

# npm-prune

## Synopsis

```
npm prune [[<@scope>/]<pkg>...]
```

## Description

This command removes “extraneous” packages. If a package name is provided, then only packages matching one of the supplied names are removed.

Extraneous packages are those present in the `node_modules` folder that are not listed as any package’s dependency list.

If the `--production` flag is specified or the `NODE_ENV` environment variable is set to `production`, this command will remove the packages specified in your `devDependencies`. Setting `--no-production` will negate `NODE_ENV` being set to `production`.

If the `--dry-run` flag is used then no changes will actually be made.

If the `--json` flag is used, then the changes `npm prune` made (or would have made with `--dry-run`) are printed as a JSON object.

In normal operation, extraneous modules are pruned automatically, so you’ll only need this command with the `--production` flag. However, in the real world, operation is not always “normal”. When crashes or mistakes happen, this command can help clean up any resulting garbage.

## Configuration

### `omit`

- Default: ‘dev’ if the `NODE_ENV` environment variable is set to ‘production’, otherwise empty.
- Type: “dev”, “optional”, or “peer” (can be set multiple times)

Dependency types to omit from the installation tree on disk.

Note that these dependencies *are* still resolved and added to the `package-lock.json` or `npm-shrinkwrap.json` file. They are just not physically installed on disk.

If a package type appears in both the `--include` and `--omit` lists, then it will be included.

If the resulting omit list includes `'dev'`, then the `NODE_ENV` environment variable will be set to `'production'` for all lifecycle scripts.

#### **dry-run**

- Default: false
- Type: Boolean

Indicates that you don't want npm to make any changes and that it should only report what it would have done. This can be passed into any of the commands that modify your local installation, eg, `install`, `update`, `dedupe`, `uninstall`, as well as `pack` and `publish`.

Note: This is NOT honored by other network related commands, eg `dist-tags`, `owner`, etc.

#### **json**

- Default: false
- Type: Boolean

Whether or not to output JSON data, rather than the normal output.

- In `npm pkg set` it enables parsing set values with `JSON.parse()` before saving them to your `package.json`.

Not supported by all npm commands.

#### **foreground-scripts**

- Default: false
- Type: Boolean

Run all build scripts (ie, `preinstall`, `install`, and `postinstall`) scripts for installed packages in the foreground process, sharing standard input, output, and error with the main npm process.

Note that this will generally make installs run slower, and be much noisier, but can be useful for debugging.

#### **ignore-scripts**

- Default: false
- Type: Boolean

If true, npm does not run scripts specified in `package.json` files.

Note that commands explicitly intended to run a particular script, such as `npm start`, `npm stop`, `npm restart`, `npm test`, and `npm run-script` will still run

their intended script if **ignore-scripts** is set, but they will *not* run any pre- or post-scripts.

#### **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.

#### **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.

#### **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.

### See Also

- [npm uninstall](#)
- [npm folders](#)
- [npm ls](#)