# Node.js

## 1. Node.js 개념

### 1.1 Node.js

- Chrome V8 자바스크립트 엔진으로 빌드된 자바스크립트 런타임 환경으로, 웹 브라우저 밖에서 자바스크립트 코드를 실행할 수 있게 해주는 서버 사이드 플랫폼입니다.
- 기존에 웹 브라우저 내에서만 실행되던 자바스크립트를 서버 환경에서도 실행할 수 있도록 만든 런타임 환경입니다.
- Node.js는 실시간 채팅, 협업 도구, 온라인 게임과 같은 실시간 애플리케이션 개발에 특히 적합하며 , REST API 서버, 마이크로서비스 아키텍처, 그리고 I/O가 많은 애플리케이션 개발에서 뛰어난 성능을 발휘합니다.
- 이벤트 기반 비동기 처리
  Node.js는 이벤트 기반(Event-driven) 아키텍처를 사용하여 특정 이벤트가 발생할 때 미리 등록된 콜백함수를 실행하는 방식으로 동작합니다.
- 논블로킹 I/O
  논블로킹(Non-blocking) I/O 모델을 통해 한 작업이 완료되기를 기다리지 않고 다른 작업을 동시에 처리할 수 있어 높은 성능을 보장합니다.
- 단일 스레드
  기본적으로 단일 스레드 이벤트 루프 모델을 사용하지만, 내부적으로는 멀티 스레드를 활용하여 효율적인 처리를 수행합니다.
- 단일 언어 사용
  프론트엔드와 백엔드 모두에서 자바스크립트를 사용할 수 있어 개발자의 생산성이 향상되고 러닝 커브가 줄어듭니다.

- 경량화와 확장성
  모듈화된 구조로 인해 매우 가볍고, 새로운 모듈 추가나 업데이트가 간단하여 확장성이 뛰어납니다.

### 1.2 Node.js vs JRE

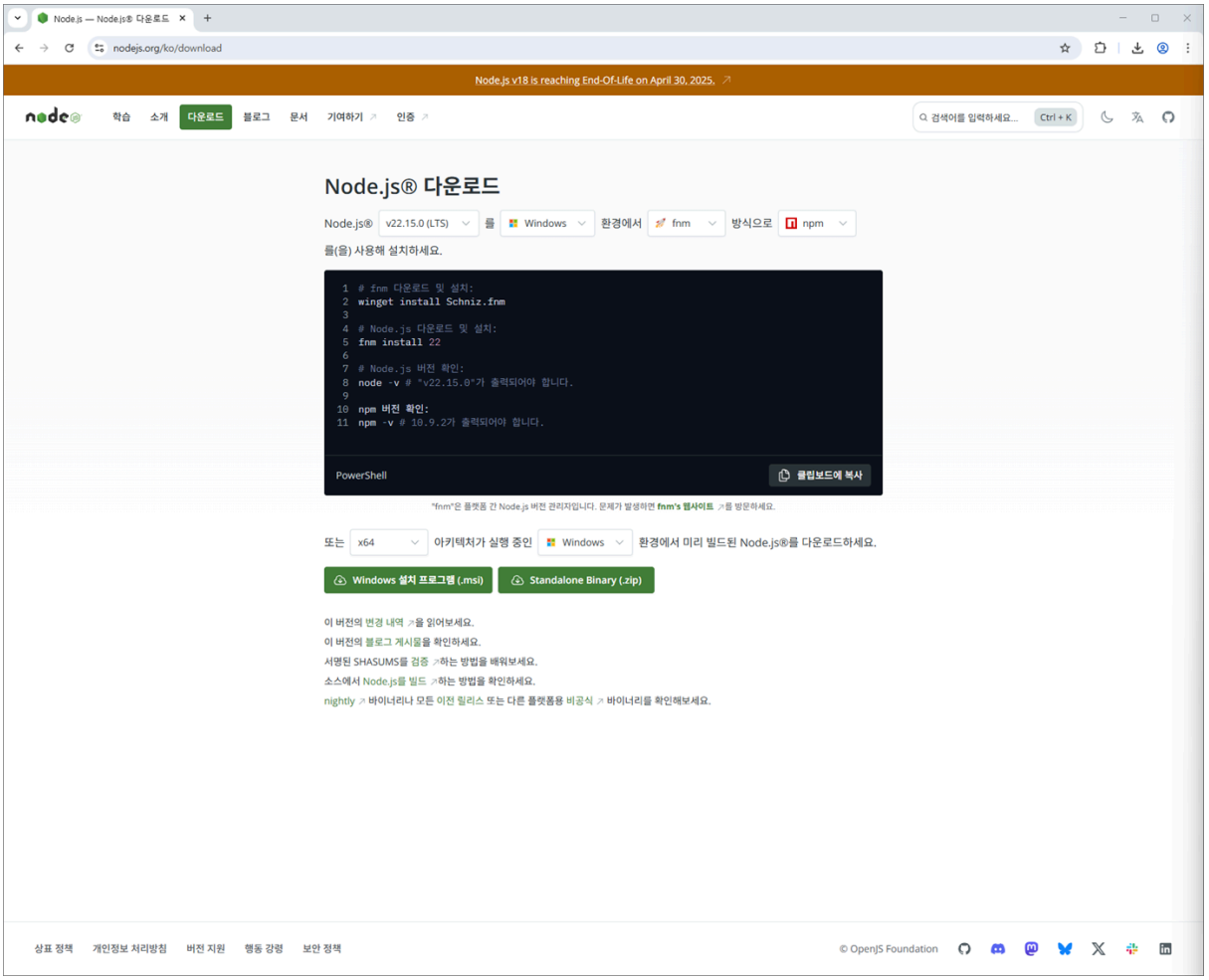
Node.js와 JRE는 매우 비슷한 목적을 가지고 있습니다. 둘 다 특정 언어로 작성된 코드를 실행하기 위한
런타임 환경(Runtime Environment)을 제공하는 역할을 합니다.

