE clipes/src/favicon.ico (948 bytes)
CREATE clipes/src/index.html (291 bytes)
CREATE clipes/src/main.ts (372 bytes)
CREATE clipes/src/polyfills.ts (2820 bytes)
CREATE clipes/src/styles.css (80 bytes)
CREATE clipes/src/test.ts (788 bytes)
CREATE clipes/src/assets/.gitkeep (0 bytes)
CREATE clipes/src/environments/environment.prod.ts (51 bytes)
CREATE clipes/src/environments/environment.ts (658 bytes)
CREATE clipes/src/app/app-routing.module.ts (245 bytes)
CREATE clipes/src/app/app.module.ts (393 bytes)
CREATE clipes/src/app/app.component.html (24617 bytes)
CREATE clipes/src/app/app.component.spec.ts (1070 bytes)

```

On my PC with Angular 20.0.5, I am getting different options for installation:

A conversation between me and ChatGPT:

I have Angular 20 installed on my pc, I want to create a new project called clips with routing and css configured, what options should I use ?

ChatGPT said:

To create a new Angular 20 project named `clips` with **routing enabled** and **CSS for styling**, you can use the following CLI command:

```

bash
CopyEdit
ng new clips --routing --style=css

```

🔍 Breakdown of Options:

Option	What it does
<code>clips</code>	Project name
<code>--routing</code>	Adds Angular Router configuration (creates <code>app.routes.ts</code> in standalone)
<code>--style=css</code>	Sets CSS as the default styling language (instead of SCSS, Sass, etc.)

✓ What Happens After Running This:

- It creates a folder called `clips`
- Sets up a standalone Angular app (in Angular 20 this is default)
- Adds routing using a `app.routes.ts` file (instead of `app-routing.module.ts`)

- Configures the project to use plain CSS
-

👉 What to Do Next:

1. Navigate into the project:

```
bash  
CopyEdit  
cd clips
```

2. Serve the app locally:

```
bash  
CopyEdit  
ng serve
```

3. Open browser at: <http://localhost:4200>
-

Would you like me to explain the routing file setup in Angular 20 (standalone), or how to add new pages/components using routing?

You said:

Do you want to create a 'zoneless' application without zone.js (Developer Preview)? (y/N)

ChatGPT said:

That prompt means Angular CLI is giving you the **option to create a "zoneless" application**, which is a **new experimental feature** introduced in Angular 17+.

💡 What does it mean?

- **Zone.js** is used by default in Angular to detect changes in the UI and trigger re-renders.
 - A **zoneless app** skips `zone.js` and uses **signals** or manual change detection instead.
 - Zoneless apps are **faster**, more **predictable**, and **leaner**, but also **require more manual control** over change detection.
-

✓ For most users, especially when starting out:

Choose **no** by pressing `Enter` or typing `n`.

`txt`

CopyEdit

Do you want to create a 'zoneless' application without zone.js (Developer Preview)? (y/N)

Just press `Enter` to accept the default (N = No), which is recommended unless:

- You're building a performance-critical app,
 - You're familiar with Angular Signals and manual change detection,
 - Or you want to experiment with the latest cutting-edge features.
-

Let me know if you want a side-by-side comparison of zoned vs zoneless apps or a guide for when to choose which.

You said:

? Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? (y/N)

ChatGPT said:

That prompt is asking if you want to enable **Angular Universal**, which adds:

- **Server-Side Rendering (SSR)** — Angular renders the initial HTML on the server, improving SEO and performance.
 - **Static Site Generation (SSG)** — Angular pre-renders pages at build time, useful for blogs or mostly static content.
-

Should you enable it?

Your Situation	Recommendation
Beginner or building a basic app like "clips"	<input checked="" type="checkbox"/> No (just press <code>Enter</code>)
Building a SEO-heavy site (e.g., marketing page, blog)	<input checked="" type="checkbox"/> Yes
Building a dynamic app or dashboard	<input checked="" type="checkbox"/> No
Want to learn SSR/SSG later	<input checked="" type="checkbox"/> No for now, can add it later using <code>ng add @angular/ssr</code>

Summary

For your use case:

- Just press `Enter` or type n — you don't need SSR/SSG for now.

You'll get a faster and simpler development experience without the extra complexity.

Would you like me to show how to add SSR later if you ever need it?

You said:

The app I am building is a platform for uploading and sharing gaming highlights

ChatGPT said:

Great! Since you're building a **gaming highlights platform** — something like a **video-sharing web app** — here's what I recommend for those Angular CLI prompts:

✓ Final CLI Setup for Your Case

Run:

