How to publish packages to npm (the way the industry does things)

It's simple to publish a package onto npm. There are two steps:

1. Create your package.
2. Publish the package.

But publishing packages the way the industry does it? Not so simple. There are more steps. We'll go through what steps are required, and I'll show you an easy way to publish and update your package.

Creating your first package

This section is for you if you haven't published a package to npm before. Feel free to skip to the next section if you've published one before.

To publish your first package to npm, you need to go through these steps:

First, you need to have an npm account . Create one here if you don't have one yet.

Second, you need to login to your npm account through the command line. (You need to have Node and npm installed on your system before you perform this step. Install them here).

To sign in, you use `npm login` .

```
npm login
```

You'll be prompted to enter your username, password, and email address.

```
[~] npm login
Username: zellwk
Password:
Email: (this IS public) zellwk@gmail.com
Logged in as zellwk on https://registry.npmjs.org/.
[~]
```

Third, you need to create a package . To do so, create a folder somewhere on your computer and navigate to it. The command line version is:

```
# Creating a folder named how-to-publish-to-npm
mkdir how-to-publish-to-npm

# Navigating into the folder
cd how-to-publish-to-npm
```

Next, you want to begin the project with the `npm init` command.

```
npm init
```

This command runs you through a few questions and creates a `package.json` file for you at the end. This `package.json` file contains the bare necessities you need to publish your project. (Feel free to skip questions that don't make sense).

```
package name: (how-to-publish-to-npm)
version: (1.0.0)
description: An example repository to teach people how to publish to npm
entry point: (index.js)
test command:
git repository:
keywords:
license: (MIT)
```

The final step is to publish your package with the `npm publish` command.

```
npm publish
```

If the package already exists on npm (because your package has the same name as another package on npm), you won't be able to publish it. You'll get an error.

```
[how-to-publish-to-npm] npm publish
npm notice
npm notice 📦  how-to-publish-to-npm@1.0.0
npm notice === Tarball Contents ===
npm notice 325B package.json
npm notice === Tarball Details ===
npm notice name:          how-to-publish-to-npm
npm notice version:       1.0.0
npm notice package size:  319 B
npm notice unpacked size: 325 B
npm notice shasum:        760cfc1f848ef476eddb78807f7e3d327e3fbd15
npm notice integrity:     sha512-IF3RjKivzgTUG[...]vnv/jonXTlQtw==
npm notice total files:   1
npm notice
npm ERR! publish Failed PUT 403
npm ERR! code E403
npm ERR! You do not have permission to publish "how-to-publish-to-npm". Are you
logged in as the correct user? : how-to-publish-to-npm
[how-to-publish-to-npm]
```

You'll need to change your package name.

To change your package name, you change the `name` property in the `package.json` file. Here, I changed it to `publishing-to-npm` .

(You can check for naming collisions by doing a search on npm, or through the `npm search` command).

```
{
  "name": "publishing-to-npm",
  "version": "1.0.0",
  "description": "An example repository to teach
  people how to publish to npm",
```

It's also a good idea to update the folder name as well for consistency. Here's the command line equivalent.

```
# Command to change folder names by moving everything
mv how-to-publish-to-npm publishing-to-npm
```

Try the `publish` command again. You should get a success message now.

```
[publishng-to-npm] npm publish
npm notice
npm notice 📦  publishing-to-npm@1.0.0
npm notice === Tarball Contents ===
npm notice 321B package.json
npm notice === Tarball Details ===
npm notice name:          publishing-to-npm
npm notice version:       1.0.0
npm notice package size:  318 B
npm notice unpacked size: 321 B
npm notice shasum:        8fabb6d5b58a7211728d530549173ca20dc88d5b
npm notice integrity:     sha512-sn+O03KqAKaTH[...]OKxstKPE1+OBw==
npm notice total files:   1
npm notice
+ publishing-to-npm@1.0.0
[publishng-to-npm]
```


What to do if every name you came up with is taken up already

This is a common problem since many people create packages on npm. It's difficult to get the package name you desire sometimes. (It's like how I can *never* find a good `.com` domain).

To combat against this problem, npm lets you publish to a scope. This means you can publish the package under your own username (or npm organization), so you're free from naming problems.

To publish to a scope, you can either:

1. Change the `name` to `@username/package-name` manually in `package.json`
2. Run `npm init --scope=username` instead of `npm init`

If your repository has a scope, you need to adjust the publish command slightly:

```
npm publish --access public
```

That's all you need to do to publish a package to npm.

Now, let's move on to how the industry does publishes packages.

