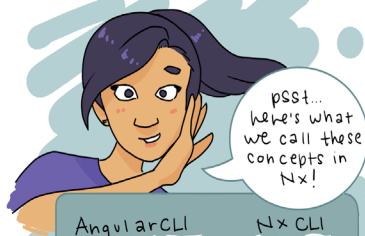




FOR A *modern* INTEGRATED DEVELOPER EXPERIENCE



psst...
here's what
we call these
concepts in
Nx!

AngularCLI		Nx CLI
architect	➡	targets
builder	➡	executor
schematic	➡	generator



here's what
you get **out of
the box** when
you generate
a new angular
app with Nx...



...and there's
even more
tooling
integrations
available via
generators!

LOCAL COMPUTATION CACHING

Nx stores the results of your targets' computations (terminal output & artifacts) and replays them — speeding up code verification and builds!

Guess I'm
going to have
to rebuild
this...



no need!
we already built
this version, here
it is!

→ Read more at
<https://nx.dev/technologies/angular>

Nx is growing quickly,
with over
24 MILLION
NPM downloads
/ month

come join us!
- your friends
@Nx

drawn by @nxcodes

A quick guide to Nx + Angular

Getting Started

Create a new Nx workspace to host your Angular application.

```
npx create-nx-workspace <workspace-name> --preset=angular
```

Install Nx globally or use npx

You can install Nx globally with: `npm i nx -g`

Or just prefix your commands with npx.

NX CONSOLE VSCODE EXTENSION

Use the dedicated VSCode extension to explore and run generators and navigate the workspace much more easily. <https://nx.dev/getting-started/editor-setup>

Run tasks with Nx

```
nx <target> <project-name>
```

Example: Run tests for my-project: `nx test my-project`

Run only affected Projects

Nx allows only running targets like test, lint, build on projects that have been changed in a given PR, using so-called “affected commands”.

```
nx affected --target=<target>
```

Example: run all tests for the projects that have been changed in the current PR:

```
nx affected: --target=test
```

Setup Remote Distributed Caching

All Nx workspaces come with local caching abilities, but the real benefit comes when this cache is distributed and shared with other team members and your CI servers. This is what Nx Cloud does. Here's how to connect your Nx workspace:

```
nx connect
```

<https://nx.dev/features/remote-cache>

NX CLOUD FOR SMART, SCALABLE, SELF-HEALING CI

Nx Cloud's remote cache and automatic task distribution speed up CI, while its self-healing system automatically reruns flaky tests and fixes code issues. It's free to start and has flexible pricing for teams of all sizes.

Setup Distributed Task Execution on CI

Generate a complete CI configuration that uses Nx Cloud's distributed task execution to optimally parallelize tasks focusing on maximum agent utilization!

```
nx generate @nx/workspace:ci-workflow --ci=github
```

<https://nx.dev/ci/features/distribute-task-execution>

💡 GENERATORS & EXECUTORS, AKA SCHEMATICS & BUILDERS

Nx uses a slightly different terminology:

- Angular Schematics -> Nx Generators
- Angular Builders -> Nx Executors

See <https://nx.dev/technologies/angular/recipes/nx-devkit-angular-devkit>

Generators (aka Schematics)

```
nx generate <package-name>:<generator> <options>
```

Tip: use `--dry-run` to simulate the execution of the generator.

Generate a new Application

```
nx g @nx/angular:application <name>
```

Generate a new Library

```
nx g @nx/angular:lib libs/libName
```

💡 APPS AND LIBS

Nx recommends having thin applications and moving features into dedicated workspace libraries. This helps with establishing a cleaner, more defined architecture as well as speeding up CI times..

Generate a component

Navigate to the directory where you want the component generated and run:

```
nx g @nx/angular:component hello-world
```

💡 GENERATORS AT YOUR FINGERTIPS

With Nx Console installed, you can right-click on a project in Visual Studio Code and directly invoke the `Nx Generate` menu item.

Installing Angular Packages

Nx doesn't come with a default `ng add` command. Here's how you can still use it:

```
npm i @angular/material @angular/cdk
```

```
npx nx g @angular/material:ng-add --project=<project name>
```

Generate a Module Federation Setup

Looking into Microfrontends or just want to speed up your Angular app builds? Here's how to generate a new Webpack Module Federation app:

```
nx g @nx/angular:host shell --remotes=featureA,featureB,featureC
```

<https://nx.dev/technologies/module-federation/concepts/faster-builds-with-module-federation>

Powerful Integrations: Cypress, Playwright, Storybook, Tailwind

You can choose an e2e tool when creating your workspace. You can run e2e tests with:

```
nx e2e <project-name>-e2e
```

Add **Storybook** support with a single command, including generating stories for components and setting up Cypress-based e2e tests for them.

```
nx g @nx/angular:storybook-configuration <project-name>
```

Add **Tailwind** support by running:

```
nx g @nx/angular:setup-tailwind <project-name>
```

Automated updates

```
nx migrate latest
```

Migrate from the Angular CLI

You can easily migrate from an Angular CLI project using the following command:

```
npx nx@latest init
```

For more complex scenarios, go to:

<https://nx.dev/technologies/angular/migration/angular>

💡 LEARN MORE!



Nx website: <https://nx.dev>



Nx on Github: <https://github.com/nrwl/nx>



Nx Community Discord: <https://go.nx.dev/community>



Nx YouTube Channel: <https://www.youtube.com/@nxdevtools>



Nx Twitter: <https://twitter.com/nxdevtools>