```
bash
CopyEdit
ng new clips --routing --style=css
```

And when prompted:

► Do you want to create a 'zoneless' application without zone.js?

Answer: No (or just press Enter)

► Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)?

Answer: No (or just press Enter)

🧠 Why?

✗ Zoneless:

Your app will likely rely on Angular's standard change detection to keep components in sync with video uploads, likes, comments, etc. Going zoneless is advanced and unnecessary right now.

✗ SSR/SSG:

SSR helps with SEO, but gaming highlight platforms usually require:

- Client-heavy features (video playback, user interactions)
- Realtime updates
- Authenticated content

SSR adds complexity and isn't essential unless you're optimizing for search engine indexing of public content.

What to Focus On Instead

Since it's a video-sharing platform, here's what you'll need:

Feature	Angular Tool/Tech
File/video upload	Angular with Firebase Storage or Node.js backend
Routing between pages	--routing already sets this up
Authentication	Firebase Auth / Auth0
Realtime updates (likes/comments)	Firebase Realtime DB / Firestore
Video embedding & playback	HTML5 <video> tag or libraries like Plyr
Responsive UI	Angular Material / Tailwind CSS

After this setup as proposed, will I have the following file app-routing.module.ts inside src/app directory ?

ChatGPT said:

 **No**, you will **not** get app-routing.module.ts in Angular 20 if you're using the default (standalone) setup — even when you use the --routing flag.

What You'll Get Instead (Angular 20 with Standalone API)

After running:

```
bash
CopyEdit
ng new clips --routing --style=css
```

You'll get a **file called**:

```
bash
CopyEdit
src/app/app.routes.ts
```

This file contains a **routes array** that looks like this:

```
ts
CopyEdit
```

```
import { Routes } from '@angular/router';

export const routes: Routes = [];
```

Why the Change?

Angular 15+ introduced the **standalone API**, and as of Angular 17–20, it's the **default**. This simplifies the app by removing the need for NgModules like `AppRoutingModule`.

In standalone Angular:

- Components can directly declare their own routing imports
 - `app.routes.ts` replaces `app-routing.module.ts`
-

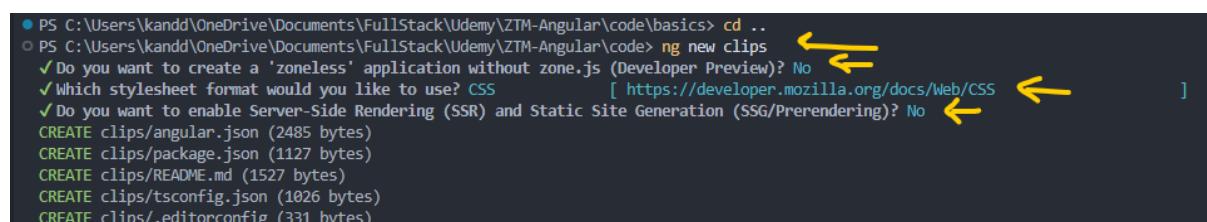
If You *Do* Want `app-routing.module.ts`

If you prefer the old, module-based style (e.g., for tutorials or legacy compatibility), you can **opt out of standalone** by adding:

```
bash
CopyEdit
ng new clips --routing --style=css --standalone=false
```

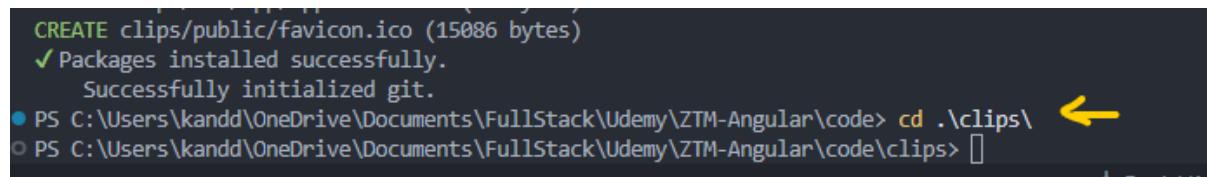
That will generate:

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



