Hello and welcome, everyone!

How NPM or Yarn are working; Going to create our own package manager

Seyyed Ali Mohammadiyeh (Max Base)
Faculty of Mathematics, University of Kashan
Open-Source Maintainer, GitHub
CTO, Asrez
maxbasecode@gmail.com

WHO I AM

ALI, (Aka Max Base)

- GitHub and Open-Source world **(^_^)**
- Open-Source Maintainer
- Software Engineer, Programming more then 10 years
- Mathematics background
- **@basemax** on GitHub

Introduction

As a developer, it's important to understand how package managers work because they play a vital role in managing dependencies and streamlining the development process.

In this presentation, we will take a deep dive into how NPM and Yarn, the most popular package managers in the JavaScript ecosystem, work and understand the process of creating our own package manager.

We will cover the basics of what a package manager is, how it's used, and the command-line interface.

By the end of this presentation, you will have a solid understanding of how package managers work and the knowledge to create your own package manager.

Let's begin!

What is a package manager?

A **package manager** is a software tool that helps developers manage and share libraries, modules, and other software components.

Two of the most popular package managers in the JavaScript ecosystem are **NPM** (Node Package Manager) and **Yarn** (Yet Another Resource Negotiator).

A package manager does the following:

- Manages dependencies, ensuring that all the required packages are installed and up to date.
- Automates the process of installing, updating, and removing packages.
- Provides a way for developers to share their own packages with the community.

How NPM and Yarn work

NPM and Yarn both work by keeping track of the packages that a project depends on in a file called package.json.

The package.json file lists the packages and their versions that the project depends on.

When a package is installed, it is placed in a directory called **node_modules**, which is located in the project's root directory.

NPM and Yarn have slight differences in their command-line interface and package installation process, but the overall concept is the same.

Installing packages

Installing packages using NPM or Yarn is a simple process.

To install a package using NPM, you would use the command **npm install <package-name>**.
To install a package using Yarn, you would use the command **yarn add <package-name>**.

Packages can be installed globally or locally. A global package can be used in any project on the same computer, while a local package is only available to the current project.

To install a package globally using NPM, you would use the command **npm install -g <package-name>**.
To install a package globally using Yarn, you would use the command **yarn global add <package-name>**.

Marketplace

https://www.npmjs.com/

https://yarnpkg.com/

Private packages

NPM and Yarn both provide a feature to create private packages.

Private packages can only be accessed and installed by the users who have been granted access to them.

To create a private package on NPM, you can use a paid plan, and for Yarn you can use a self-hosted solution called Yarn workspaces.

Private packages can be useful for organizations that want to keep their codebase private, or for developers who want to share a package with a specific group of people.

Security and vulnerabilities

Package managers rely on packages that are created by the community, and sometimes packages can have security vulnerabilities.

NPM and Yarn both have built-in security features, such as the ability to audit packages for known vulnerabilities.

To audit packages using NPM, you can use the command **npm audit**.

To audit packages using Yarn, you can use the command **yarn audit**.

It's important to regularly check your project's packages for vulnerabilities and update them when necessary.

Node API

https://registry.npmjs.org/

For checking a package with the name “node-fetch”:

https://registry.npmjs.org/node-fetch/

About “node-fetch” package


```
▼ object {17}
    _id : node-fetch
    _rev : 341-06f9a00f5ec9233537e3ba2cd3178c67
    name : node-fetch
    description : A light-weight module that brings Fetch API to node.js
  ▼ dist-tags {5}
      latest : 3.3.0
      next : 3.0.0-beta.10
      cjs : 2.6.7
      beta : 4.0.0-beta.4
      release-2.x : 2.6.8
  ▼ versions {88}
    ▼ 0.1.0 {22}
        name : node-fetch
        version : 0.1.0
        description : A light-weight module that brings window.fetch to node.js
        main : index.js
      ▼ scripts {1}
          test : mocha test/test.js
      ▼ repository {2}
          type : git
          url : https://github.com/bitinn/node-fetch.git
      ▼ keywords [3]
```


Asrez

About “node-fetch-xxxxxx” package

https://registry.npmjs.org/node-fetch-xxxxxx

404

{"error":"Not found"}

Creating and publishing packages

Creating and publishing packages is a powerful feature of NPM and Yarn.

To create a package, you need to create a **package.json** file that contains information about the package.

To publish a package, you need to create an account on NPM or Yarn and then use the command **npm publish** or **yarn publish**.

Once a package is published, other developers can install it using the package name.

