



# What I learned from analysing 1.65M versions of Node.js modules in NPM



Karl Düüna · Follow

Published in Security and Node.js · 18 min read · Jun 21, 2016



379



10



Sign up to discover human stories that deepen your understanding of the world.

## Free

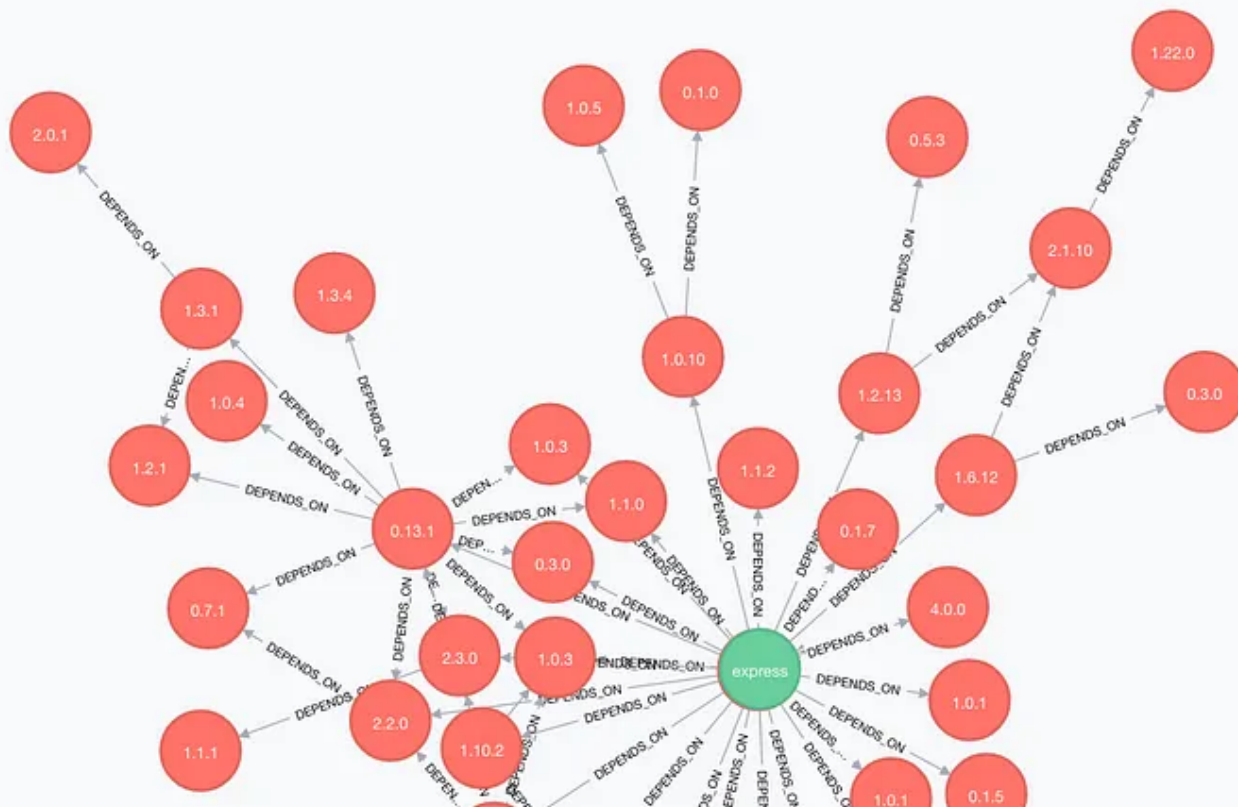
- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

## Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

## Prologue: How it all began

Let's start at the beginning. It all started by me trying to solve the age old question of how to properly include modules for deploy processes. We all know that including the whole *node\_modules* folder is not a good solution as it breaks compiled libraries. At the same time having a *package.json* + *npm-shrinkwrap.json* and using NPM install is ALSO not as reliable as one would hope. NPM does go down from time to time, network fails, packages change etc.



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

information from there, such as:

1. Some useful commands like [npm version](#) – which allows you to update your `package.json/npm-shrinkwrap.json`, create a git commit and tag it with the new version, all with a single command. Or [npm pack](#) – which allows you to download the requested package as a `.tgz` to the root directory of current packages.
2. Various ways to [configure NPM](#) – cache location and timeout, `registry url`, `ignore-scripts`, `save-exact`, `sign-git-tag` etc. Or that you can actually set these properties per project using a `.npmrc` file.
3. How NPM resolves package installation folders in [v2 vs v3](#), and how the [v3 dependency resolver tries to avoid duplicates](#) and how [it is non-deterministic](#).



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

keeps all the packages and metadata it ever downloads in its cache folder indefinitely? Well it does. However, if the data is older than 10s, then it will check with the repository if there have been changes before it uses the cache.

From there I got an idea that maybe it would be possible to use the cache mechanism to store files in the local directory and install from there. After giving it a try, it was actually quite easy to set up — created a `.npmrc` file into my project test folder it was good to go.