```
PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code\basics> cd ..
PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code> ng new clips
  ✓ Do you want to create a 'zoneless' application without zone.js (Developer Preview)? No ↵
  ✓ Which stylesheet format would you like to use? CSS [ https://developer.mozilla.org/docs/Web/CSS ] ↵
  ✓ Do you want to enable Server-Side Rendering (SSR) and Static Site Generation (SSG/Prerendering)? No ↵
CREATE clips/angular.json (2485 bytes)
CREATE clips/package.json (1127 bytes)
CREATE clips/README.md (1527 bytes)
CREATE clips/tsconfig.json (1026 bytes)
CREATE clips/.editorconfig (331 bytes)
```

After the installation is complete, let's move in to the new project.

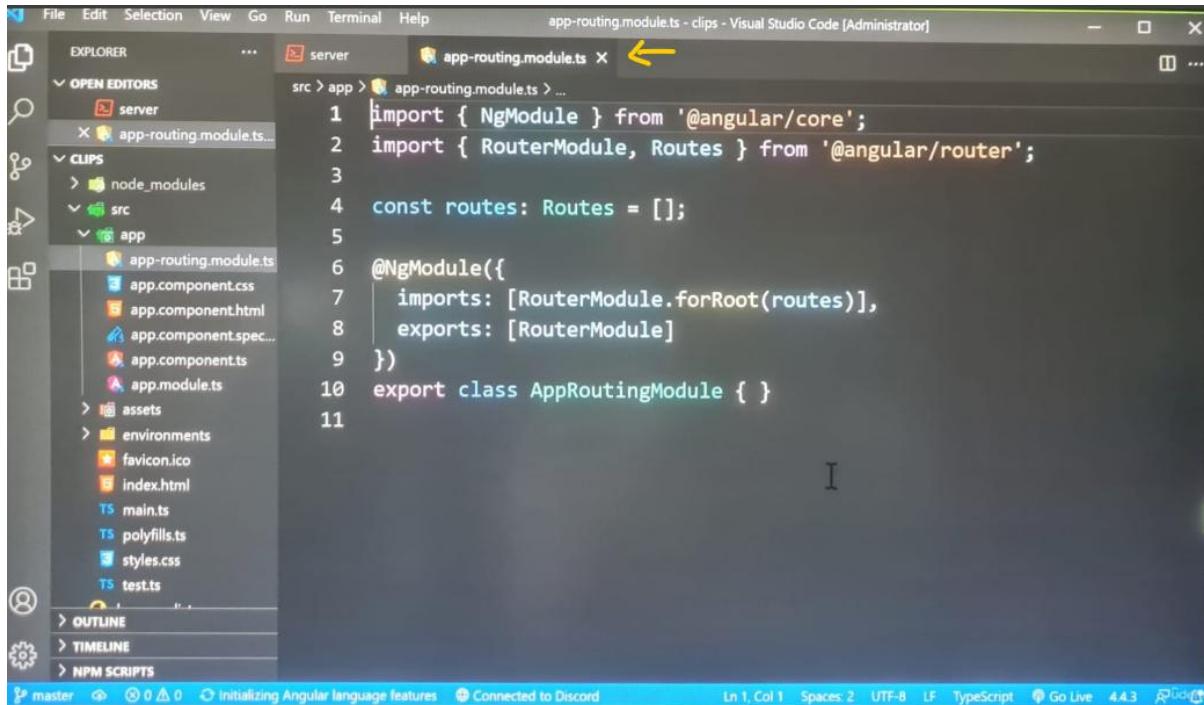


```
CREATE clips/public/favicon.ico (15086 bytes)
✓ Packages installed successfully.
  Successfully initialized git.
PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code> cd .\clips\
PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code\clips> ↵
```

There aren't significant changes to the default project.

The most noticeable difference is a new file called `app-routing.module.ts` inside the `src/app`

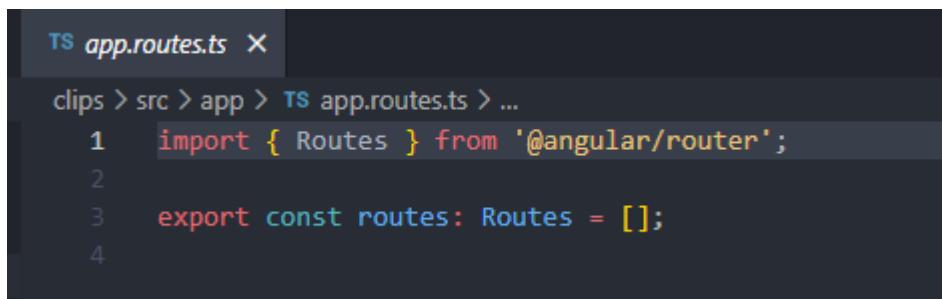
directory. [In my case, I don't have this file. Instead, I have `src/app/app.routes.ts` – see the conversation with ChatGPT above].



