Yarn Node package manager

START HERE

`yarn install`
Install all dependencies

`yarn dlx <pkg>`
Run a package once

`yarn add <pkg>`
Add a dependency

`yarn up <pkg>`
Upgrade a dependency

`yarn remove <pkg>`
Remove a dependency

`yarn workspaces foreach -A run <script>`
Run in every workspace

`yarn run <script>`
Run a package script

`yarn install --immutable`
Clean install for CI

package.json ranges you want → `resolve` → **yarn.lock** exact versions → `fetch` → **Cache** zipped packages → `link` → **Install** PnP or node_modules

CONCEPTS

Yarn 4 is set per project: yarn set version stable writes the version into package.json (packageManager), and Corepack picks it up.

By default Yarn 4 uses Plug'n'Play and writes no node_modules; set nodeLinker: node-modules in .yarnrc.yml if a tool needs the folder.

Settings live in .yarnrc.yml, not .npmrc; every one can also be set as a YARN_* environment variable.

INSTALL AND ADD

`yarn install`
Install all dependencies

`yarn add <pkg>`
Add a dependency

`yarn add -D <pkg>`
Add a dev dependency

`yarn add <pkg>@<version>`
Add a specific version or tag

`yarn add -E <pkg>`
Save the exact version

`yarn install --immutable`
Fail if the lockfile would change
Already the default on CI, so a plain yarn install behaves the same there.

`yarn install --mode=skip-build`
Install without running build scripts

REMOVE AND UPDATE

`yarn remove <pkg>`
Remove a dependency

`yarn up <pkg>`
Upgrade a package

`yarn up "<pkg>@^2"`
Upgrade to a given range

`yarn up "*"`
Upgrade everything

REMOVE AND UPDATE (cont.)

`yarn up -i`
Pick upgrades interactively

`yarn dedupe`
Merge duplicate versions in the lockfile

`yarn set version stable`
Pin the latest stable Yarn

`yarn set version <version>`
Pin an exact Yarn version

RUN SCRIPTS

`yarn run <script>`
Run a package script

`yarn <script>`
Shorthand for yarn run
Built-in commands win when a script has the same name; use yarn run then.

`yarn run <script> <args>`
Pass arguments to a script

`yarn run <bin>`
Run a dependency's binary

`yarn dlx <pkg> [args]`
Fetch and run without installing

`yarn dlx -p <pkg> <cmd>`
Run a command from a package

`yarn exec <cmd>`
Run a shell command in the Yarn environment

`yarn node <file>`
Run Node with Yarn's loader

`yarn init`
Create a package.json

`yarn init -w`
Create a workspace root

INSPECT

`yarn info <pkg>`
Show info about a package

`yarn info <pkg> --dependents`
Show who depends on a package

INSPECT (cont.)

`yarn why <pkg>`
Show why a package is installed

`yarn npm audit`
Check for known vulnerabilities

`yarn config`
Show the current settings

`yarn config get <key>`
Read a setting

WORKSPACES

`yarn workspaces list`
List all workspaces

`yarn workspace <name> <cmd>`
Run a command in one workspace

`yarn workspace <name> add <pkg>`
Add a dependency to one workspace

`yarn workspaces foreach -A run <script>`
Run in every workspace

`yarn workspaces foreach -Apt run <script>`
Parallel, dependencies first

`yarn workspaces foreach -R run <script>`
Current workspace and its dependencies

`yarn workspaces foreach --since=<ref> -A run <script>`
Only workspaces changed since a ref

`yarn workspaces foreach --exclude <glob> -A <cmd>`
Skip workspaces matching a glob

`yarn workspaces focus <name>`
Install one workspace and its dependencies

Workspaces in package.json

```
{
  "private": true,
  "workspaces": ["packages/*"]
}
```

WORKSPACES (cont.)

Depend on a sibling with "<pkg>": "workspace:^"; Yarn swaps it for the real version when publishing.

CONFIG, PATCHING AND PUBLISHING

`yarn config set <key> <value>`
Write a project setting

`yarn cache clean`
Delete the local cache

`yarn patch <pkg>`
Extract a package to edit

`yarn patch-commit -s <dir>`
Save the edits as a patch

`yarn link <path>`
Link a local project in

`yarn npm publish`
Publish the workspace

`yarn npm publish --access public`
Publish a scoped package publicly

.yarnrc.yml

```
nodeLinker: node-modules
npmRegistryServer: "https://
registry.npmjs.org"
enableTelemetry: false
```

RECIPES

Start a project

```
yarn set version stable
yarn init
yarn add express
yarn add -D typescript vitest
yarn run test
```

CI install and test

```
yarn install --immutable
yarn workspaces foreach -Apt
run build
yarn workspaces foreach -A run
test
```

Test only what changed

```
yarn workspaces foreach -A --
since=origin/main run test
```

Patch a dependency

```
yarn patch <pkg>
# edit the printed folder
yarn patch-commit -s <dir>
```