

NAME**npm-uninstall** – Remove a package**Synopsis**

```
npm uninstall [<@scope>/]<pkg>[@<version>]... [-S|--save|--no-save]
```

aliases: remove, rm, r, un, unlink

Description

This uninstalls a package, completely removing everything npm installed on its behalf.

It also removes the package from the **dependencies**, **devDependencies**, **optionalDependencies**, and **peerDependencies** objects in your **package.json**.

Further, if you have an **npm-shrinkwrap.json** or **package-lock.json**, npm will update those files as well.

--no-save will tell npm not to remove the package from your **package.json**, **npm-shrinkwrap.json**, or **package-lock.json** files.

--save or **-S** will tell npm to remove the package from your **package.json**, **npm-shrinkwrap.json**, and **package-lock.json** files. This is the default, but you may need to use this if you have for instance **save=false** in your **npmrc** file

In global mode (ie, with **-g** or **--global** appended to the command), it uninstalls the current package context as a global package. **--no-save** is ignored in this case.

Scope is optional and follows the usual rules for npm help **scope**.

Examples

```
npm uninstall sax
```

sax will no longer be in your **package.json**, **npm-shrinkwrap.json**, or **package-lock.json** files.

```
npm uninstall lodash --no-save
```

lodash will not be removed from your **package.json**, **npm-shrinkwrap.json**, or **package-lock.json** files.

Configuration

```
<!-- AUTOGENERATED CONFIG DESCRIPTIONS START --> <!-- automatically generated, do not
edit manually --> <!-- see lib/utils/config/definitions.js -->
```

save

- Default: true
- Type: Boolean

Save installed packages to a package.json file as dependencies.

When used with the **npm rm** command, removes the dependency from package.json. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

workspace

- Default:
- Type: String (can be set multiple times)

Enable running a command in the context of the configured workspaces of the current project while filtering by running only the workspaces defined by this configuration option.

Valid values for the **workspace** config are either:

- Workspace names
- Path to a workspace directory

- Path to a parent workspace directory (will result in selecting all workspaces within that folder)

When set for the **npm init** command, this may be set to the folder of a workspace which does not yet exist, to create the folder and set it up as a brand new workspace within the project.

This value is not exported to the environment for child processes. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

workspaces

- Default: null
- Type: null or Boolean

Set to true to run the command in the context of **all** configured workspaces.

Explicitly setting this to false will cause commands like **install** to ignore workspaces altogether. When not set explicitly:

- Commands that operate on the **node_modules** tree (install, update, etc.) will link workspaces into the **node_modules** folder. – Commands that do other things (test, exec, publish, etc.) will operate on the root project, *unless* one or more workspaces are specified in the **workspace** config.

This value is not exported to the environment for child processes. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

include-workspace-root

- Default: false
- Type: Boolean

Include the workspace root when workspaces are enabled for a command.

When false, specifying individual workspaces via the **workspace** config, or all workspaces via the **workspaces** flag, will cause npm to operate only on the specified workspaces, and not on the root project. <!-- automatically generated, do not edit manually --> <!-- see lib/utils/config/definitions.js -->

<!-- AUTOGENERATED CONFIG DESCRIPTIONS END -->

See Also

- npm help prune
- npm help install
- npm help folders
- npm help config
- npm help npmrc