NPM vs NVM vs NPX

Instructor - Pramod

📦 What is `npm`?

🔹 Full Form:

npm = Node Package Manager

🔹 What is it?

- A **tool** that comes bundled with **Node.js**
- It lets you **install, manage, and update packages** (libraries or tools) used in your JavaScript/Node.js projects.

🛠️ What is a Package?

A package is a collection of **code**, usually written by other developers, that solves a common problem.

📦 Examples of packages:

- `react` – React library
- `lodash` – utility functions
- `express` – backend framework

✅ Key Uses of `npm`:

Command	Description

`npm install <package>`	Install a package locally
`npm install -g <package>`	Install a package globally
`npm init`	Create a `package.json` file
`npm update`	Update packages
`npm uninstall <package>`	Remove a package

📂 Local vs Global Installation

📍 Local (used in your project only)

```
npm install react
```

- Installs `react` inside the `node_modules` folder
- Project-specific

🌍 Global (used anywhere on your computer)

```
npm install -g nodemon
```

- You can run `nodemon` from any terminal

📘 What is `package.json`?

It's a file that keeps track of all your project's packages, scripts, and metadata.

Example:

```
{
  "name": "my-app",
  "dependencies": {
    "react": "^18.2.0"
  }
}
```

What is npx?

🔹 Full Form:

npx = Node Package Execute

🔹 What is it?

- A tool that **runs Node.js packages without installing them permanently** in your project.
- Comes bundled with **npm 5.2.0 and above**.

Why use npx?

Some tools are only needed **once**, or you don't want to clutter your project with unnecessary packages.

Example: Create a new React app using CRA
Instead of doing:

```
npm install -g create-react-app
create-react-app my-app
```

You can simply do:

```
npx create-react-app my-app
```

👉 It downloads `create-react-app`, uses it, and removes it afterward.

✅ Common Use Cases of `npx`:

Task	Command
Run a one-time package	`npx create-react-app my-app`
Run a CLI from node_modules	`npx eslint .`
Test packages before installing	`npx cowsay Hello!`

🧾 Summary Table

Tool	Use	Installs Packages?	Scope
`npm`	To install/manage packages	Yes	Local/Global
`npx`	To run a package instantly	No (temporary use)	One-time

🎯 Real-World React Example

Using `npm`:

Install React manually:

```
npm install react react-dom
```

Then configure the project yourself.

Using `npx`:

Start a full project instantly:

```
npx create-react-app my-app
```

🧠 Analogy (for Students):

- `npm` is like **installing an app** on your phone (you keep it).
- `npx` is like **using an app temporarily in the browser** (like a demo version).

🧰 What is NVM?

🔹 Full Form:

NVM = Node Version Manager

🔹 What is it used for?

NVM is a tool that allows you to:

- **Install multiple versions** of Node.js on your machine
- **Switch** between different versions easily
- **Manage versions** per project

Why do we need NVM?

Different projects may require **different versions of Node.js**.

Real-world Example:

- Project A needs **Node 14**
- Project B needs **Node 18**
- Your system has Node 16 by default

Without NVM: 😩 You would have to uninstall/reinstall different versions repeatedly
With NVM: 😎 You can switch versions in seconds!

⚙️ Installing NVM (Linux / macOS)

Run this command in terminal:

```
curl -o-
https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.5/install.sh | bash
```

Then reload terminal config:

```
source ~/.bashrc   # or ~/.zshrc depending on your shell
```

⚙️ Installing NVM (Windows)

Use **nvm-windows**:

1. Download from: https://github.com/coreybutler/nvm-windows/releases
2. Run installer
3. Use it from Command Prompt or PowerShell

🛠️ Common NVM Commands

Command	**Description**
`nvm install <version>`	Install a specific version of Node.js
`nvm use <version>`	Use that version in the current terminal
`nvm ls`	List all installed versions
`nvm ls-remote`	Show all available Node.js versions
`nvm uninstall <version>`	Remove a version from your system
`nvm current`	Show the currently active version

Example Workflow

```
nvm install 18
nvm install 14
nvm use 14
node -v  # Outputs: v14.x.x
nvm use 18
node -v  # Outputs: v18.x.x
```

Important Notes

- NVM **only works in the terminal** where you run `nvm use`.
- To make a version default globally:

```
nvm alias default 18
```

Analogy (for Students):

Think of **NVM** like a **remote control** for Node.js:

- Press button `14` → You're using Node v14
- Press button `18` → Switch to Node v18 instantly

Summary

Tool	Purpose
`npm`	Install/manage JS packages
`npx`	Run JS packages instantly
`nvm`	Manage multiple Node.js versions