| 구분 | Node.js | JRE (Java Runtime Environment) |
|---|---|---|
| 기본 목적 | JavaScript 런타임 환경 | Java 런타임 환경 |
| 실행 엔진 | Chrome V8 JavaScript 엔진 | Java Virtual Machine (JVM) |
| 실행 방식 | 인터프리터 방식 (Just-in-time 컴파일) | 바이트코드 컴파일 후 실행 |
| 스레드 모델 | 단일 스레드 이벤트 루프 | 멀티 스레드 |
| 동시성 처리 | 비동기 논블로킹 I/O | 스레드 기반 동시성 |
| I/O 성능 | 뛰어남 (비동기 처리) | 블로킹 I/O로 인한 오버헤드 가능 |
| CPU 집약적 작업 | 제한적 (단일 스레드) | 뛰어남 (멀티 스레드 활용) |
| 메모리 사용 | 상대적으로 경량 | 상대적으로 무거움 |
| 패키지 관리 | npm, yarn | Maven, Gradle |
| 주요 활용 분야 | 웹 서버, REST API, 실시간 애플리케이션 | 엔터프라이즈 애플리케이션, 대용량 시스템 |
| 학습 곡선 | 상대적으로 낮음 | 상대적으로 높음 |
| 대표 프레임워크 | Express.js, Koa.js, NestJS | Spring, Spring Boot, Hibernate |

