Node.js Tools: npm & npx

Node.js/JavaScript package manager

- **Node Package Manager** - JavaScript's package ecosystem
- **World's largest** software registry (2M+ packages)
- Comes bundled with Node.js
- **Think of it as:** App Store for JavaScript libraries

Installing Node.js

Before using npm/npx, you need Node.js installed

Method 1: Official Website (Recommended for Beginners)

Visit: https://nodejs.org

- **LTS version** (Long Term Support) - Stable, recommended
- **Current version** - Latest features, may be unstable

For students: Always choose the LTS version!

Method 2: Package Managers (Advanced)

Windows (using Chocolatey)

```
choco install nodejs
```

macOS (using Homebrew)

```
brew install node
```

Ubuntu/Debian Linux

```
sudo apt update
sudo apt install nodejs npm
```

Verify Installation

Check if Node.js is installed:

```
node --version
# Should output: v18.17.0 (or similar)
```

Check if npm is installed:

```
npm --version
# Should output: 9.6.7 (or similar)
```

If both commands work → You're ready to go! 🎉

Package.json - Your Project's DNA

```
{
  "name": "my-web-app",
  "version": "1.0.0",
  "dependencies": {
    "express": "^4.18.0",
    "lodash": "^4.17.21"
  },
  "devDependencies": {
    "jest": "^29.0.0"
  },
  "scripts": {
    "start": "node app.js",
    "test": "jest"
  }
}
```

```
"express": "4.18.0"
```

- Installs exactly version 4.18.0
- No automatic updates - even for bug fixes

```
"express": "^4.18.0"
```

- Installs 4.18.0 or higher
- But stays within major version 4.x.x
- Allows: 4.18.1, 4.19.0, 4.20.5
- Blocks: 5.0.0 (major version change)

Essential commands

```
# Initialize new project
npm init

# Install packages
npm install express          # Save to dependencies
npm install -D jest          # Save to devDependencies
npm install -g nodemon       # Install globally

# Install from package.json
npm install

# Update packages
npm update
```

Package-lock.json - The Exact Recipe

Think of it as: Detailed cooking instructions vs. general recipe

- **package.json**: "Add some flour" (flexible)
- **package-lock.json**: "Add exactly 2.5 cups of King Arthur flour" (precise)

Automatically created when you run `npm install`

Why Package-lock.json Exists

The Problem: `^4.18.0` means "4.18.0 or higher"

- **Monday**: You install Express 4.18.0
- **Tuesday**: Your teammate installs Express 4.19.1 (newer version released)
- **Result**: Different versions = potential bugs! 🐛

The Solution: package-lock.json locks exact versions

Real Example: package.json vs package-lock.json

package.json (flexible)

```
{
  "dependencies": {
    "express": "^4.18.0",
    "lodash": "^4.17.21"
  }
}
```

package-lock.json (exact)

```
{
  "dependencies": {
    "express": {
      "version": "4.18.2",
      "resolved": "https://registry.npmjs.org/express/-/express-4.18.2.tgz",
      "integrity": "sha512-5/PsL6iGPdfQ/lKM1UuielYgv3BUoJfz1aUwU9vHZ+J7gyvwdQXFEBIEIaxeGf0GIcreATNyBExtalisDbuMqQ=="
    }
  }
}
```

Key Benefits of Package-lock.json

✅ Reproducible Builds

- Everyone gets identical dependencies
- Same versions on dev, staging, production

✅ Faster Installs

- Skip version resolution (already solved)
- Download exact files from cache

✅ Security

- Prevents malicious package substitution
- Integrity checksums verify authenticity

Best Practices with Package-lock.json

✅ DO:

- Commit `package-lock.json` to git
- Use `npm ci` in production/CI
- Let npm manage it automatically

❌ DON'T:

- Edit `package-lock.json` manually
- Delete it when facing conflicts
- Add it to `.gitignore`

npm install vs npm ci

npm install

- Uses `package.json` as source of truth
- Updates `package-lock.json` if needed
- Good for development

npm ci

- Uses `package-lock.json` as source of truth
- Fails if the lock file is outdated
- Perfect for production/CI environments

📦 Dependencies vs DevDependencies

Dependencies

- **Needed when app runs** (production)
- Without them, → app will break at runtime
- Saved to `"dependencies"` in `package.json`

Examples

- React
- Express
- Axios

```
npm install express axios
```

DevDependencies

Needed only during development

- Without them, → app can still run, but dev/build/testing won't work
- Saved to `"devDependencies"` in `package.json`

Examples

- Jest
- ESLint
- Webpack

```
npm install -D jest eslint nodemon
```

Quick Analogy

Type	Role	Example
dependencies	Ingredients 🥗 (must be in dish)	React, Axios
devDependencies	Kitchen tools 🔪 (needed to prepare, not serve)	Jest, Webpack

Local Installation (Default)

```
npm install express
npm install lodash
```

- **Installed in:** `./node_modules/` folder
- **Available to:** Current project only
- **Saved to:** `package.json` dependencies
- **Think of it as:** Project-specific tools in your toolbox

```
my-project/
├── node_modules/      ← Packages here
│   ├── express/
│   └── lodash/
├── package.json       ← Dependencies listed
└── app.js
```

Global Installation with -g

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

- **Installed in:** System-wide location
- **Available to:** All projects and command line
- **Not saved to:** `package.json`
- **Think of it as:** System tools in your workshop

Available everywhere in the terminal

For windows

```
C:\Users\YourName\AppData\Roaming\npm\node_modules
```

For Linux (WSL2)

```
~/.npm-global/lib/node_modules/  ← Packages here
├── nodemon/
├── create-react-app/
└── eslint/
```

npx: Run packages without installing

Real-World Analogy: Rental vs. Buying

- **npm install**: Buying a tool (permanent)
- **npx**: Renting a tool (temporary use)

When we don't want to install the tool, but use it once and throw it away, we use `npx` *instead of* `npm`.

npx replaces npm -g

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

npm -g (Global Install)

- Installs permanently on your system
- Available everywhere from the command line
- One version per system - can cause conflicts
- Takes up disk space permanently

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

npx (Execute Without Installing)

- Downloads temporarily and runs immediately
- No permanent installation
- Always gets the latest version
- No global pollution

npx is used for many applications, especially **React**.

```
# Create a React app without installing create-react-app
npx create-react-app my-app
# Run the latest version of a tool
npx cowsay "Hello Students!"
# Check if the website is up
npx is-up google.com
# Create a QR code
npx qrcode "https://github.com"
```