You can also unpublish packages using the command **npm unpublish <package-name>**.

Creating your own package manager

Creating your own package manager can be a challenging task, but it can also be a rewarding learning experience.

There are many open-source package managers that can be used as a starting point, such as Lerna. https://github.com/lerna/lerna

It's important to understand the various components of a package manager, such as the command-line interface, package installation process, and dependency management.

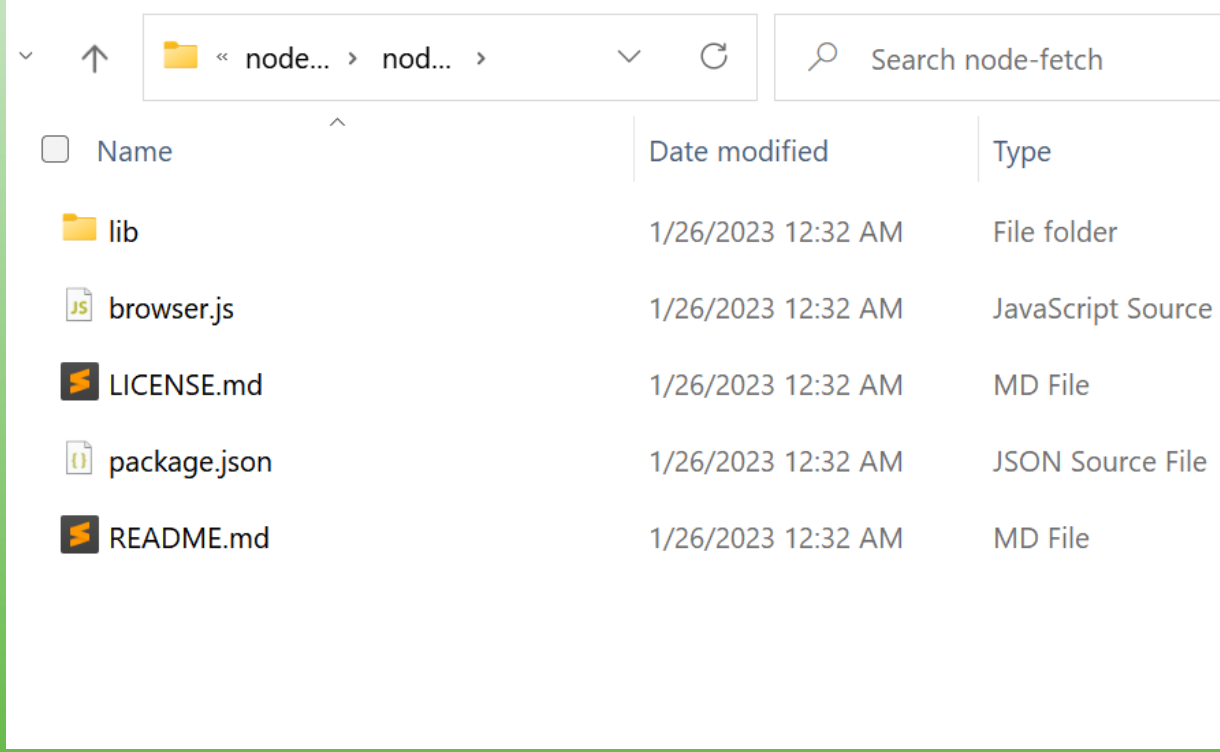
Commands

```
$ node cli.js

$ node cli.js help

$ node cli.js install

$ node cli.js install name1 name2 name3

$ node cli.js remove name1 name2 name3
```

node_modules directory

Name	Date modified	Type
.bin	1/26/2023 12:23 AM	File folder
available-typed-arrays	1/26/2023 12:23 AM	File folder
call-bind	1/26/2023 12:23 AM	File folder
chownr	1/26/2023 12:18 AM	File folder
for-each	1/26/2023 12:23 AM	File folder
fs	1/26/2023 12:23 AM	File folder
fs-minipass	1/26/2023 12:18 AM	File folder
function-bind	1/26/2023 12:23 AM	File folder
get-intrinsic	1/26/2023 12:23 AM	File folder
gopd	1/26/2023 12:23 AM	File folder
has	1/26/2023 12:23 AM	File folder
has-symbols	1/26/2023 12:23 AM	File folder
has-tostringtag	1/26/2023 12:23 AM	File folder
inherits	1/26/2023 12:23 AM	File folder
is-arguments	1/26/2023 12:23 AM	File folder
is-callable	1/26/2023 12:23 AM	File folder
is-generator-function	1/26/2023 12:23 AM	File folder
is-typed-array	1/26/2023 12:23 AM	File folder
minipass	1/26/2023 12:18 AM	File folder
minizlib	1/26/2023 12:18 AM	File folder
mkdirp	1/26/2023 12:18 AM	File folder

