npm Node package manager

START HERE

`npm install`
Install all dependencies

`npm ci`
Clean install from lockfile

`npm install <pkg>`
Add a dependency

`npx <pkg>`
Run a package binary

`npm run <script>`
Run a package script

`npm outdated`
What is behind?

`npm test`
Run the tests

`npm audit`
Check for vulnerabilities

package.json version ranges → `npm install` → **package-lock.json** exact versions → `npm ci` → **node_modules** installed tree → `npm run` → **Scripts** build, test, start

CONCEPTS

package.json declares what you want, package-lock.json pins what you got. Commit both.

Use npm ci in CI: it installs exactly the lockfile and fails if package.json disagrees.

yarn and pnpm read the same package.json but use their own lockfiles and commands. Do not mix them in one project.

INSTALL

`npm install`
Install all dependencies
Reads package.json and updates the lockfile if needed.

`npm install <pkg>`
Add a dependency
Saved under dependencies by default.

`npm install -D <pkg>`
Add a dev dependency

`npm ci`
Clean install from the lockfile
Deletes node_modules first and never rewrites the lockfile.

`npm install <pkg>@<version>`
Install a specific version or tag
A tag works too: <pkg>@latest or <pkg>@next.

`npm install -g <pkg>`
Install globally

`npm uninstall <pkg>`
Remove a dependency

`npm install -E <pkg>`
Save an exact version
Skips the ^ range in package.json.

`npm ci --omit=dev`
Install production deps only

`npm install --package-lock-only`
Update lockfile only

`npm prune`
Remove extraneous packages

`npm dedupe`
Flatten duplicate packages

`npm rebuild`
Rebuild native addons

SCRIPTS & NPX

`npm run <script>`
Run a package.json script

`npm run`
List available scripts

SCRIPTS & NPX (cont.)

`npm test`
Run the test script
Shorthand for npm run test; npm start works the same.

`npm start`
Run the start script

`npm run <script> -- <args>`
Pass arguments to the script
Everything after -- goes to the script, not to npm.

`npx <pkg> [args]`
Run a package binary
Uses a local binary, else downloads it temporarily.

`npm exec -- <cmd>`
Run a command from the local tree

`npm run <script> --if-present`
Skip silently if missing

`npm init -y`
Create package.json with defaults

`npm init <initializer>`
Scaffold from a create-* package
npm init vite runs create-vite.

`npm pkg set <key>=<value>`
Edit package.json from the shell

INSPECT & UPDATE

`npm ls`
Show the dependency tree
Add --depth=0 for top-level packages only.

`npm ls <pkg>`
Who depends on this package?

`npm outdated`
Show packages with newer versions

`npm update`
Update within allowed ranges
Respects the ranges in package.json; it will not cross a major version.

`npm update -g`
Update global packages

`npm view <pkg> versions`
List published versions

`npm view <pkg> version`
Latest published version

`npm search <term>`
Search the registry

`npm explain <pkg>`
Why is this installed?

`npm doctor`
Check the environment

`npm root -g`
Global install directory

AUDIT

`npm audit`
List known vulnerabilities

`npm audit fix`
Apply compatible fixes
Stays within semver; --force may install breaking upgrades.

`npm audit --omit=dev`
Ignore dev dependencies

`npm audit --audit-level=high`
Fail only at high severity
Exit code is non-zero when findings reach that level, useful in CI.

`npm audit signatures`
Verify registry signatures

`npm fund`
Show funding links

VERSION & PUBLISH

`npm version patch`
Bump the patch version
Also commits and tags when in a git repo. Use minor or major as needed.

`npm version <x.y.z>`
Set an exact version

`npm version patch --no-git-tag-version`
Bump without commit and tag

`npm pack --dry-run`
Preview the tarball contents

`npm login`
Log in to the registry

`npm publish`
Publish the package

`npm publish --access public`
Publish a scoped package publicly
Scoped packages default to private.

`npm publish --dry-run`
Check a publish without sending

`npm publish --tag <tag>`
Publish under a dist-tag

`npm dist-tag ls`
List dist-tags

`npm deprecate <pkg>@<range> "<msg>"`
Mark versions deprecated

`npm link`
Symlink a local package
Run in the package, then npm link <pkg> in the consumer.

`npm whoami`
Show the logged-in user

WORKSPACES

`npm install -w <name>`
Install in one workspace

WORKSPACES (cont.)

`npm run <script> -w <name>`
Run a script in one workspace

`npm run <script> -ws`
Run in all workspaces

`npm run <script> -ws --if-present`
Skip workspaces without the script

`npm init -w <dir>`
Create a new workspace

`npm ls -ws`
List workspace packages

CONFIG

`npm config list`
Show effective settings

`npm config get <key>`
Read one setting

`npm config set <key>=<value>`
Write a setting
Goes to the user ~/.npmrc; use --location=project for ./.npmrc.

`npm config delete <key>`
Remove a setting

`npm config set registry <url>`
Use another registry

`npm cache verify`
Check and tidy the cache

`npm cache clean --force`
Wipe the cache

`npm help <term>`
Open the docs for a topic

`npm <command> -h`
Short usage summary

RECIPES

New project

```
npm init -y
npm install express
npm install -D typescript
npm pkg set scripts.start="node index.js"
```

CI install and test

```
npm ci
npm run build --if-present
npm test
npm audit --audit-level=high
```

Upgrade, then release

```
npm outdated
npm update
npm test
npm version minor
npm publish
```

Typical scripts block

```
# package.json
"scripts": {
  "build": "tsc",
  "test": "node --test",
  "start": "node dist/index.js"
}
```