```

File Edit Selection View Go Run Terminal Help
EXPLORER OPEN EDITORS CLIPS
server app-routing.module.ts < ...
src > app > app-routing.module.ts > ...
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3
4 const routes: Routes = [];
5
6 @NgModule({
7   imports: [RouterModule.forRoot(routes)],
8   exports: [RouterModule]
9 })
10 export class AppRoutingModule { }
11

```



```

TS app.routes.ts < ...
clips > src > app > TS app.routes.ts > ...
1 import { Routes } from '@angular/router';
2
3 export const routes: Routes = [];
4

```

We will be exploring this file in an upcoming lecture, Angular has provided us with some **routes** for the default project. We can safely ignore it for the time being.

Other than that, we're good to go.

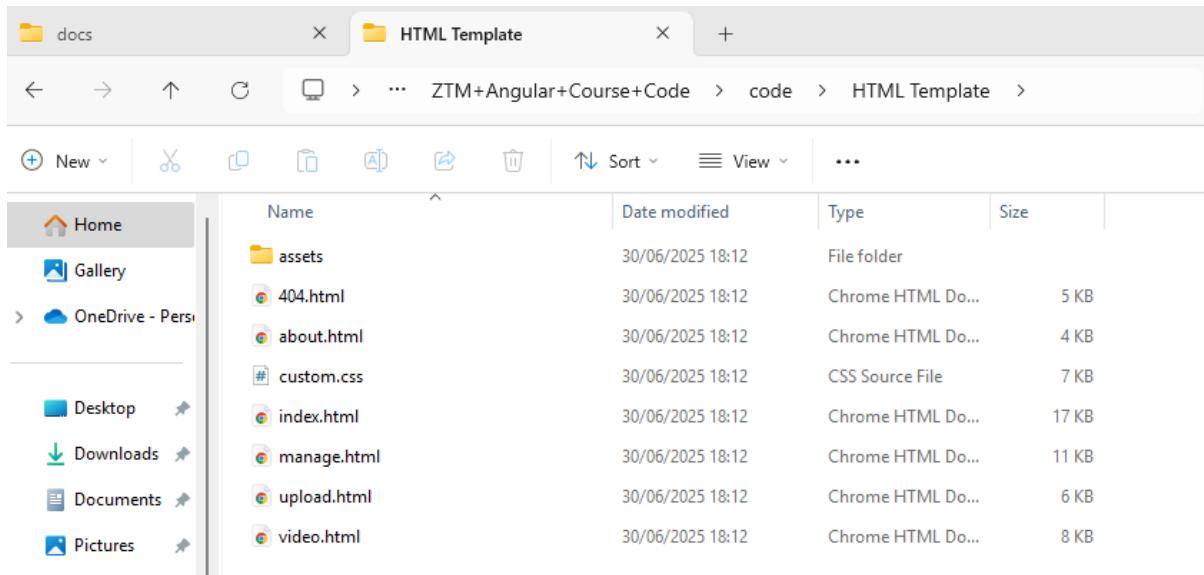
Now the question is where do we begin? What's the first step we should take for developing an app?

It can be a daunting task to get started.

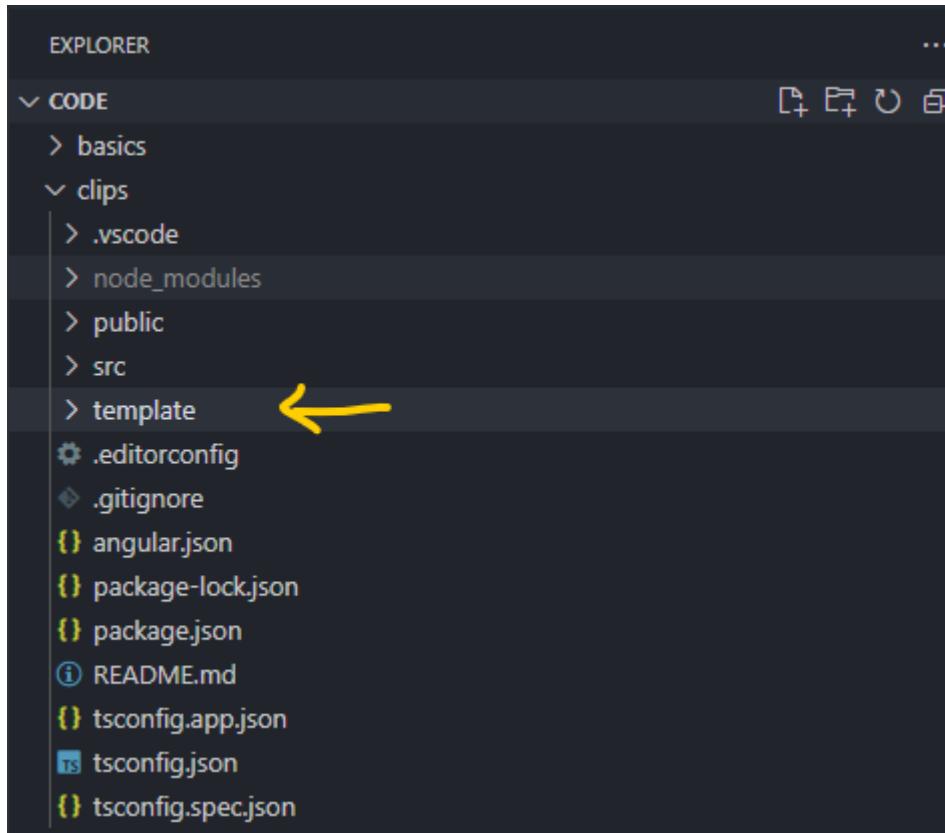
From my experience, I like to start by importing the static design. We already have the template in hands. We should start adding it to our project.

In the resource section of this lecture. I provide a zip file with the HTML and CSS for our app.