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

So what happens is that when you install things, NPM will store the tarballs and metadata into the *packages* folder. When I delete the *node\_modules* folder and run *npm install* again then it won't talk to the repository. So far so good. But then I ran into a wall — what happens if I want to update something? Running *npm update* will surely ignore the cache and ask repository, right? Right?

Well, actually no. It seems that if you set the cache, then every GET request will use the cache instead. This means you have no way to update your packages unless you manually overwrite the *cache-min* setting. How does this make sense? We will find out in the next section — How NPM handles install.



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

- when installing a module, all the metadata for the module is downloaded (including readmes and package.json-s of ALL versions of the said module)
- shrinkwrapped install is much more efficient

In the previous section we reached a dead end because NPM unexpectedly uses cache too much — even for the *update* command, which doesn't seem to make sense. Looks like we don't have enough information! Time to dig deeper and explore how does NPM actually install things.



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month





Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

4. Figure out best match (latest possible version for a given semver range) and download the tarball for each dependency
  5. Take the first dependency alphabetically
  6. Once the module has been downloaded and unpacked, check its dependencies:
    - a) If there are dependencies → back to point 1. , but now in the context of the module
    - b) If no dependencies → continue
  7. Take the next dependency alphabetically:
    - a) Got dependency → move back to step 6
    - b) No more dependencies → continue
  8. Check for install scripts and if any exist, run in order
- 



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

Now the question remains — why does it download the whole metadata?  
With no professional ties to NPM, I can only speculate, but my (semi-educated) guess would be that:

- They want all versions and dependencies to resolve the range optimally and not create subsequent requests afterwards when it encounters a different range
- Match node versions
- Shasum for package content comparison
- It is simply easier to download everything and sort it out on location

Which all probably boils down to a fact that they have not yet gotten



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

dependency, then the second version will be downloaded after the previous has been resolved.

The following example shows how *bytes@2.2.0* will be downloaded only after all the other packages are downloaded and unpacked.



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

All this gave enough information to actually solve the first of the initial quests — how to include modules with your code for deployment. Queue *npm-ducttape*.

### Part3 — The birth of npm-ducttape

*EDIT: I was notified of a more mature module solving the same problem called shrinkpack, be sure to check it out. However if you need a lightweight solution (25 dependencies vs 0) then ducttape might be your tool.*

tl;dr

- You can use the npm-ducttape to include all your dependencies into your repository for deploy



Sign up to discover human stories that deepen your understanding of the world.

#### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

#### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

However, as it turns out that shrinkwrap actually includes urls to the required resources and doesn't otherwise talk to the registry, then we could try to use it. A small test later it was confirmed that shrinkwrap like *package.json* accepts *file:* urls. Armed with this knowledge all we need to do is:

1. run *shrinkwrap*
2. download all the files listed in *npm-shrinkwrap* file
3. move all the package files under a specified folder
4. rewrite *npm-shrinkwrap* file so that urls to point to local files instead
5. ???



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



## Part4 — The part where I analyse all modules and versions

tl;dr

- there were 1.65M versions of NPM modules at the time
- used Google Cloud and various scripts to analyse almost all of them

So what does it take to analyse NPM and all its modules and versions? To begin with, you need data about what is in the repository. This is actually surprisingly easy, because NPM provides you with a simple way to set up your own copy of it. All it takes is to:



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

2. analyse metadata
3. download the code for all versions of the module
4. unpack and analyse the code

Easy right? *Right?*

Not so fast! Turns out that downloading and analysing a version takes on average about 1s. That's not a lot, but when there are 1.65M versions, then it means ~20 days of non stop download and analysis. I didn't have that much time.

Luckily Google helped me out – I happened to have 300\$ trial money



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

‘hglv2<! — \” onmouseover=alert(1)”’, packages with invalid tar headers etc.

This in turn created a mind boggling amount of edge cases, which had to be handled. Especially since I wanted to analyse whole dependency trees — which of course meant that the dependency requirements had to be solved. Oh no, not the dependencies!



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

PS. For the following statistics, I analysed only the latest version of each module.

First off there is actually a surprising amount of placeholder modules in NPM — almost 2% of all modules have no content besides a *package.json* and maybe a licence. While that doesn't necessarily mean that they are placeholders then mostly they are. Some, however, have scripts and the *package.json* is the whole content of the package.

Speaking of licences — MIT is by far the most popular one with over 50% of modules. Then of course we have a large chunk of missing licence information and the last quarter is divided between myriad of less popular licenses. For me the most interesting one was WTFPL (do What The Fuck you want to Public Licence), which is surprisingly popular, with nearly a



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

get old and people stop paying attention to them. Looking at the graph below, we see that about 40% of modules were not updated last year.



Sign up to discover human stories that deepen your understanding of the world.

