HOW SOFTWARE WORKS · BEGINNER · 14 MIN READ

npm and packages: what npm install really does

What package.json and the lockfile are for, the npm commands you'll run every week, how version numbers work, and how to stay safe when an AI suggests a package.

Pass the check (4 of 5) online to earn the “Package Pro” badge:
https://digitalbuilders757.com/university/study/npm-packages-and-dependencies

Packages and npm

Almost every JavaScript project is built from packages: code other people wrote and shared so you don't have to, like React, Next.js or a library for dates. npm is the public registry where millions of packages are published, and also the command-line tool that comes with Node.js for installing them.

Packages depend on other packages, so a new app made by an AI builder or coding agent often pulls in hundreds. You don't need to know them all, but you do need to know where they're listed, how they're installed, and how to tell a good one from a bad one.

package.json: your project's ID card

Open package.json in any JavaScript project and you'll find:

- scripts: named commands. npm run dev starts the app on localhost, npm run build makes the production version, and npm test runs the tests.
- dependencies: packages the app needs to run, with the versions it accepts, like "next": "^16.0.0".
- devDependencies: tools needed only while you build and test, like TypeScript, ESLint or a test runner.
- Version numbers are MAJOR.MINOR.PATCH (semantic versioning). A patch fixes bugs, a minor version adds features, and a major version may break code that used the old one.
- ^4.2.1 (npm's default) accepts 4.2.1 or any later 4.x, but never 5.0.0. ~4.2.1 accepts only 4.2.x patches. 4.2.1 on its own means exactly that version.

node_modules and the lockfile

npm install reads package.json, picks versions that fit each range, downloads them into a folder called node_modules, and writes the exact versions it chose into package-lock.json.

Commit package.json and package-lock.json to Git. The lockfile is how your laptop, a teammate and your host end up with the same versions. Never commit node_modules: it's huge, and npm install rebuilds it. A .gitignore file keeps it out.

Hosts and CI usually run npm ci (clean install) instead. It deletes node_modules, installs exactly what the lockfile says, and stops with an error if package.json and the lockfile disagree, rather than quietly changing either one.

Commands you'll use every week

- npm install: install everything the project lists. Run it after you clone a project or pull changes.
- npm install zod adds a package to dependencies; npm install -D vitest adds a devDependency; npm uninstall zod removes one.
- npm run dev, npm run build, npm test: run the project's scripts. npm run on its own lists them all.
- npx create-next-app@latest: run a package's command without adding it to your project.
- npm outdated: see which packages have newer versions. npm update installs the newest versions your ranges allow.
- npm audit: check your installed versions against known security problems. npm audit fix applies safe updates; --force can make breaking major upgrades, so use it only when you understand what will change.

Before you install a package an AI suggests

AI models sometimes invent package names that sound right. A 2025 study of 16 code-writing models (USENIX Security) found that at least 5% of the packages suggested by commercial models, and about 22% from open-source models, didn't exist, and many made-up names came back again and again. Attackers publish packages under those names with harmful code inside. It's called slopsquatting, a cousin of typosquatting, which uses look-alike names one letter off a popular package.

- Look a new package up on npmjs.com before you install it: the exact spelling, a real source repository, recent releases, lots of weekly downloads, and the name the official docs use.

- Repeating the question proves nothing. If an AI suggests the same name twice, it can still be made up.
- Fewer packages, fewer risks. Don't add one for something a few lines of code can do.
- Packages can include install scripts that run on your computer when they're installed. From npm 12 (July 2026), npm skips them unless you approve them (npm approve-scripts). Approve only packages you trust that need one, such as packages that build native code.
- When an AI agent edits package.json or the lockfile, review what it added like any other change.
- Update regularly: run npm outdated and npm audit, or let GitHub's Dependabot open update pull requests, and let your tests and CI check each one.

Reading npm errors

- command not found: npm means Node.js isn't installed. Install the LTS version from nodejs.org.
- Module not found or Cannot find module 'x': the package isn't installed. Run npm install, or npm install x if it's missing from package.json.
- Missing script: "dev": package.json has no script with that name. Run npm run to list the real ones, and check you're in the project's folder.
- ERESOLVE unable to resolve dependency tree: two packages want clashing versions of a third. Read the 'peer' line, then look for a newer version of the older package before you reach for --force or --legacy-peer-deps, which only hide the clash.
- npm ci says package.json and package-lock.json aren't in sync: run npm install on your computer, then commit and push both files.

