About semantic versioning

Table of contents

To keep the JavaScript ecosystem healthy, reliable, and secure, every time you make significant updates to an npm package you own, we recommend publishing a new version of the package with an updated version number in the `package.json` file that follows the semantic versioning spec. Following the semantic versioning spec helps other developers who depend on your code understand the extent of changes in a given version, and adjust their own code if necessary.

Note: If you introduce a change that breaks a package dependency, we strongly recommend incrementing the version major number; see below for details.

Incrementing semantic versions in published packages

To help developers who rely on your code, we recommend starting your package version at `1.0.0` and incrementing as follows:

Code status	Stage	Rule	Example version
First release	New product	Start with 1.0.0	1.0.0
Backward compatible bug fixes	Patch release	Increment the third digit	1.0.1
Backward compatible new features	Minor release	Increment the middle digit and reset last digit to zero	1.1.0
Changes that break backward compatibility	Major release	Increment the first digit and reset middle and last digits to zero	2.0.0

Using semantic versioning to specify update types your package can accept

You can specify which update types your package can accept from dependencies in your package's `package.json` file.

For example, to specify acceptable version ranges up to 1.0.4, use the following syntax:

- Patch releases: `1.0` or `1.0.x` or `~1.0.4`
- Minor releases: `1` or `1.x` or `^1.0.4`
- Major releases: `*` or `x`

For more information on semantic versioning syntax, see the npm semver calculator.

Example

```
"dependencies": {
  "my_dep": "^1.0.0",
  "another_dep": "~2.2.0"
},
```

Resources