**Free**

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

**✦ Membership**

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

The mean number of packages installed for a module is 35.3 , with the maximum being a whopping 1615 for [npm-collection-explicit-installs](#) — a module that collects popular NPM modules under one package.

While the outliers do skew the data and over 50% of modules install 4 or fewer packages, over 10% of the modules pack 100+ packages, which I find rather disturbing. Especially when put into context of my average project (not just a package, but an actual service) where I usually have a heck of a lot more dependencies than 2–3 in my *package.json* and that means the actual amount of dependencies is very high.

Another equally interesting dependency related nuance is that the mean amount of original code vs dependency code is about 45%. (Note: I



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

Original code % based on size



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



6. `minimist` — for parsing command line arguments
7. `lodash` — utility function collection
8. `minimatch` — A minimal matching utility.
9. `commander` — For parsing command line arguments
10. `assert-plus` — A wrapper around core `assert` to provide convenience functions

What to make out of this? Well apparently we are not confident in the language itself. There are a lot of people who need to depend on a separate module to check if something is an array. That function has been in the language since Chrome 5 (2010) FFS.



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

your dependencies. And that means it will be enough to compromise any one of the developers of the said modules. It's a terrifying world out there that is made even more terrifying by the possibility of lifetime scripts in *package.json*.

For those who do not know — you can specify commands that get run during install or uninstall of a module. These scripts can include anything and will be run with the user's privileges.

Looking at packages in the repository, about 2% contain such scripts and they vary greatly in purpose and implementation. One of the funnier ones I found was on a package called [closing-time](#). What it does is download and execute a shell script, which in turn downloads Semisonic's — Closing



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

And while my quick analysis didn't find any purposely malicious scripts, I did find horrors that for example on install run the NPM install again against a different repository.

If that's not enough, there was recently an interesting [NPM Worm](#) concept attack, which could use the lifetime scripts to propagate through the NPM packages.

Whether you like it or not, scripts are dangerous and thus I dearly recommend running install with the *ignore-scripts* flag or setting it true as a default.

```
npm config set ignore-scripts true
```



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

But not everything in NPM is dark and out to get you. There were some weirdly modules that stood out and gave a good laugh during the analysis.

For example a module called 0126af95c0e2d9b0a7c78738c4c00a860b04acc8 — which ironically exports a function to produce a random string.



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month



There are serious issues with trust and security that inhibit the average Node.js development's push to production spotlight. But for now, we as a community, like to close our eyes and hope for the best.

Only time will tell if this tactic will pay off, but NPM is here to stay (for now).

*There were a lot of interesting security issues and topics I researched during this whole process. While I would have liked to include those in this post, it had already become a wall of text. So I will most probably write another wall of text to cover those sometime in the future.*



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

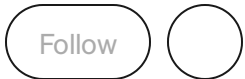
Try for \$5/month



## Written by Karl Düüna

532 Followers · Editor for Security and Node.js

Entrepreneur & Hacker by heart. CTO of <http://www.nodeswat.com> — researching and developing scalable & secure #nodejs apps



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

○ Henrik Peinar in Security and Node.js

## Simple node.JS and Slack WebHook integration

This post will walk you through the process of how to turn this awesome chat tool into a...

4 min read · Aug 6, 2019


 338  3

○ Karl Düüna in Security and Node.js

## Concurrency, MySQL and Node.js: A journey of discovery

Our story begins like so many others with a code loving protagonist — someone we all c...

10 min read · Feb 6, 2018

  1.3K  2 



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

# JSON is incredibly slow: Here's What's Faster!

Unlocking the Need for Speed: Optimizing JSON Performance for Lightning-Fast Apps...

16 min read · Sep 28



7.4K



94



16



# How to Update Node.js to Any Version: A Step-by-Step Guide

Introduction

2 min read · Aug 3

## Lists

### Stories to Help You Grow as a Software Developer

19 stories · 521 saves

### It's never too late or early to start something

15 stories · 195 saves

### General Coding Knowledge

20 stories · 545 saves

### Modern Marketing

42 stories · 233 saves



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month

○ Umar Farooque Khan

## Difference between package.json and package-lock.json

The package.json and package-lock.json files are both important components of a Node.js...

2 min read · Jun 20

👏 31    💬 1



○ fatfish in JavaScript in Plain English

## Interview: Can You Stop “forEach” in JavaScript?

there are 3 ways to stop forEach in JavaScript

🌟 · 5 min read · Aug 3

👏 2.8K    💬 64



Sign up to discover human stories that deepen your understanding of the world.

### Free

- ✓ Distraction-free reading. No ads.
- ✓ Organize your knowledge with lists and highlights.
- ✓ Tell your story. Find your audience.

Sign up for free

### ✦ Membership

- ✓ Access the best member-only stories.
- ✓ Support independent authors.
- ✓ Listen to audio narrations.
- ✓ Read offline.
- ✓ Join the Partner Program and earn for your writing.

Try for \$5/month