It's completely static. I haven't done anything to this template to prepare it for Angular.



We are going to make it functional from top to bottom. Download this file before moving on. After downloading the zip file. Create a new directory in our project called **templates**.



This is not a normal folder to have in a project. It's not going to stick around, once we've converted the templates, it can be safely removed.

We don't have to worry about Angular doing anything with this folder.

The purpose of this folder is to temporarily store our template files so that we can reference them.

Drag and drop the template files from the zip file to this new folder.

After doing so, we can start to transfer the templates.

We will begin with the **home page**.

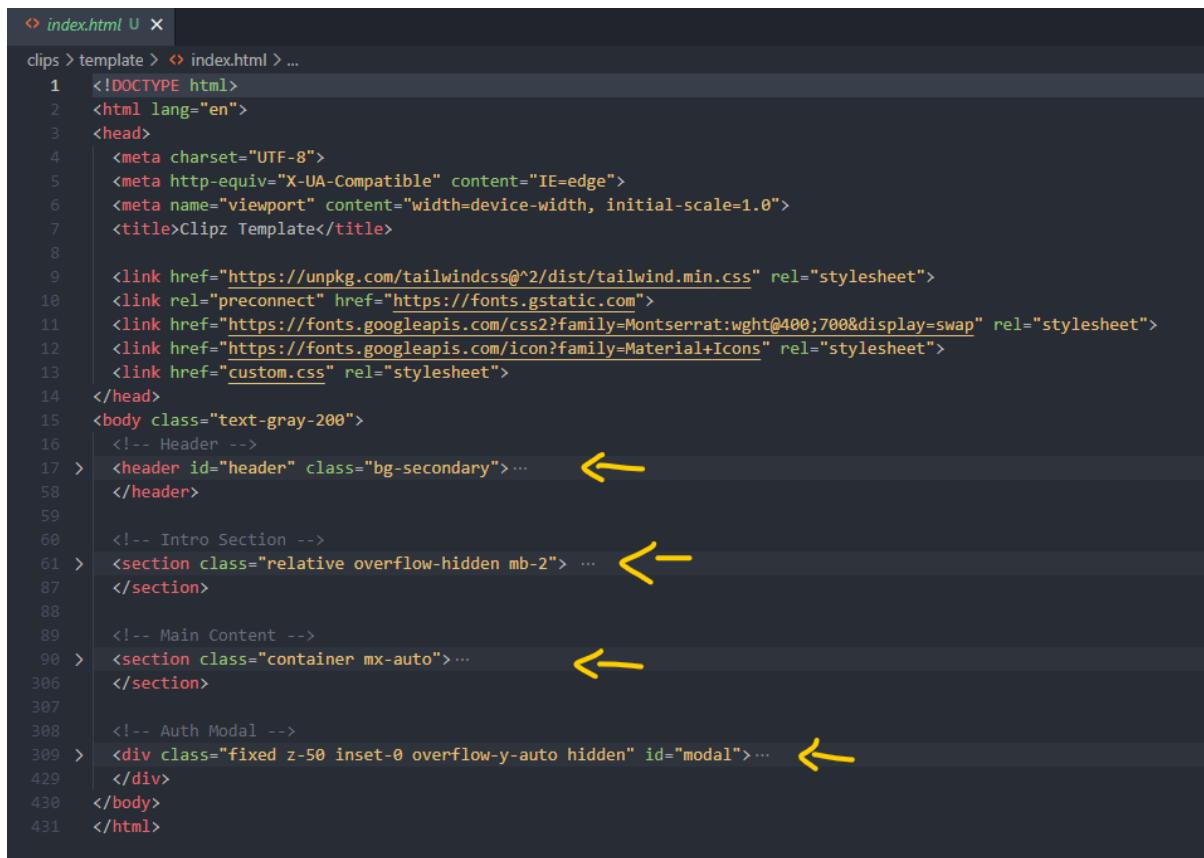
Open the **index.html** file, in the **template** directory. Inside this file, I've added dozens of comments.

These comments will help you navigate the sections of the project.

I will be referring to them whenever we need to work on a particular section.

I'm going to minimize some of the sections in the body tag.

The home page has a total of four sections:



