Professional Comparison: npm vs Yarn for Angular Development.

Author: Mahmoud Alfaiyumi

Date: 29/1/2025

Table of Contents

Introduction1
Overview of npm1
Overview of Yarn2
Feature Comparison: npm vs Yarn2
Which is Better for Angular?
When to Use npm or Yarn3
When to Use npm
When to Use Yarn4
Installation Guides4
npm4
Yarn4
Usage in Angular Projects5
Creating a New Angular Project5
Common Commands5
Conclusion5

Introduction

Package managers like **npm** (Node Package Manager) and **Yarn** are essential tools for modern JavaScript development. They streamline dependency management, ensuring projects remain consistent and efficient. This article compares npm and Yarn, focusing on their suitability for **Angular** projects, and provides installation guides, usage examples, and actionable recommendations.

Overview of npm

- Developed By: npm, Inc. (now part of GitHub/Microsoft).
- Release Year: 2010.
- Key Features:
- Largest registry of JavaScript packages.
- Built into Node.js installations.
- Supports package-lock.json for deterministic installs.
- Integrated with npx for executing packages.

Detailed Overview:

npm (Node Package Manager) is the default package manager for Node.js and has been a cornerstone of the JavaScript ecosystem since its inception. It provides access to the largest registry of opensource libraries and tools, making it a go-to choice for developers. npm uses a package.json file to manage project dependencies and scripts, and it introduced the package-lock.json file to ensure consistent installs across environments. While npm has historically been slower due to its sequential installation process, recent updates have improved its performance significantly. Its tight integration with Node.js and widespread adoption make it a reliable choice for many developers.

Overview of Yarn

- Developed By: Facebook (now maintained by OpenJS Foundation).
- Release Year: 2016.
- Key Features:
- Faster installations via parallel downloads.
- Deterministic dependency resolution with yarn.lock.
- Workspaces for monorepo support.
- Enhanced security with integrity checks.

Detailed Overview:

Yarn was developed by Facebook in collaboration with Google, Exponent, and Tilde to address some of npm's limitations, particularly in performance and consistency. Yarn introduced a faster installation process by leveraging parallel downloads and a global cache. It also introduced the yarn.lock file, which ensures deterministic dependency resolution, meaning every install produces the exact same file structure. Yarn's workspaces feature makes it a strong contender for monorepo projects, allowing developers to manage multiple packages within a single repository. Additionally, Yarn emphasizes security with checksum verification and offers advanced features like Plug'n'Play (PnP) and zero-install, which further optimize the development workflow. Yarn's modern approach and corporate backing have made it a popular alternative to npm.

Feature Comparison: npm vs Yarn

Criteria	NPM	Yarn
Performance	Slower (sequential installs)	Faster (parallel installs)
Lockfile	package-lock.json	yarn.lock
Security	Basic audit tools	Checksum verification
CLI Usability	Verbose output	Clean, concise output
Offline Cache	Manual configuration	Built-in zero-install feature
Monorepo Support	Limited	Native workspaces
Community	Larger ecosystem	Growing, corporate-backed

Which is Better for Angular?

Angular CLI defaults to npm, but Yarn is equally compatible. Below is a framework-specific comparison:

Factor	NPM	Yarn
CLI Integration	Default (ng new)	Usepackage-manager=yarn
Install Speed	Moderate	Faster (caching advantage)
Dependency Consistency	Relies on package-lock.json	Strict yarn.lock enforcement
Team Collaboration	Suitable	Ideal for large teams

Recommendation:

- Use **Yarn** for large-scale Angular projects requiring speed and strict dependency control.

- Use **npm** for simplicity or if already integrated into your workflow.

When to Use npm or Yarn

Choosing between npm and Yarn depends on your project requirements, team preferences, and development workflow. Below are scenarios to help you decide:

When to Use npm

1. Small Projects: For smaller projects or prototypes, npm's simplicity and default integration with Node.js make it a convenient choice.

2. **Familiarity**: If your team is already comfortable with npm, sticking with it can reduce the learning curve.

3. **Ecosystem Integration**: npm is deeply integrated with the Node.js ecosystem, making it ideal for projects relying heavily on Node.js tools.

4. Quick Setup: Since npm comes pre-installed with Node.js, it requires no additional setup.

When to Use Yarn

1. **Large Projects**: Yarn's speed and efficient dependency resolution make it ideal for large-scale projects with numerous dependencies.

2. **Monorepos**: If you're working with monorepos, Yarn's native workspaces provide better support compared to npm.

3. **Team Collaboration**: Yarn's deterministic installs and strict lockfile enforcement ensure consistency across team members.

4. **Performance-Critical Applications**: Yarn's parallel downloads and caching mechanisms significantly reduce installation times.

5. Advanced Features: If you need features like Plug'n'Play (PnP) or zero-install, Yarn is the better choice.

Installation Guides

npm

1. Install Node.js (includes npm):

```
- Windows/macOS: Download from nodejs.org.
```

- Linux:

```
sudo apt-get update
sudo apt-get install nodejs npm
```

2. Verify Installation:

node -v npm -v

Yarn

1. Install via npm:

npm install -g yarn

2. Verify Installation:

yarn --version

Usage in Angular Projects

Creating a New Angular Project

- With npm:

ng new my-angular-app

- With Yarn:

ng new my-angular-app --package-manager=yarn

Common Commands

Action	NPM	Yarn
Install Dependencies	npm install	yarn install
Add a Package	npm install <package></package>	yarn add <package></package>
Remove a Package	npm uninstall <package></package>	yarn remove <package></package>
Run Scripts	npm run <script></script>	

Conclusion

Both npm and Yarn excel in managing dependencies, but **Yarn** edges ahead for Angular projects due to its speed, reliability, and advanced features. However, npm remains a solid choice for developers prioritizing ecosystem familiarity. Evaluate your project's scale and team needs to make an informed decision.