# 2. Node.js 설치

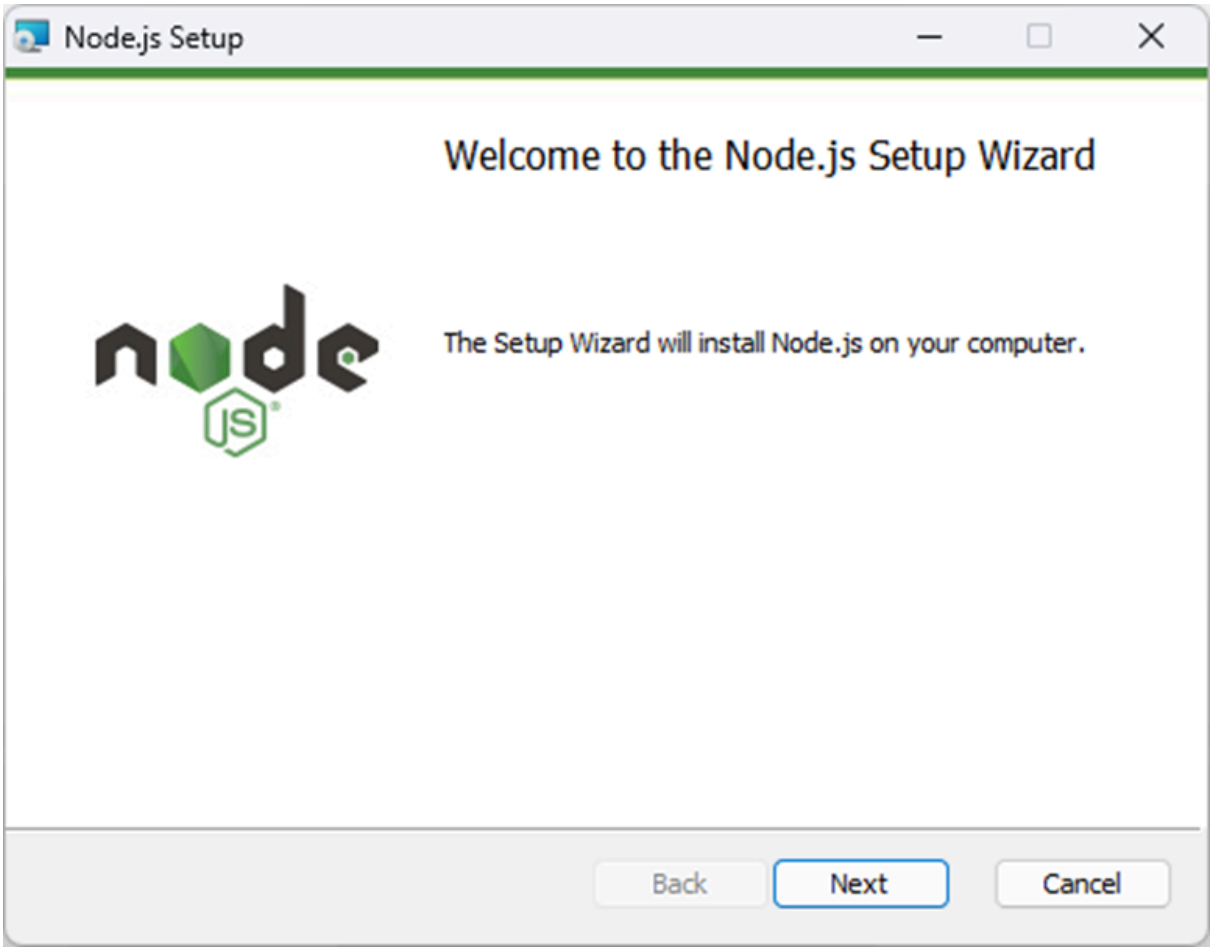
## 2.1 Node.js 다운로드

- https://nodejs.org/ko/download

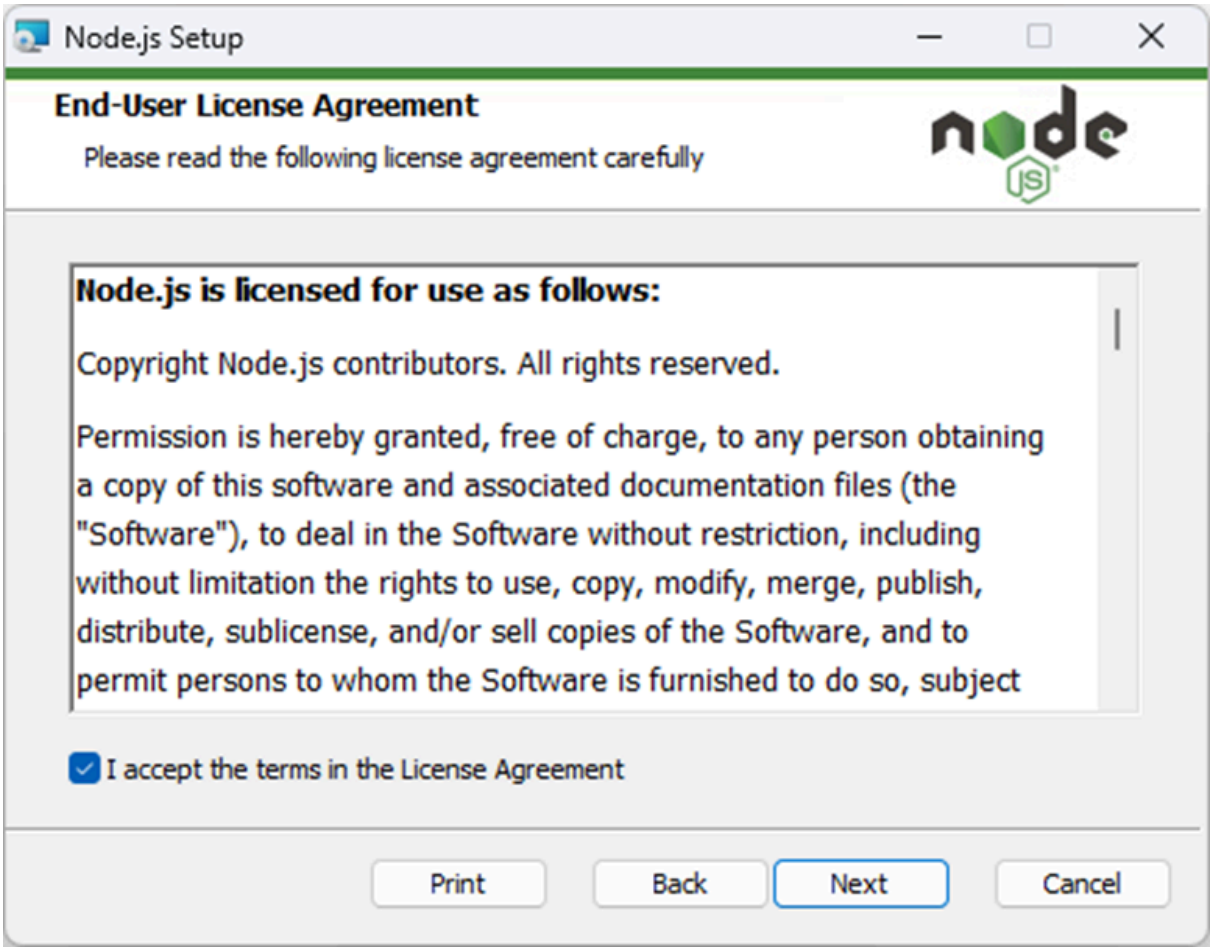