node_modules/node-fetch directory



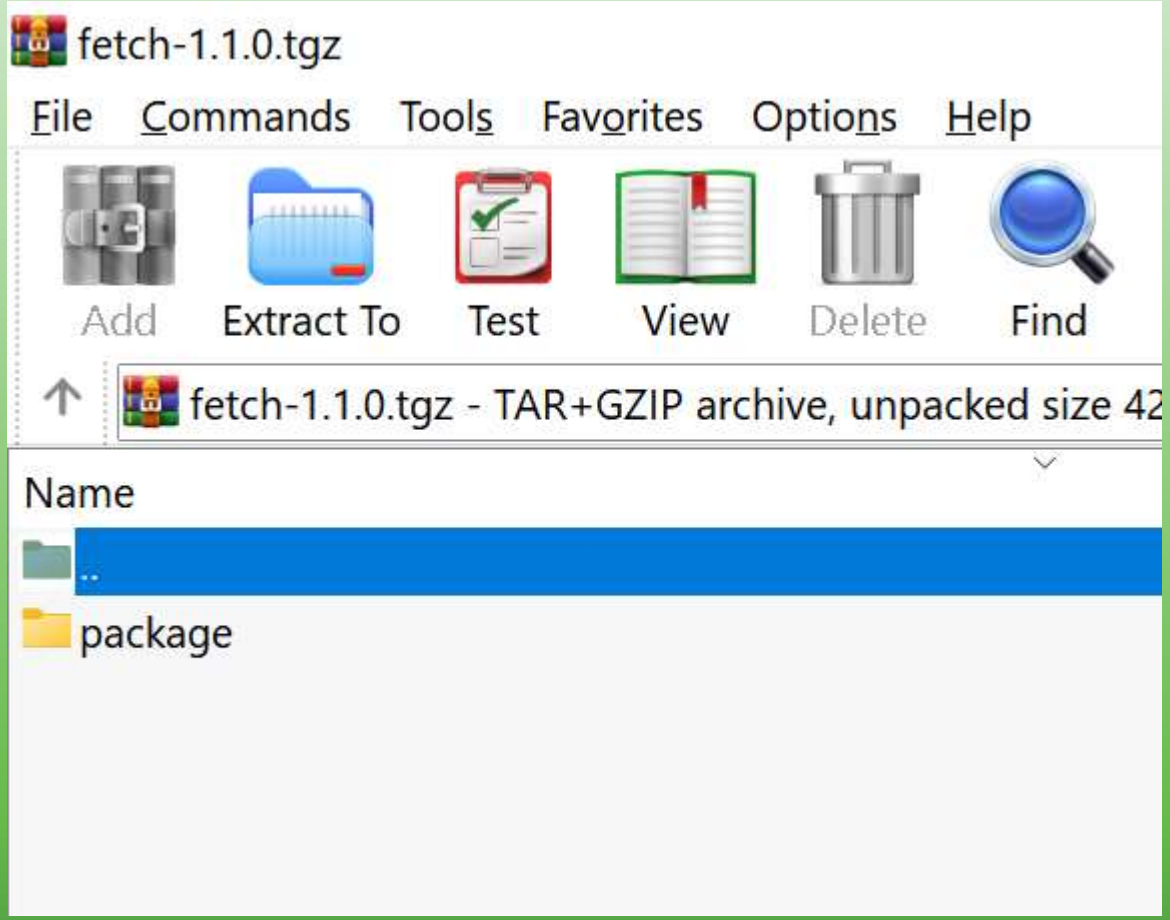
Downloading package tgz

https://registry.npmjs.org/**$name**/-/**$name-x.y.z**.tgz

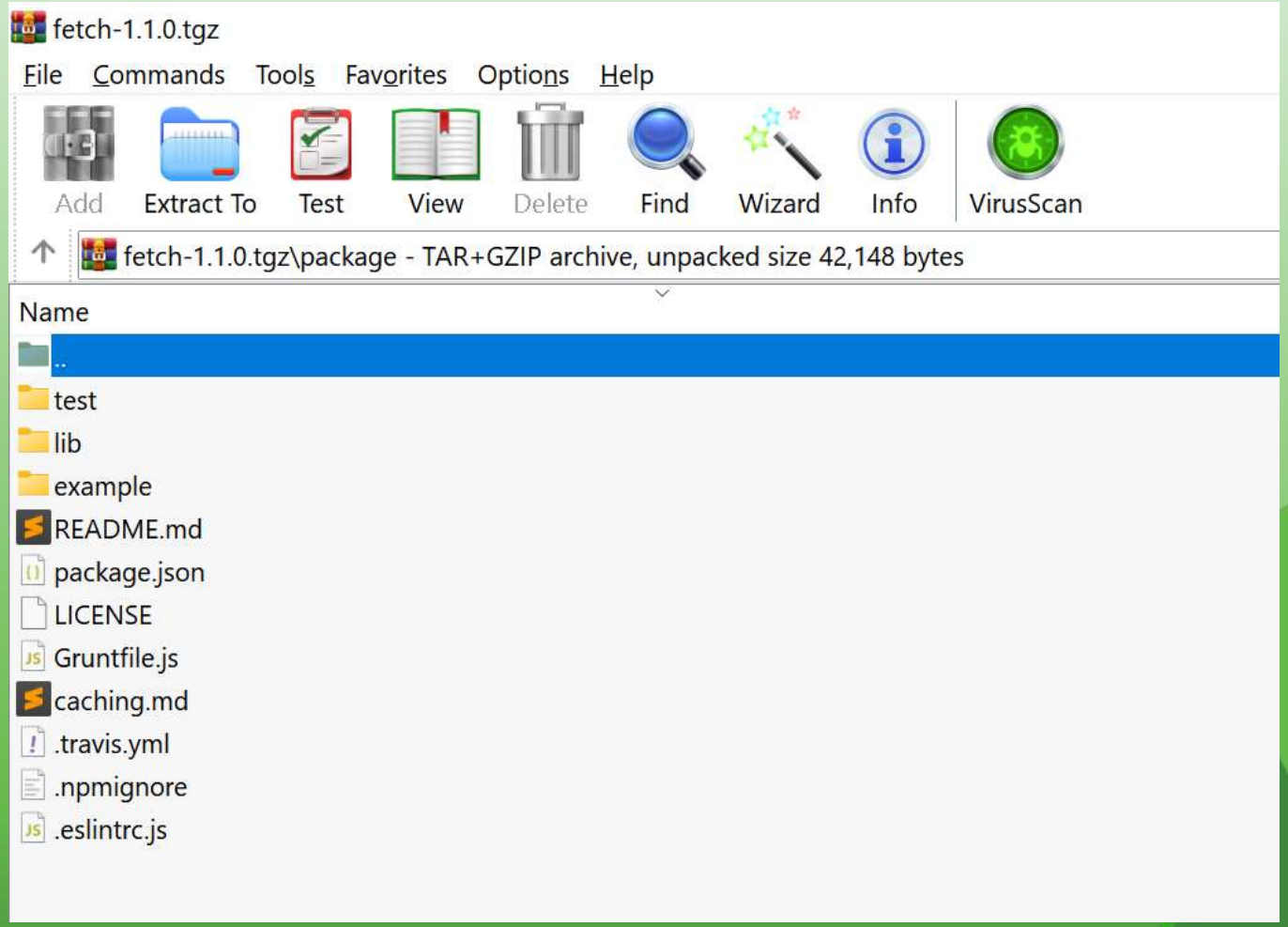
e.g: Version 1.1.0

https://registry.npmjs.org/node-fetch/-/node-fetch-1.1.0.tgz

In a package .tgz file



Files of a package tgz (inside the package dir)



Conclusion

Understanding how NPM and Yarn work and the process of creating your own package manager can be a valuable skill for developers.

Package managers play a vital role in managing dependencies and streamlining the development process.

By understanding how package managers work, you will be able to make better decisions when choosing a package manager for a project and troubleshoot issues more effectively.

Additionally, creating your own package manager can be a great learning experience and can open up new opportunities for contributing to open-source projects.

I hope this presentation has provided you with a deeper understanding of how NPM and Yarn work and the process of creating your own package manager.

References

- https://maxbase.org
- https://github.com/BaseMax/MyNodePackageManager

IN MEMORY TO

My supervisor, Iranian mathematician Prof. **Ali Reza Ashrshi**

Vice-President of the International Academy of Mathematical Chemistry

Thank you for your attention.

Tabatabaei Historical House,
Kashan, I.R.

For any questions in the future:
maxbasecode@gmail.com