Angular

Angular commands to install the CLI, create a project, and start it:

Step-by-Step Angular Setup:

1. Install Angular CLI (one-time only):

**npm install -g @angular/cli**

This installs the Angular CLI globally so you can use ng command anywhere.

2. Create a new Angular project:

**ng new my-angular-app**

• It will ask:

• Do you want to add routing? (Choose y or n)

• Which stylesheet format? (Choose CSS, SCSS, etc.)

3. Go into the project folder:

**cd my-angular-app**

4. Start the development server:

**ng serve**

Or with a specific port:

**ng serve --port 4300**

Then, open your browser and go to: <http://localhost:4200>

⸻

Angular Project Structure Overview:

**ng new angular-app** command run chesaka, Angular CLI project structure ni neatly create chestundi.

angular-app/

│

├── node\_modules/ --> All installed packages

├── src/ --> Main source code of your app

│ ├── app/ --> All your components, services, modules

│ │ └── app.component.ts / html / css / spec.ts

│ ├── assets/ --> Images, icons, static assets

│ ├── environments/ --> Environment configs (dev, prod)

│ ├── index.html --> Main HTML file (single-page)

│ ├── main.ts --> Entry point for Angular app

│ ├── styles.css --> Global styles

│ └── polyfills.ts --> Browser compatibility

│

├── angular.json --> Angular CLI config (build, serve settings)

├── package.json --> Dependencies and npm scripts

├── tsconfig.json --> TypeScript configuration

├── .editorconfig / .gitignore / README.md etc.

Important Folders and Files Explained:

Path/File Purpose

src/app/ Where you write your app’s logic (components, modules, services).

app.component.ts Root component of your Angular app.

index.html Only HTML file loaded in the browser. Angular loads inside this.

main.ts Entry point that bootstraps the AppModule.

angular.json Controls how the app builds, serves, and what assets are included.

package.json Lists npm packages and scripts (like ng serve, ng build).

environments/ Separate files for dev and prod configurations.

assets/ Public images, icons, etc.

⸻

**ng serve** ante em avthundi?

• ng serve development server ni start chestundi.

• App ni compile chesi, local server lo http://localhost:4200 lo run chestundi.

• Files browser lo live ga chupistayi.

• Changes chesinappudu auto reload avutundi (Hot Reloading).

• Build files disk lo save avavu, memory lo temporary ga untaayi.

**Use case:** App ni develop chese stage lo use cheyyali.

**ng build** ante em avthundi?

• ng build app ni compile & bundle chestundi.

• Output ni **dist/your-project-name/** folder lo static files ga save chestundi:

* index.html
* main.js
* styles.css etc.

• Production ki ready ga untundi (especially with --configuration production)

• Ivi browser lo serve cheyyali using a static server.

**Use case:** App ni deploy cheyyali ante **ng build** cheyyali.

Simple Analogy:

**Command** **Purpose** **Result**

ng serve Development/testing while coding Runs app at localhost:4200

ng build Creates deployable production files Outputs files in **dist/** folder

NOTE: **ng build alone doesn’t start a server**.

⸻

Difference between **ng build** and

**ng build --configuration production** (formerly **--prod**):

1. ng build (default)

• Builds the app in development mode

• No optimizations

• Bigger file sizes

• Source maps included (helps debugging)

• Suitable only for local testing

2. ng build --configuration production (or ng build --prod ⚠️ Works in older Angular, but deprecated in newer versions)

• Builds the app in production mode

• Optimizations enabled:

• Minification (removes white space, shortens variable names)

• Dead code elimination (removes unused code)

• Tree-shaking (removes unused modules)

• Ahead-of-Time (AOT) compilation

• Disables debug tools

• Smaller file sizes

• Faster performance

• Ready for deployment to server or hosting

ng build // Dev build

ng build --configuration production // Prod build (optimized)

⸻

React (CRA) or Angular CLI project create chesinappudu:

1. React using CRA (npx create-react-app)

• CRA lo npm run start ane command development server ni start chestundi using react-scripts.