## 2.2 Node.js 설치

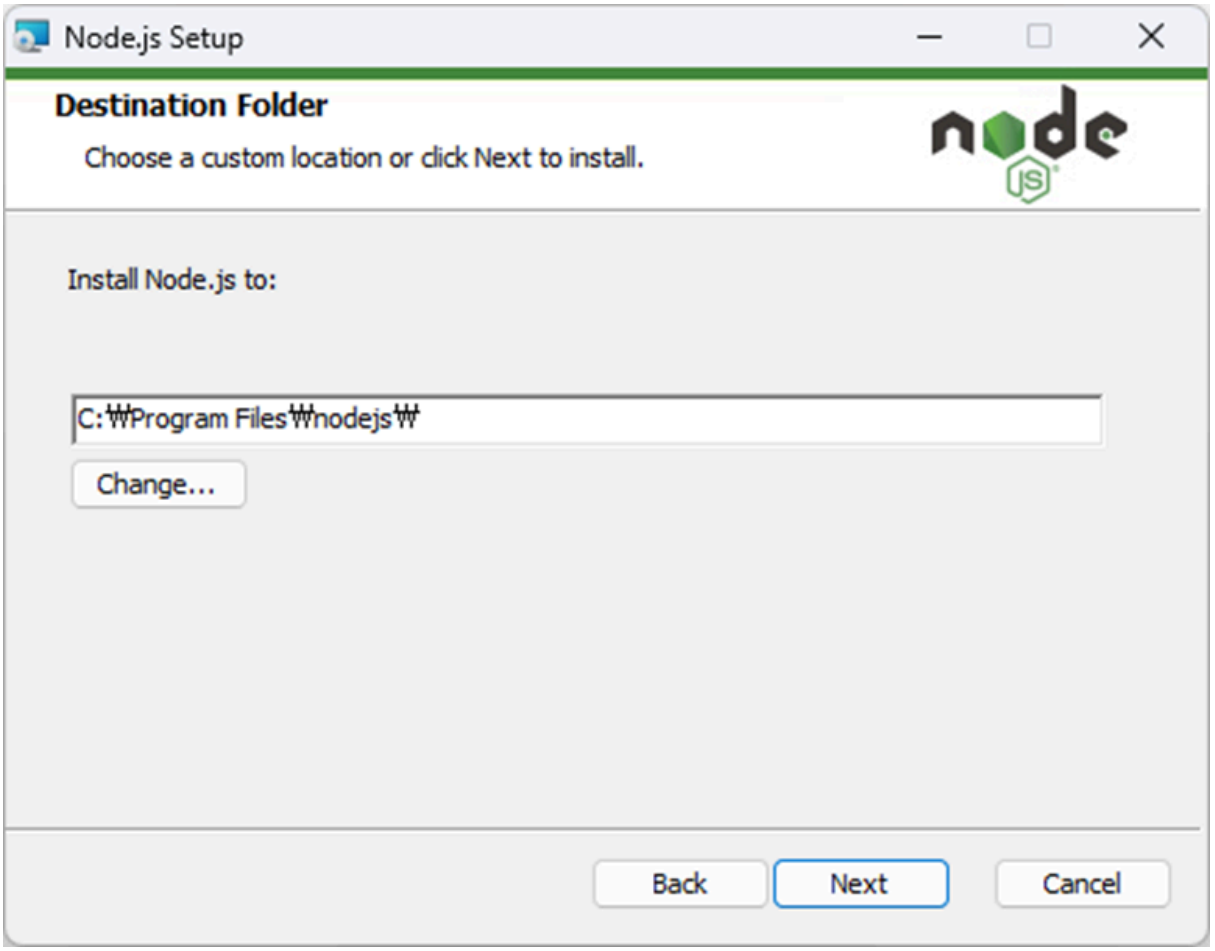
- 설치 시작


Node.js Setup
Welcome to the Node.js Setup Wizard
The Setup Wizard will install Node.js on your computer.
Back
Next
Cancel