```

index.html U
clips > template > index.html > ...
1  <!DOCTYPE html>
2  <html lang="en">
3  <head>
4    <meta charset="UTF-8">
5    <meta http-equiv="X-UA-Compatible" content="IE=edge">
6    <meta name="viewport" content="width=device-width, initial-scale=1.0">
7    <title>Clipz Template</title>
8
9    <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
10   <link rel="preconnect" href="https://fonts.gstatic.com">
11   <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap" rel="stylesheet">
12   <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
13   <link href="custom.css" rel="stylesheet">
14 </head>
15 <body class="text-gray-200">
16   <!-- Header -->
17   > <header id="header" class="bg-secondary"> ... ←
18   </header>
19
20   <!-- Intro Section -->
21   > <section class="relative overflow-hidden mb-2"> ... ←
22   </section>
23
24   <!-- Main Content -->
25   > <section class="container mx-auto"> ... ←
26   </section>
27
28   <!-- Auth Modal -->
29   > <div class="fixed z-50 inset-0 overflow-y-auto hidden" id="modal"> ... ←
30   </div>
31 </body>
32 </html>

```

We have the header, intro, main and authentication sections.

Let's make a copy of them:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Clipz Template</title>
  <link href="https://unpkg.com/tailwindcss@^2/dist/tailwind.min.css" rel="stylesheet">
  <link rel="preconnect" href="https://fonts.gstatic.com">
  <link href="https://fonts.googleapis.com/css2?family=Montserrat:wght@400;700&display=swap" rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
  <link href="custom.css" rel="stylesheet">
</head>
<body class="text-gray-200">
  <!-- Header -->
  <header id="header" class="bg-secondary"> ...
  </header>

  <!-- Intro Section -->
  <section class="relative overflow-hidden mb-2"> ...
  </section>

  <!-- Main Content -->
  <section class="container mx-auto"> ...
  </section>

  <!-- Auth Modal -->
  <div class="fixed z-50 inset-0 overflow-y-auto hidden" id="modal"> ...
  </div>
</body>
</html>

```

Next, open the **app.component.html** file:

