Pro Teams Pricing Documentation

npm

Sign Up Sign In

Search packages Search

## npm-upgrade

3.1.0 • Public • Published a year ago

Readme

Code Beta

16 Dependencies

7 Dependents

30 Versions

# npm-upgrade

Interactive CLI utility to easily update outdated NPM dependencies with changelogs inspection support.

npm v3.1.0 downloads 443k

## What is this for?

If you are tired of manually upgrading `package.json` every time your package dependencies are getting out of date then this utility is for you.

Take a look at this demo:

```
/Volumes/Work/npm-upgrade master $
```

## Installation

First, install **Node.js** (at least `v10.19`).

Then install this utility as global npm-module:

```
npm i -g npm-upgrade
```

## Usage

This utility is supposed to be run in the root directory of your Node.js project (that contains `package.json`). Run `npm-upgrade --help` to see all available top-level commands:

```
check [filter]          Check for outdated modules
ignore <command>        Manage ignored modules
changelog <moduleName>  Show changelog for a module
```

Run `npm-upgrade <command> --help` to see usage help for corresponding command. `check` is the default command and can be omitted so running `npm-upgrade [filter]` is the same as `npm-upgrade check [filter]`.

## `check` command

It will find all your outdated deps and will ask to updated their versions in `package.json`, one by one. For example, here is what you will see if you use outdated version of `@angular/common` module:

```
Update "@angular/common" in package.json from 2.4.8 to 2.4.10? (Use arro
❯ Yes
  No
  Show changelog
  Ignore
  Finish update process
```

- `Yes` will update `@angular/common` version in `package.json` to `2.4.10`, but not immediately (see explanation below)
- `No` will not update this module version.
- `Show changelog` will try to find changelog url for the current module and open it in default browser.
- `Ignore` will add this module to the ignored list (see details in `Ignoring module` section below).
- `Finish update process` will ...hm... finish update process and save all the changes to `package.json`.

A note on saving changes to `package.json`: when you choose `Yes` to update some module's version, `package.json` won't be immediately updated. It will be updated only after you will process all the outdated modules and confirm update **or** when you choose `Finish update process`. So if in the middle of the update process you've changed your mind just press `Ctrl+C` and `package.json` will remain untouched.

If you want to check only some deps, you can use `filter` argument:

```
# Will check all the deps with `babel` in the name:
npm-upgrade '*babel*'

# Note quotes around `filter`. They are necessary because without them

# Will check all the deps, excluding any with `babel` in the name:
npm-upgrade '!*babel*'

# You can combine including and excluding rules:
npm-upgrade '*babel* !babel-transform-* !babel-preset-*'
```

If you want to check only a group of deps use these options:

```
-p, --production   Check only "dependencies"
-d, --development  Check only "devDependencies"
-o, --optional     Check only "optionalDependencies"
```

Alternatively, you can use the `-g` ( `--global` ) flag to upgrade your global packages. **Note** that this flag is mutually exclusive and `npm-upgrade` will only recognise the global flag if supplied with others. Also **Note** that this option will automatically attempt to upgrade your global packages using `npm install -g <package>@<new-version>` .

## Ignoring module

Sometimes you just want to ignore newer versions of some dependency for some reason. For example, you use `jquery v2` because of the old IE support and don't want `npm-upgrade` to suggest you updating it to `v3` . Or you use `some-funky-module@6.6.5` and know that the new version `6.6.6` contains a bug that breaks your app.

You can handle these situations by ignoring such modules. You can do it in two ways: choosing `Ignore` during update process or using `npm ignore add` command.

You will asked two questions. First is a version range to ignore. It should be a valid **semver** version. Here are a few examples:

- `6.6.6` - will ignore only version `6.6.6`. When the next version after `6.6.6` will be published `npm-upgrade` will suggest to update it. Can be used in `some-funky-module` example above.
- `>2` - will ignore all versions starting from `3.0.0`. Can be used in `jquery v2` example above.
- `6.6.x || 6.7.x` - will ignore all `6.6.x` and `6.7.x` versions.
- `*` - will ignore all new versions.

And after that `npm-upgrade` will ask about the ignore reason. The answer is optional but is strongly recommended because it will help to explain your motivation to your colleagues and to yourself after a few months.

All the data about ignored modules will be stored in `.npm-upgrade.json` file next to your project's `package.json`.

## `ignore` command

It will help you manage ignored modules. See **Ignoring module** section for more details. It has the following subcommands:

```
npm-upgrade ignore <command>

Commands:
  add [module]          Add module to ignored list
  list                  Show the list of ignored modules
  reset [modules...]    Reset ignored modules
```

- `add` - will add a module from your deps to ignored list. You can either provide module name as optional `module` argument or interactively select it from the list of project's deps.
- `list` - will show the list of currently ignored modules along with their ignored versions and reasons.
- `reset` - will remove modules from the ignored list. You can either provide module names as `modules` argument (separated by space) or interactively select them from the list of

project's deps.

### `changelog` command

```
npm-upgrade changelog <moduleName>
```

Will try to find changelog url for provided module and open it in default browser.

## Troubleshooting

**Wrong changelog shown for *<moduleName>* or not shown at all!**

Yes, It can happen sometimes. This is because there is no standardized way to specify changelog location for the module, so it tries to guess it, using these rules one by one:

1. Check `db/changelogUrls.json` from `master` branch on GitHub or the local copy if it's unreachable.
2. Check `changelog` field from module's `package.json` .
3. Parse module's `repository.url` field and if it's on GitHub, try to request some common changelog files ( `CHANGELOG.md` , `History.md` etc.) from `master` branch and if it fails, open `Releases` page.

So, if it guessed wrong it would be great if you could either **fill an issue** about this or submit a PR which adds proper changelog URL to `db/changelogUrls.json` . There is a tool in the repository for you to make it as easy as possible:

```
./tools/addModuleChangelogUrlToDb.js <moduleName> <changelogUrl>
```

## License

MIT

### Keywords

npm update outdated dependencies cli interactive automatic changelog ignore

**Install**

```
> npm i npm-upgrade
```

**Repository**

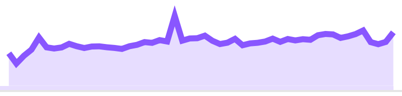
github.com/th0r/npm-upgrade

**Homepage**

github.com/th0r/npm-upgrade

**Weekly Downloads**

7,894



| | |
|---|---|
| **Version** | **License** |
| 3.1.0 | MIT |
| **Unpacked Size** | **Total Files** |
| 80.6 kB | 39 |
| **Issues** | **Pull Requests** |
| 16 | 8 |

**Last publish**

a year ago

**Collaborators**

>_Try on RunKit

Report malware

## Support

Help

Advisories

Status

Contact npm

## Company

About

Blog

Press

## Terms & Policies