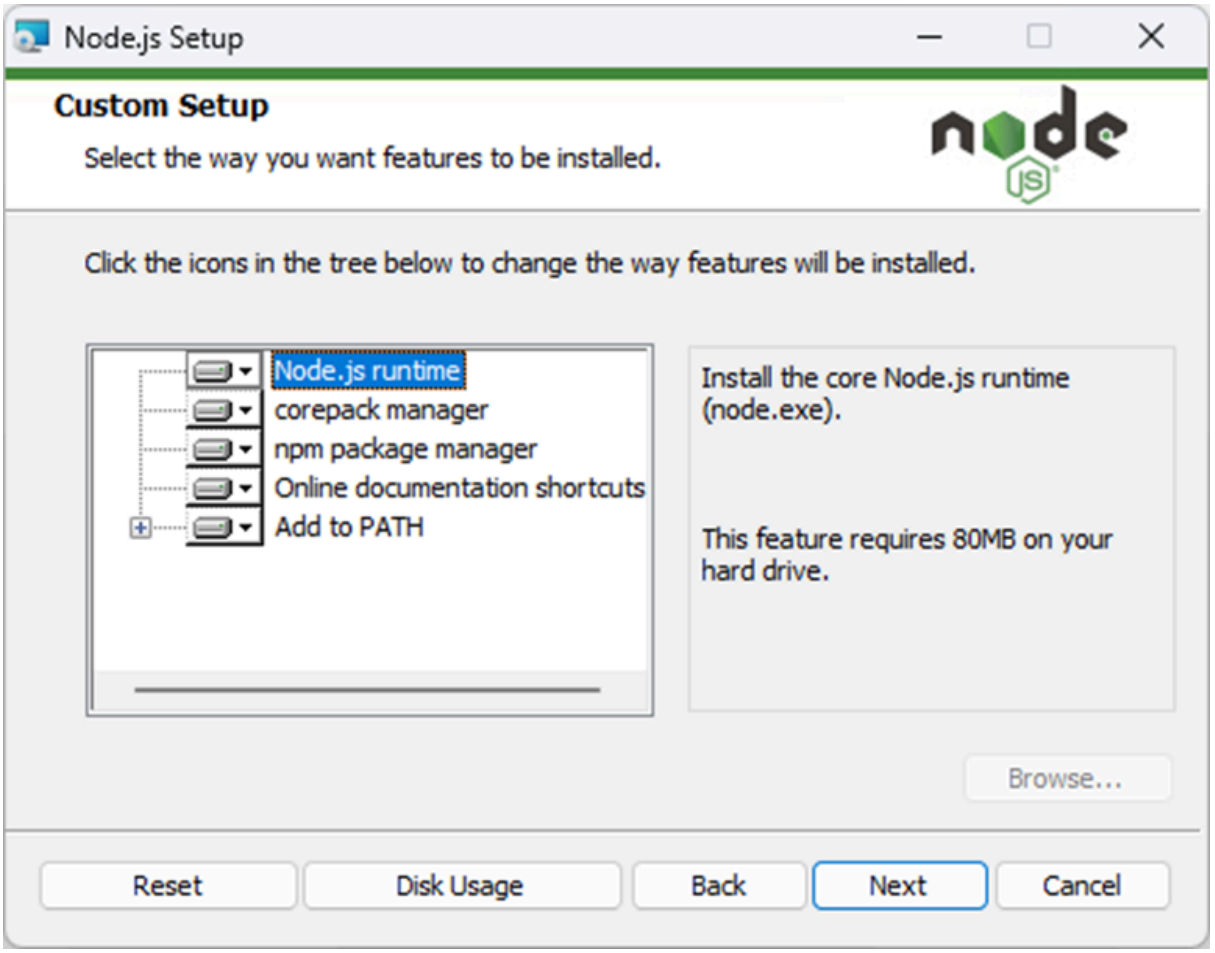
- 약관 동의


Node.js Setup
End-User License Agreement
Please read the following license agreement carefully
Node.js is licensed for use as follows:
Copyright Node.js contributors. All rights reserved.
Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject
I accept the terms in the License Agreement
Print
Back
Next
Cancel

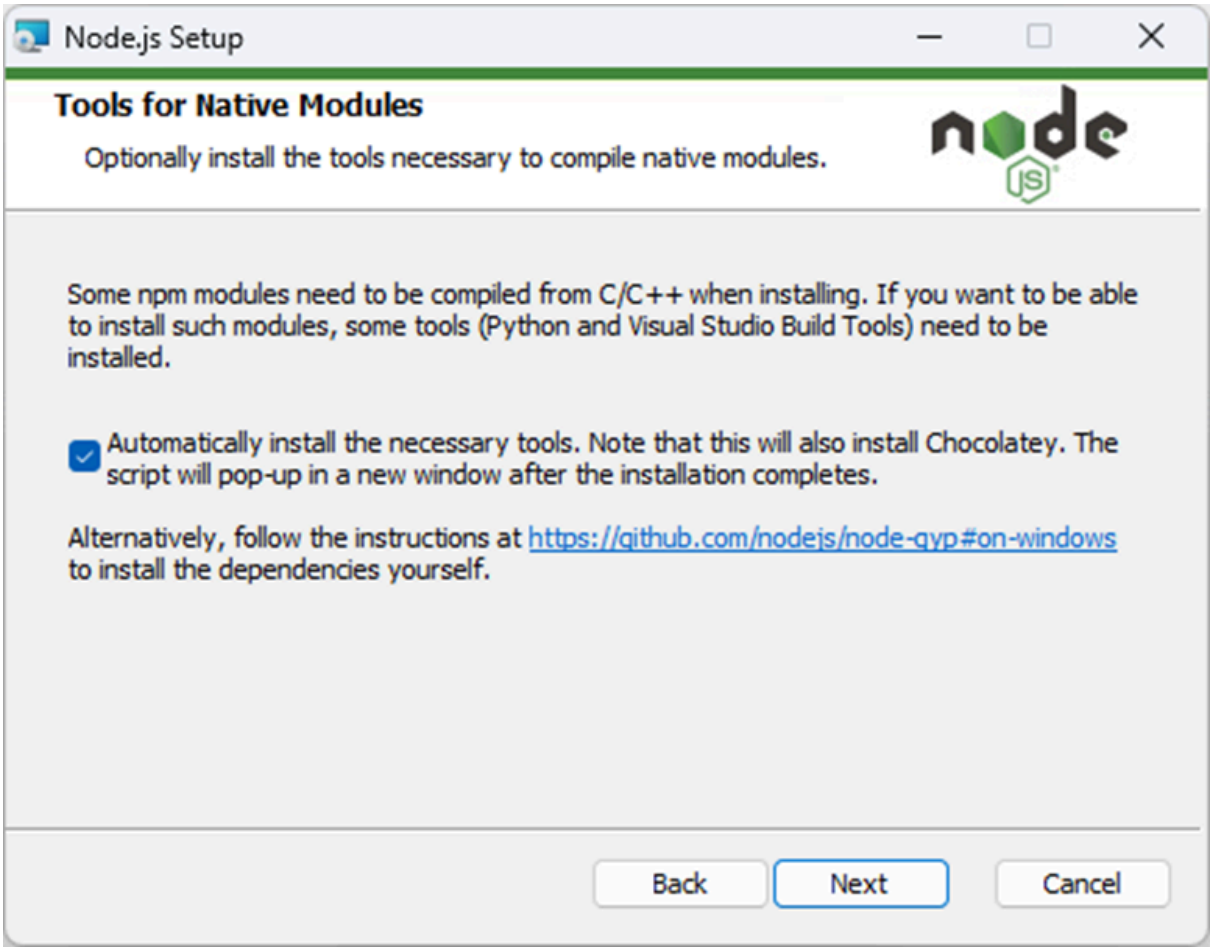
- 설치 경로


Node.js Setup
Destination Folder
Choose a custom location or click Next to install.
Install Node.js to:
C:₩Program Files₩nodejs₩
Change...
Back
Next
Cancel

