



Linux Ubuntu 26.04 Manual Pages on command 'npm-outdated.1'

\$ man npm-outdated.1

NPM-OUTDATED(1)

General Commands Manual

NPM-OUTDATED(1)

NAME

npm-outdated

Synopsis

```
<!-- AUTOGENERATED USAGE DESCRIPTIONS -->
```

Description

This command will check the registry to see if any (or, specific) installed packages are currently outdated.

By default, only the direct dependencies of the root project and direct dependencies of your configured workspaces are shown.

Use --all to find all outdated meta-dependencies as well.

In the output:

? wanted is the maximum version of the package that satisfies the semver range specified in package.json. If there's no available semver range (i.e. you're running npm outdated --global, or the package isn't included in package.json), then wanted shows the currently-installed version.

? latest is the version of the package tagged as latest in the registry.

Running npm publish with no special configuration will publish the package with a dist-tag of latest. This may or may not be the maximum version of the package, or the most-recently published version of the package, depending on how the package's developer manages the latest dist-tag.

- ? location is where in the physical tree the package is located.
- ? depended by shows which package depends on the displayed dependency
- ? package type (when using --long / -l) tells you whether this package is a dependency or a dev/peer/optional dependency. Packages not included in package.json are always marked dependencies.
- ? homepage (when using --long / -l) is the homepage value contained in the package's packument
- ? Red means there's a newer version matching your semver requirements, so you should update now.
- ? Yellow indicates that there's a newer version above your semver requirements (usually new major, or new 0.x minor) so proceed with caution.

An example

```
$ npm outdated
```

Package	Current	Wanted	Latest	Location	Depended by
glob	5.0.15	5.0.15	6.0.1	node_modules/glob	dependent-package-name
nothingness	0.0.3	git	git	node_modules/nothingness	dependent-package-name
npm	3.5.1	3.5.2	3.5.1	node_modules/npm	dependent-package-name
local-dev	0.0.3	linked	linked	local-dev	dependent-package-name
once	1.3.2	1.3.3	1.3.3	node_modules/once	dependent-package-name

With these dependencies:

```
{
  "glob": "^5.0.15",
  "nothingness": "github:othiym23/nothingness#master",
  "npm": "^3.5.1",
  "once": "^1.3.1"
}
```

A few things to note:

- ? glob requires ^5, which prevents npm from installing glob@6, which is outside the semver range.
- ? Git dependencies will always be reinstalled, because of how they're specified. The installed committish might satisfy the dependency specifier (if it's something immutable, like a commit SHA), or it might

not, so npm outdated and npm update have to fetch Git repos to check.

This is why currently doing a reinstall of a Git dependency always forces a new clone and install.

? npm@3.5.2 is marked as "wanted", but "latest" is npm@3.5.1 because

npm uses dist-tags to manage its latest and next release channels.

npm update will install the newest version, but npm install npm

(with no semver range) will install whatever's tagged as latest.

? once is just plain out of date. Reinstalling node_modules from

scratch or running npm update will bring it up to spec.

Configuration

<!-- AUTOGENERATED CONFIG DESCRIPTIONS -->

See Also

? package spec

? npm update

? npm dist-tag

? npm registry

? npm folders

? npm workspaces

9.2.0

March 2026

NPM-OUTDATED(1)