Key terms

npm / package
npm is the registry of reusable JavaScript code; each reusable piece is a package.

package.json
A project's ID card: its name, its scripts and the packages it depends on, with the versions it accepts.

Dependency
A package your project needs, listed in package.json. Packages have dependencies of their own too.

devDependency
A package needed only while you build or test, like TypeScript or a linter, not by the running app.

node_modules
The folder where npm puts installed packages. It's big and npm install rebuilds it, so it stays out of Git.

Lockfile (package-lock.json)
Records the exact version of every installed package, so every computer installs the same. Commit it.

Semantic versioning (semver)
Version numbers as MAJOR.MINOR.PATCH: patches fix bugs, minor versions add features, major versions can break things.

npm script
A named command in package.json's scripts, run with npm run, like npm run dev or npm run build.

Slopsquatting
Publishing a package under a name AI tools tend to make up, so people who install the AI's suggestion get the attacker's code.

Supply-chain risk
Danger from the packages and tools your project depends on, including fake look-alike packages.

Try it: Look inside a real project

1. Open package.json in any JavaScript project (or make one with npm init -y in an empty folder). Find the scripts, dependencies and devDependencies.
2. Run npm install, then find the exact version of one package in package-lock.json. Does it fit the range in package.json?
3. Add a small package with npm install zod. Use git diff to see exactly what changed in package.json and the lockfile.
4. Run npm outdated and npm audit, and ask your AI helper to explain the output in plain words.
5. Ask an AI which package to use for a job, then check its answer on npmjs.com before installing anything.

KNOWLEDGE CHECK

npm and packages: what npm install really does

5 questions. Circle one answer for each; 4 or more correct passes. Answers are at the back.

1. What is package-lock.json for?

A. It locks package.json with a password, so only the project's owner can change dependencies
B. It records the exact version of every package installed, so installs match everywhere
C. It keeps a copy of your npm sign-in so installs on a new computer don't ask you to log in
D. It lists the packages you plan to add later, and npm installs them once you approve

2. package.json says "react": "^19.1.0". Which version could npm install?

A. 20.0.0
B. 18.3.1
C. 19.2.4
D. Only 19.1.0, exactly

3. You just cloned a project from GitHub. What do you run before npm run dev?

A. git push, so GitHub knows the project is on your computer
B. npm audit fix --force, so every package is on its newest version
C. Nothing: node_modules comes with the clone
D. npm install, to download the packages in package.json

4. An AI assistant tells you to install a package you've never heard of. What first?

A. Look it up on npmjs.com: exact spelling, a real repository, recent releases, downloads
B. Install it straight away: if the name were made up, npm would refuse to install it
C. Ask the same AI again, and if it gives the same name twice, the package must be real
D. Install it with sudo, so the package can't change anything else on your computer

5. Which files from npm belong in Git?

A. node_modules, so nobody has to run npm install again
B. package.json and package-lock.json, but not node_modules
C. Only package.json, because each computer makes its own lockfile
D. Nothing from npm: hosts work out what to install by reading your code

Official sites

npm docs: About semantic versioning: https://docs.npmjs.com/about-semantic-versioning

npm docs: npm ci: https://docs.npmjs.com/cli/v12/commands/npm-ci
npm docs: install scripts and approvals: https://docs.npmjs.com/cli/v12/commands/npm-install-scripts
USENIX Security 2025: package hallucinations by code-generating AI:
https://www.usenix.org/conference/usenixsecurity25/presentation/spracklen
OpenSSF: npm best practices:
https://github.com/ossf/package-manager-best-practices/blob/main/published/npm.md

Last reviewed 2026-11-25. AI tools change fast; check each tool's own site for current features, plans and prices.

ANSWER KEY

npm and packages: what npm install really does

1. B: It records the exact version of every package installed, so installs match everywhere
npm writes the exact version it chose for every package. Commit the lockfile so teammates, CI and your host install the same thing.

2. C: 19.2.4
^ accepts later minor and patch versions (19.x from 19.1.0 up) but not 20.0.0, a major version that may break your code.

3. D: npm install, to download the packages in package.json
node_modules isn't in Git. npm install reads package.json and the lockfile and downloads everything the project needs.

4. A: Look it up on npmjs.com: exact spelling, a real repository, recent releases, downloads
AI tools sometimes invent package names, and attackers publish packages under them (slopsquatting). The same fake name often comes back, so repetition proves nothing.

5. B: package.json and package-lock.json, but not node_modules
package.json says what you need and the lockfile pins exact versions. node_modules is rebuilt by npm install and stays out of Git.