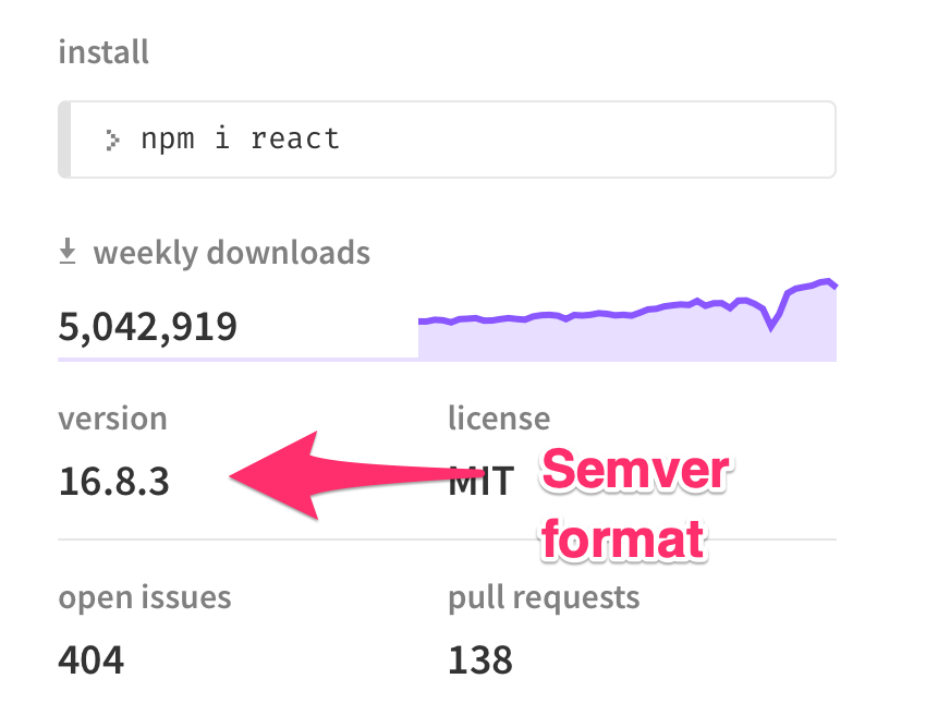
The way the industry publishes (and updates) packages.

Consider a popular framework like React. If you dig around React, you'll notice a few things:

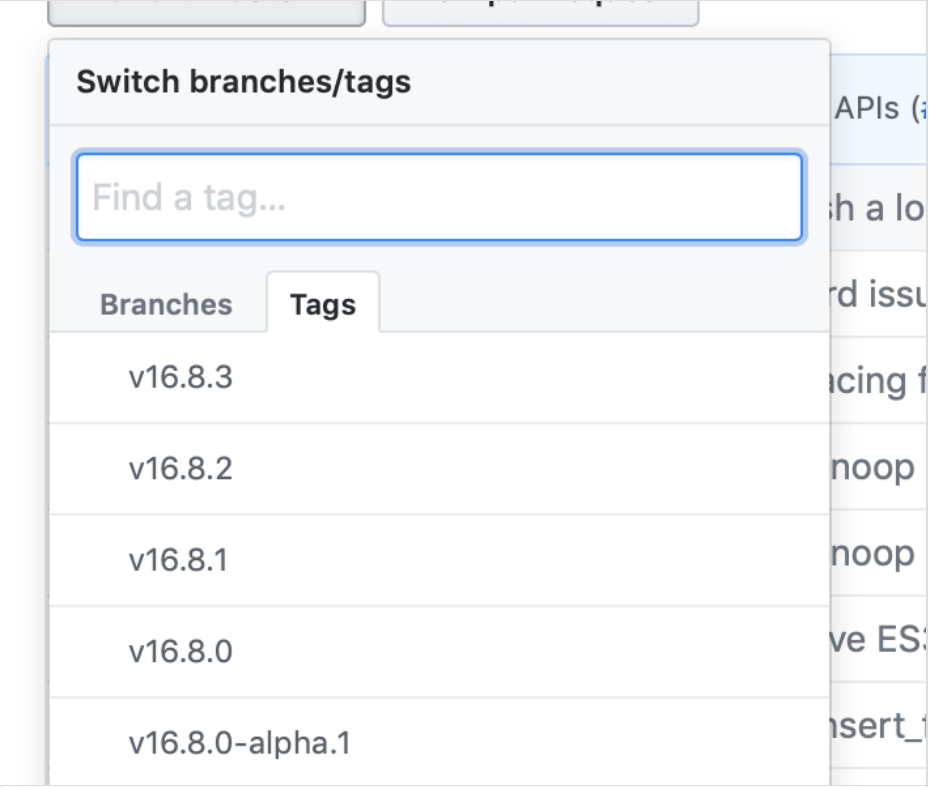
First, React has a Github repository.

Second, React is published on npm.

Third, React follows Semantic versioning (Semver for short).



Fourth, each update to React has a git tag associated with it. This git tag follows Semver as well.



Fifth, there are release notes for every React update.

This means publishing a package involves many steps. At the very least, you need to:

1. Run tests (if there are any)
2. Update `version` in `package.json` according to Semver
3. Create a git tag according to Semver
4. Push the package to Github
5. Push the package to npm
6. Create release notes for every update

It's common to forget one of these things when we're ready to push. Sometimes we `npm publish` and we enjoy a break. When we're back, we screw ourselves for forgetting.

There's an easier way. It's with a tool called `np`.

np

np (created by Sindre Sorhus) makes it easier for us to publish packages without missing any of the steps I detailed above.