We will replace the contents of this file with the code we've copied:

```

<header id="header" class="bg-secondary"> ...
</header>

<!-- Intro Section -->
<section class="relative overflow-hidden mb-2"> ...
</section>

<!-- Main Content -->
<section class="container mx-auto"> ...
</section>

<!-- Auth Modal -->
<div class="fixed z-50 inset-0 overflow-y-auto hidden" id="modal"> ...
</div>

```

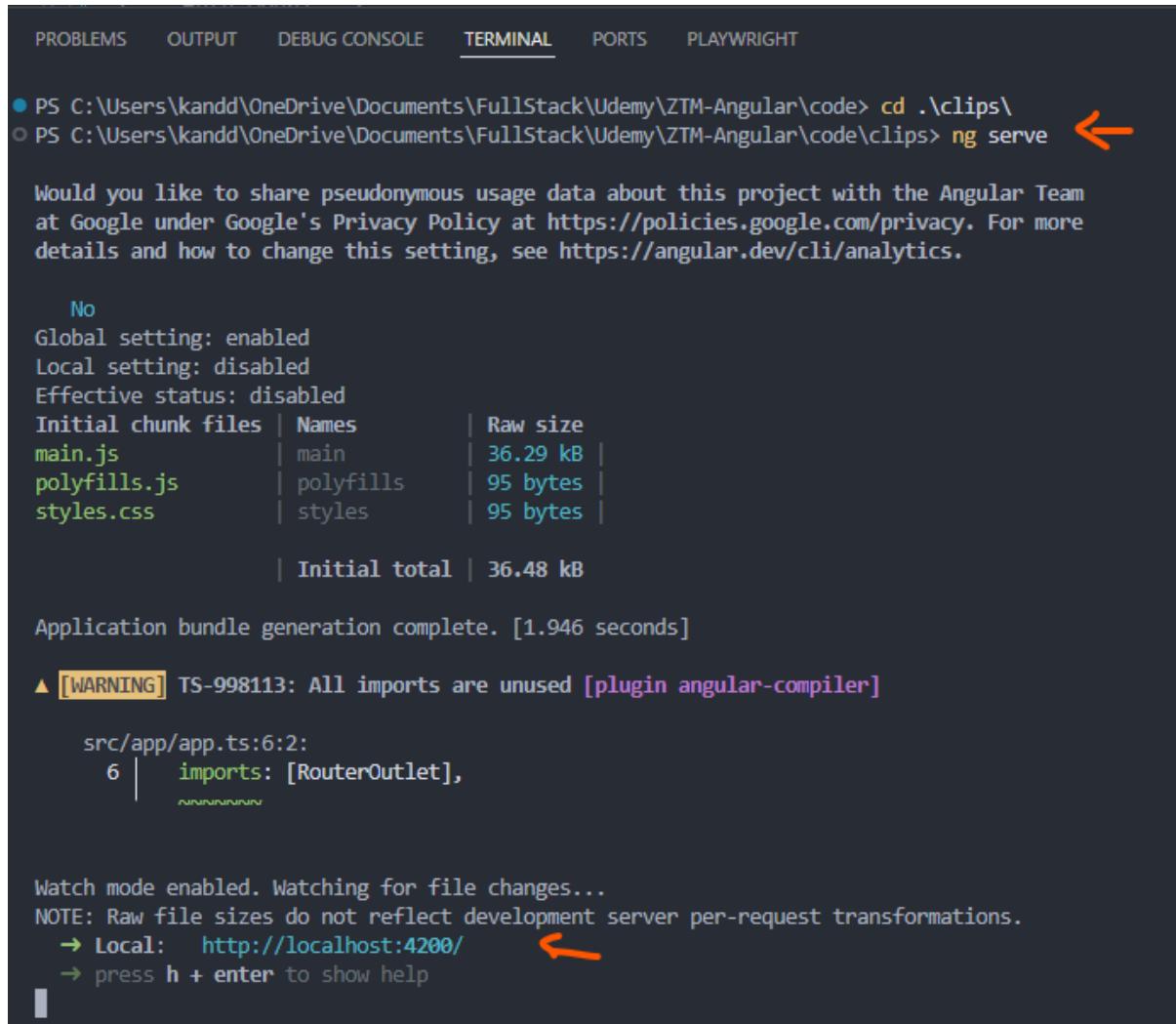
You may be thinking, why are we adding this code to the **index.html** file in the source directory?