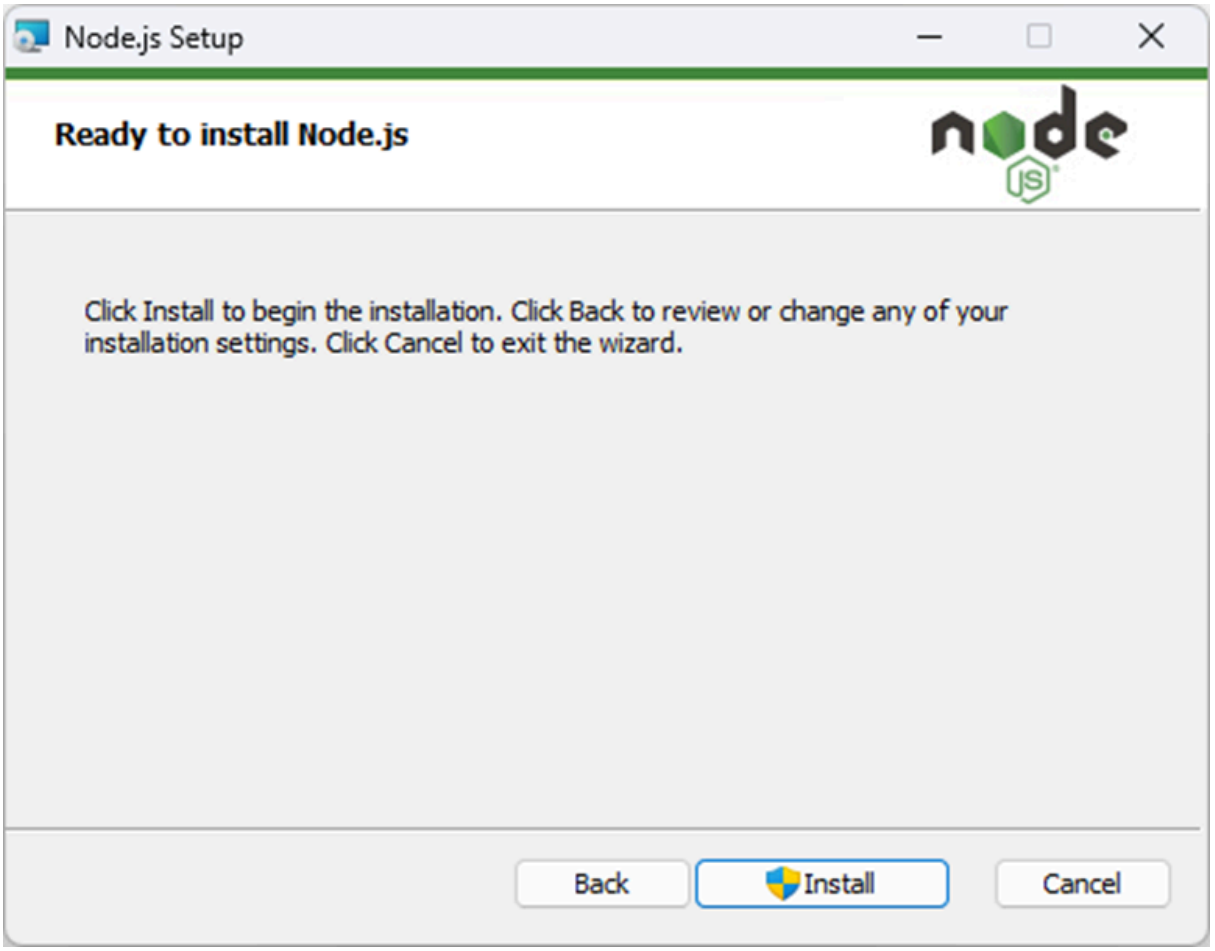
- 설치 리스트


Node.js Setup
Custom Setup
Select the way you want features to be installed.
Click the icons in the tree below to change the way features will be installed.
Node.js runtime
corepack manager
npm package manager
Online documentation shortcuts
Add to PATH
Install the core Node.js runtime (node.exe).
This feature requires 80MB on your hard drive.
Browse...
Reset
Disk Usage
Back
Next
Cancel