Installing np

To install `np`, you can run the following command:

```
npm install np
```

This works. But I prefer installing `np` globally on my computer so I can

run the `np` command anywhere.

```
sudo npm install --global np
```

Before using np

Before you use `np` you need to make sure:

1. Your project is a Git repository
2. It needs to have a remote
3. You must have pushed to the remote at least once.
4. You also need to make sure your working directory is clean.

```
# Initialize Git
git init

# Adds a remote repository
git remote add origin some-url

# Commit changes
git add .
git commit -m "Initial Commit"
```

If your project is not a Git repository, you'll get this error:

```
[publishng-to-npm] np

Publish a new version of publishing-to-npm (current: 1.0.0)


✖ Command failed: git rev-list --max-parents=0 HEAD
fatal: not a git repository (or any of the parent directories): .git
```

np's error if project is not a Git repository

If your project doesn't have remote, you'll get this error (at a later part of the checks).

```
❯ Prerequisite check
  ✔ Ping npm registry
  ✔ Verify user is authenticated
  ✖ Check git remote
    → and the repository exists.
```

np's error if project does not have a remote. This error happens later in the process

If your working directory is dirty, you'll get this error:

```
[publishng-to-npm] np                                          master  *

Publish a new version of publishing-to-npm (current: 1.0.0)


✖ Command failed: git rev-list --max-parents=0 HEAD
fatal: ambiguous argument 'HEAD': unknown revision or path not in the working tr
ee.
Use '--' to separate paths from revisions, like this:
'git <command> [<revision>...] -- [<file>...]'
```

np's error working directory is dirty

If you haven't pushed to the Git remote at least once, `np` will just hang and do nothing.

Using npm

To use `np`, you run the `np` command.

```
np
```

`np` will prompt you to enter a Semver number.

```
? Select semver increment or specify new version (Use arrow keys)
❯ patch         1.0.1
  minor         1.1.0
  major         2.0.0
  prepatch      1.0.1-0
  preminor      1.1.0-0
  premajor      2.0.0-0
  prerelease    1.0.1-0
  ──────────────
  Other (specify)
```

Choose a number and `np` will ask you to confirm your choice.

```
? Select semver increment or specify new version patch  1.0.1
? Will bump from 1.0.0 to 1.0.1. Continue? (Y/n)
```

`np` then does the rest of the publishing stuff for you.

Error with running tests

`np` runs the `npm test` command as part of its checks.

If you followed the tutorial up to this point, you would get an error that looks like this:

```
? Will bump from 1.0.0 to 1.0.1. Continue? Yes
  ✔ Prerequisite check
  ✔ Git
  ✔ Cleanup
  ✔ Installing dependencies using npm
  ✖ Running tests using npm
    → Error: no test specified
```

This happens because our `npm test` command results in an error. You can try it yourself:

```
npm test
```

```
[publishing-to-npm] npm test                                    master

> publishing-to-npm@1.0.0 test /Users/zellwk/projects/publishing-to-npm
> echo "Error: no test specified" && exit 1

Error: no test specified
npm ERR! Test failed.  See above for more details.
```


To fix this error, we need to change the `test` script in `package.json` file.

Right now it looks like this:

```
"scripts": {
  "test": "echo \"Error: no test specified\" && exit 1"
},
```

Change it to this:

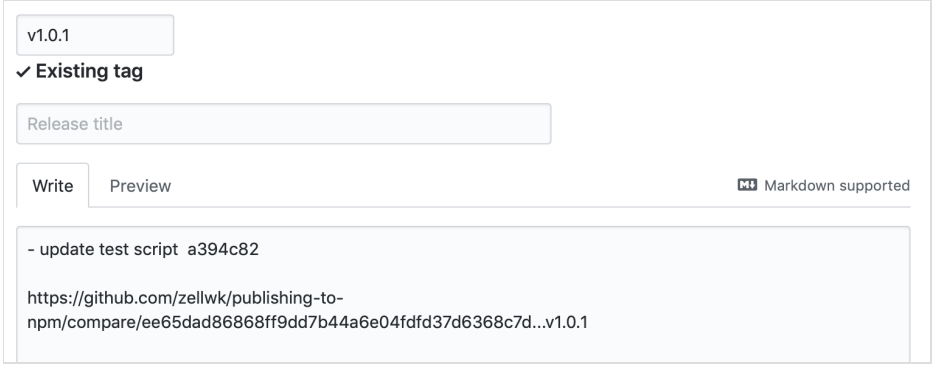
```
"scripts": {
  "test": "echo \"No test specified\""
},
```

This change works because `exit 1` creates an error.

With this change, `np` should complete the publishing process. (Remember to commit the change before running `np`).

```
? Select semver increment or specify new version patch  1.0.1
? Will bump from 1.0.0 to 1.0.1. Continue? Yes
  ✔ Prerequisite check
  ✔ Git
  ✔ Cleanup
  ✔ Installing dependencies using npm
  ✔ Running tests using npm
  ✔ Bumping version using npm
  ✔ Publishing package using npm
  ✔ Pushing tags
  ✔ Creating release draft on GitHub
```

At the end of the process, `np` launches a browser window for you to write your release notes.



In short, `np` makes publishing packages much simpler!