We want to make this content dynamic. Therefore, we need to start a template in a component.

Otherwise, we won't be able to apply **pipes**, **directives** or **anything else** we need to make a template **dynamic**.

After pasting in the contents.

Let's turn on the server with the **ng serve** command.



```

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    PLAYWRIGHT

● PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code> cd .\clips\
○ PS C:\Users\kandd\OneDrive\Documents\FullStack\Udemy\ZTM-Angular\code\clips> ng serve ↵

Would you like to share pseudonymous usage data about this project with the Angular Team
at Google under Google's Privacy Policy at https://policies.google.com/privacy. For more
details and how to change this setting, see https://angular.dev/cli/analytics.

No
Global setting: enabled
Local setting: disabled
Effective status: disabled
Initial chunk files | Names          | Raw size
main.js              | main           | 36.29 kB |
polyfills.js         | polyfills      | 95 bytes |
styles.css           | styles          | 95 bytes |

| Initial total | 36.48 kB

Application bundle generation complete. [1.946 seconds]

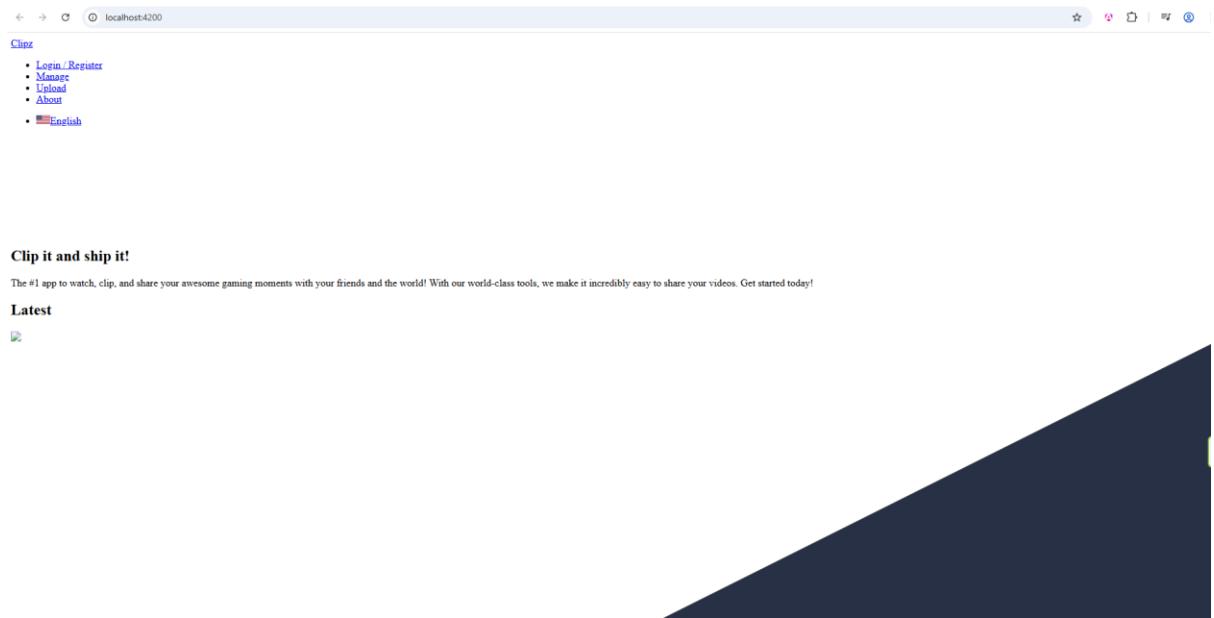
▲ [WARNING] TS-998113: All imports are unused [plugin angular-compiler]

src/app/app.ts:6:2:
  6 |   imports: [RouterOutlet],
     ~~~~~~
     ↵

Watch mode enabled. Watching for file changes...
NOTE: Raw file sizes do not reflect development server per-request transformations.
→ Local: http://localhost:4200/ ↵
→ press h + enter to show help
  ↵
  
```

Next, let's view the app in the browser:

The app is completely broken. It looks nothing like the static design:



That's to be expected since we didn't load the CSS for this template.

The template uses a framework called **Tailwind**, which requires a special installation process.

In the next lecture we will dive into **Tailwind**.