• But when you run npm run build, it just creates a build/ folder with static files. serve ane package install cheyyadu.

• If you want to serve the build/ folder, you have to manually install a static server

Which serves static files.

2. Angular using Angular CLI (ng new)

• Angular lo ng serve command development server start chestundi using Webpack Dev Server (built-in).

• But after ng build, it creates the dist/ folder only.

• Again, serve package install cheyyadu automatically.

So, if you want to serve the built files after **npm run build** or **ng build**, you have to install a static file server manually like serve, or use Python/VS Code extension.

⸻

**Serve Angular build locally after ng build:**

ng build taruvata files ni **serve** command tho local ga test cheyyadam ela?

No need to install a permanent server, just to serve a built React or Angular app locally. But oka lightweight server temporary ga run cheyyali to serve the files in the browser.

1. Build the Angular/React app

Angular: **ng build --configuration=production**

React : **npm run build**  
This creates files in dist/<your-project-name>/ or build/

React or Angular lo **build** chesaka, output files (like index.html, main.js, etc.) **build/** or **dist/** folder lo untayi. Ivi static files. Browser ki serve cheyyali ante, oka **static file server** kavali.

2. Serve the **dist/** folder using a local server.

You can use any static server. Here are **3 easy options**:

* Use **serve** package (React or Angular ki suitable)

**npm install -g serve**

Serve the build/ folder for React:

**serve -s build**

Serve the dist/ folder for Angular:

**serve -s dist/project-name**

Open browser:

[**http://localhost:3000**](http://localhost:3000)

Stops server with Ctrl + C

* Using http-server (from npm)

Install it globally (if not already):

**npm install -g http-server**

Serve the build/ folder for React:

**cd build**

Serve the **dist/** folder for Angular:

**cd dist/<your-project-name>**

Server gets start when you run:

**http-server**

serve files (like index.html, main.js, styles.css) from **dist/** folder.

Open browser:

<http://localhost:8080>

Stops with Ctrl + C

* Using live-server (auto-reloads)

**npm install -g live-server**

**cd dist/<your-project-name>**

Server gets start when you run:

**live-server**

serve files (like index.html, main.js, styles.css) from **dist/** folder.

Open browser:

<http://127.0.0.1:8080> or http://localhost:8080

Stops server with Ctrl + C

* Use Python (built-in server, no installation needed if Python is installed)

**cd dist/<your-project-name>**

**python -m http.server 8080**

Open browser:

<http://localhost:8080>

NOTE:

A lightweight server must run in your system, but install cheyyadam (like Apache or IIS) compulsory kaadu. Temporary ga serve, Python, or VS Code Live Server use cheyyochu.

servers like http-server, live-server, serve are static file servers — they **don’t care whether it’s Angular, React, or plain HTML/CSS/JS**. They just serve files (like index.html, main.js, styles.css) from a folder.

If you're just **testing final production build**, use these static servers.

If you're **actively coding and want fast reloads**, use:

* ng serve (Angular)
* npm start (React)

⸻

Ledu, neeku permanent server install cheyyadam avasaram ledu just to serve a built React or Angular app locally. But oka lightweight server temporary ga run cheyyali to serve the files in the browser.

Here’s how it works:

React or Angular lo build chesaka, output files (like index.html, main.js, etc.) build/ or dist/ folder lo untayi. Ivi static files. Browser ki serve cheyyali ante, oka static file server kavali.

Option 1: Use serve package (React or Angular ki suitable)

1. Install once using npm:

npm install -g serve

2. Serve build folder:

• React:

serve -s build

• Angular:

serve -s dist/project-name

3. Ippudu http://localhost:3000 lo open avutundi.

Option 2: Use Python (built-in server, no installation needed if Python is present)

If Python install chesuntee:

cd build

python -m http.server 3000

Conclusion:

Yes, a lightweight server must run in your system, but install cheyyadam (like Apache or IIS) compulsory kaadu. Temporary ga serve, Python, or VS Code Live Server use cheyyochu.

Kavala ante, neeku lightweight permanent server setup (like Nginx or IIS) explain cheyyacha?