Understanding the Angular 21 Project Structure

A beginner's guide to what every file and folder actually means

1. Why This Matters

When you run `ng new my-app`, Angular's CLI (Command Line Interface) generates a full project skeleton for you. It can feel overwhelming at first — dozens of files you never asked for. This guide walks through **every important file and folder** so you understand exactly what it does and why it's there.

2. The Full Folder Tree

```
my-app/
├── .angular/            # CLI build cache (auto-generated, ignore it)
├── .vscode/             # Editor-specific settings (VS Code launch/debug config)
├── node_modules/        # All installed npm packages/dependencies
├── public/              # Static assets served as-is (images, icons, robots.txt)
├── src/                 # YOUR application source code lives here
│   ├── app/
│   │   ├── app.ts           # Root component logic (TypeScript class)
│   │   ├── app.html         # Root component's HTML template
│   │   ├── app.css          # Root component's scoped styles
│   │   ├── app.spec.ts      # Unit test file for the root component
│   │   ├── app.config.ts    # App-wide providers & configuration
│   │   └── app.routes.ts    # Route definitions (URL → component mapping)
│   ├── index.html           # The single HTML page the whole app boots into
│   ├── main.ts              # Entry point — bootstraps the Angular app
│   └── styles.css           # Global styles applied app-wide
├── .editorconfig            # Consistent code formatting rules across editors
├── .gitignore               # Tells Git which files/folders to NOT track
├── angular.json             # CLI/build configuration (the "brain" of ng commands)
├── package.json             # Project metadata + dependency list
├── package-lock.json        # Exact locked versions of every installed package
├── tsconfig.json            # Base TypeScript compiler settings
├── tsconfig.app.json        # TypeScript settings specific to the app build
└── tsconfig.spec.json       # TypeScript settings specific to running tests
```

> **Note:** Angular 21 (like Angular 17+) uses **standalone components by default** — there is no `app.module.ts` anymore. Older tutorials showing an `AppModule` file are based on pre-17 Angular.

3. Root-Level Files Explained

File / Folder	What It Is
`node_modules/`	Every third-party package your app depends on (Angular core, RxJS, etc.), installed by `npm install`. Never edit this folder manually; never commit it to Git.
`public/`	Holds truly static files — things like a `favicon.ico` or images that should be copied as-is into the final build output, without any processing.
`angular.json`	The CLI's configuration file. It tells Angular how to build, serve, and test your project — output paths, styles/scripts to include, budgets, environments, etc.
`package.json`	Lists your project's dependencies (libraries it needs) and scripts (like `npm start`). This is the standard Node.js project manifest.
`package-lock.json`	Auto-generated. Locks the *exact* version of every package so that everyone on the team installs identical dependency versions.
`tsconfig.json`	Base configuration for the TypeScript compiler — things like which JS version to compile down to, strictness rules, etc.
`tsconfig.app.json` / `tsconfig.spec.json`	Extend the base tsconfig with settings specific to building the app vs. running unit tests.
`.gitignore`	Lists files/folders Git should ignore (like `node_modules/` and build output) so they never get committed.
`.editorconfig`	Keeps formatting (indentation, line endings) consistent across different code editors/IDEs used by the team.

4. Inside `src/` — The Real App Code

4.1 `src/main.ts` — The Entry Point

This is the very first file that runs. It calls `bootstrapApplication()` to start your Angular app using the root component and the configuration from `app.config.ts`. Think of it as the "ignition key" of the app.

4.2 `src/index.html` — The Host Page

Angular is a Single Page Application (SPA) — the entire app lives inside one real HTML page. This file contains the `<app-root></app-root>` tag, which is where Angular injects your whole application at runtime.

4.3 `src/styles.css`

Global CSS that applies to the entire application (fonts, resets, theme variables) — as opposed to component-level CSS files, which only affect that one component.

5. Inside `src/app/` — The Root Component & Config

In modern Angular, each component is made of up to four files that share a name:

File	Purpose
`app.ts`	The component **class** — written in TypeScript. Holds properties, methods, and the logic behind the view. Decorated with `@Component()`, which links it to its template and styles.
`app.html`	The component's **template** — regular HTML plus Angular syntax like `*ngFor`, `{{ interpolation }}`, and event bindings.
`app.css`	Styles scoped *only* to this component — they won't leak out and affect other components.
`app.spec.ts`	Unit test file (using Jasmine/Karma or Jest) that verifies the component behaves correctly. The `.spec.ts` suffix is the Angular convention for test files.

> **Tip:** This "one component, four files" pattern repeats for *every* component you generate with `ng generate component my-thing` — so once you understand `app.ts/html/css/spec.ts`, you understand the shape of every component in the app.

5.1 `app.config.ts` — Application Configuration

Replaces the old `AppModule`. This file exports an `ApplicationConfig` object containing **providers** — things like the router, HTTP client, or any service the whole app needs access to. It's passed into `bootstrapApplication()` in `main.ts`.

5.2 `app.routes.ts` — Routing Table

Defines which component should load for which URL path, e.g. mapping `/home` to `HomeComponent`. This array is fed into the router via `app.config.ts`, enabling navigation without full page reloads.

6. Folders You'll See Later (Not in a Fresh Project)

As the app grows, you (or your team) will typically create these yourself — they are conventions, not CLI defaults:

Folder	Typical Purpose
`src/app/components/`	Reusable UI building blocks (buttons, cards, modals).
`src/app/pages/` or `features/`	Top-level "screens" tied to routes (e.g. `login-page`, `dashboard-page`).
`src/app/services/`	Injectable classes that hold business logic or talk to APIs, shared across components.
`src/app/models/`	TypeScript interfaces/classes describing your data shapes (e.g. `User`, `Order`).
`src/app/guards/`	Functions that control whether a route can be entered (e.g. auth checks).
`src/environments/`	Environment-specific config (dev vs. production API URLs).

7. Quick Mental Model

- **Root files** (`angular.json`, `package.json`, `tsconfig*.json`) — configure *how the project builds/runs*. You rarely touch these day-to-day.
- `src/` — everything you actually write goes here.
- `src/app/` — your application's components, routes, and app-wide setup.
- **Four-file component pattern** — class (`.ts`), template (`.html`), styles (`.css`), tests (`.spec.ts`).