

npm Quick Reference

Package management, scripts, versioning, publishing, workspaces

Installation

Installing npm & Node

```
node -v && npm -v # check installed versions
npm install -g npm@latest # update npm itself
nvm install --lts # install Node LTS via nvm
nvm use 20 # switch to Node 20
```

Install Commands

```
npm install # Install all dependencies from package.json
npm install pkg # Add package as dependency
npm install -D pkg # Add package as devDependency
npm install -g pkg # Install package globally
npm install pkg@2.1.0 # Install specific version
npm ci # Clean install from lock file (CI/CD)
npm uninstall pkg # Remove package
```

Package Management

Managing Packages

```
npm ls # list installed packages
npm ls --depth=0 # top-level only
npm outdated # check for newer versions
npm update # update within semver range
npm audit # check for vulnerabilities
```

Management Commands

```
npm ls # List installed packages as tree
npm outdated # Show packages with newer versions
npm update [pkg] # Update packages within semver range
npm audit # Audit dependencies for vulnerabilities
npm audit fix # Auto-fix vulnerable dependencies
npm prune # Remove extraneous packages
npm dedupe # Flatten dependency tree to reduce duplication
```

Scripts

Running Scripts

```
npm run build # run "build" script
npm test # shortcut for "test" script
npm start # shortcut for "start" script
npm run lint -- --fix # pass args to script
npm run dev & # run in background
```

Script Lifecycle

```
npm test / npm t # Run scripts.test
npm start # Run scripts.start
npm run <name> # Run any custom script
pre<name> # Runs automatically before <name>
post<name> # Runs automatically after <name>
npm run # List all available scripts
```

package.json

Initialize & Fields

```
npm init # interactive setup
npm init -y # accept all defaults
npm pkg set name="my-app" # set a field
npm pkg get version # read a field
```

Key Fields

```
name # Package name (lowercase, no spaces)
version # Current version (semver: major.minor.patch)
main # Entry point for CommonJS (require)
module # Entry point for ES modules (bundlers)
type # "module" for ESM, "commonjs" for CJS (default)
scripts # Named commands (build, test, start, etc.)
dependencies # Production dependencies
devDependencies # Development-only dependencies
engines # Required Node/npm version ranges
```

Versioning

Version Commands

```
npm version patch # 1.0.0 -> 1.0.1
npm version minor # 1.0.1 -> 1.1.0
npm version major # 1.1.0 -> 2.0.0
npm version 3.2.1 # set explicit version
npm version prerelease --preid=beta # 1.0.0-beta.0
```

Semver Ranges

```
^1.2.3 # Compatible: >=1.2.3 <2.0.0 (default)
~1.2.3 # Patch-level: >=1.2.3 <1.3.0
1.2.3 # Exact version only
>=1.0.0 <2.0.0 # Explicit range
* # Any version
1.x / 1.2.x # Wildcard ranges
latest # Latest published version tag
```

Publishing

Publish Workflow

```
npm login # authenticate to registry
npm publish # publish public package
npm publish --access public # scoped package as public
npm unpublish pkg@1.0.0 # remove specific version
npm deprecate pkg@"<2" "Use v2+" # deprecate old versions
```

Publish Reference

```
npm login # Authenticate with npm registry
npm publish # Publish package to registry
npm pack # Create tarball without publishing
npm unpublish # Remove published version (within 72h)
npm deprecate # Mark versions as deprecated
.npmignore # Files to exclude from published package
files (package.json) # Allowlist of files to include in package
```

Workspaces

Workspace Commands

```
npm init -w packages/core # create workspace
npm install -w packages/core lodash # install in workspace
npm run build --workspaces # run in all workspaces
npm run test -w packages/api # run in specific workspace
npm ls --workspaces # list workspace deps
```

Workspace Config

```
workspaces (package.json) # Array of workspace globs: ["packages/*"]
-w / --workspace # Target a specific workspace
--workspaces # Run command across all workspaces
--include-workspace-root # Include root package in workspace operations
npm install (root) # Installs all workspace dependencies
Hoisting # Shared deps hoisted to root node_modules
```

npx

Running with npx

```
npx create-react-app my-app # run without installing
npx tsc --init # run local or remote bin
npx -p typescript tsc file.ts # specify package explicitly
npx --yes create-next-app # skip install prompt
npx node@18 -e "console.log('hi!')" # run with specific Node
```

npx Options

```
npx cmd # Run cmd from local node_modules/.bin or remote
npx -p pkg cmd # Install pkg, then run cmd
npx --yes cmd # Auto-confirm installation prompt
npx --no cmd # Refuse installation — fail if not local
npx -c 'cmd' # Run shell command with npx PATH
npx node@ver # Run specific Node.js version
```

Configuration

Config Commands

```
npm config list # show current config
npm config set registry https://r.npmjs.com/
npm config set init-author-name "Name"
npm config get prefix # global install path
npm config delete key # remove a config value
```

Config Reference

```
.npmrc (project) # Per-project config file
~/ .npmrc # Per-user config file
registry # Package registry URL
save-exact # true to pin exact versions on install
engine-strict # true to enforce engines field
fund # false to suppress funding messages
audit # false to skip audit on install
```

Common Patterns

One-Liners

```
npm ls --depth=0 --json | jq '.dependencies | keys[]'
npm outdated --long # show type and homepage
npm cache clean --force # clear npm cache
npm explain pkg # why is pkg installed?
npm exec -- envinfo --system # system info for bug reports
```

Recipes

```
Lock file only # npm ci — clean install from package-lock.json
Check licenses # npx license-checker --summary
Find unused deps # npx depcheck
Bundle size # npx bundlephobia-cli pkg — check package size
Upgrade all # npx npm-check-updates -u && npm install
Local registry # npx verdaccio — run private registry
```