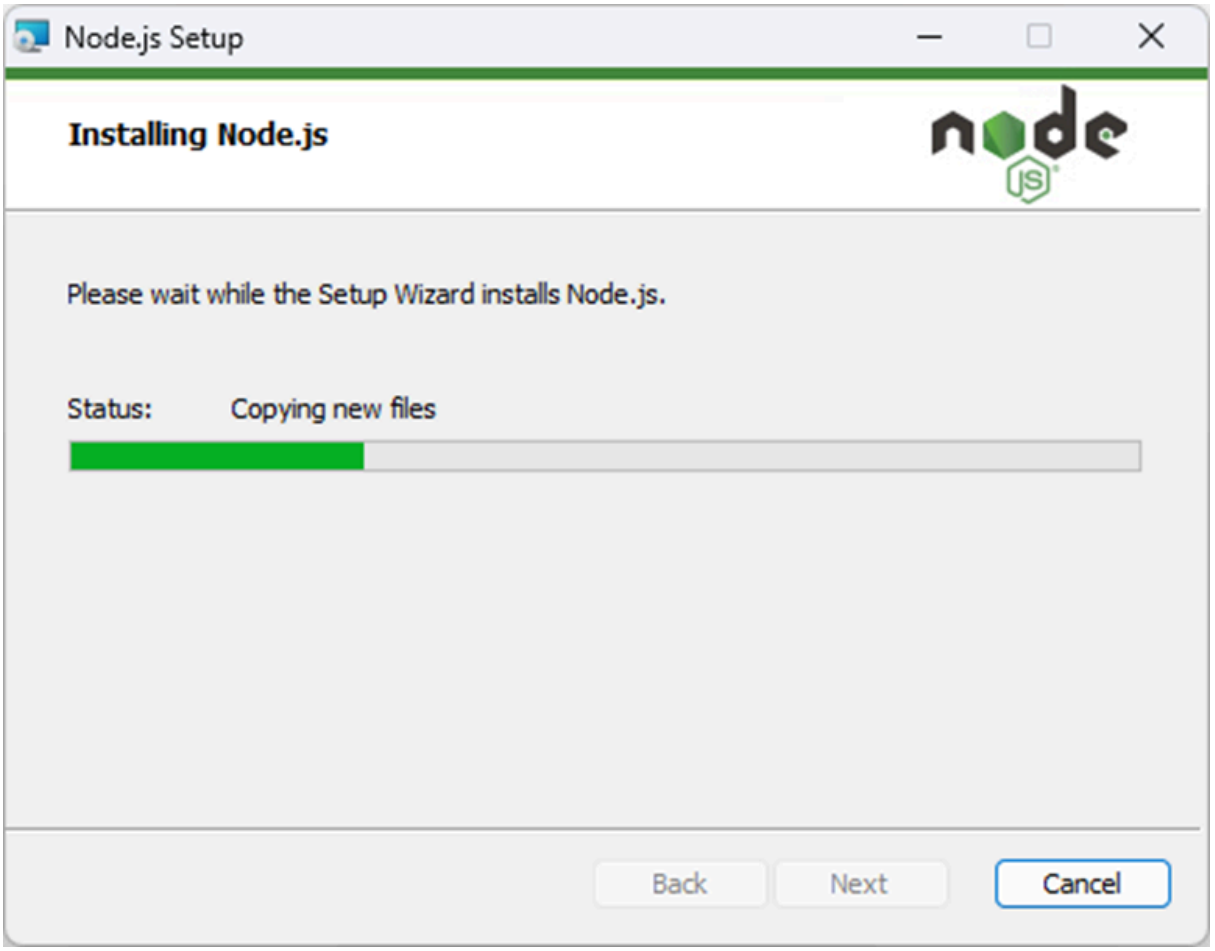
- 자동 툴 설치 체크


Node.js Setup
Tools for Native Modules
Optionally install the tools necessary to compile native modules.
Some npm modules need to be compiled from C/C++ when installing. If you want to be able to install such modules, some tools (Python and Visual Studio Build Tools) need to be installed.
Automatically install the necessary tools. Note that this will also install Chocolatey. The script will pop-up in a new window after the installation completes.
Alternatively, follow the instructions at https://github.com/nodejs/node-gyp#on-windows to install the dependencies yourself.
Back
Next
Cancel

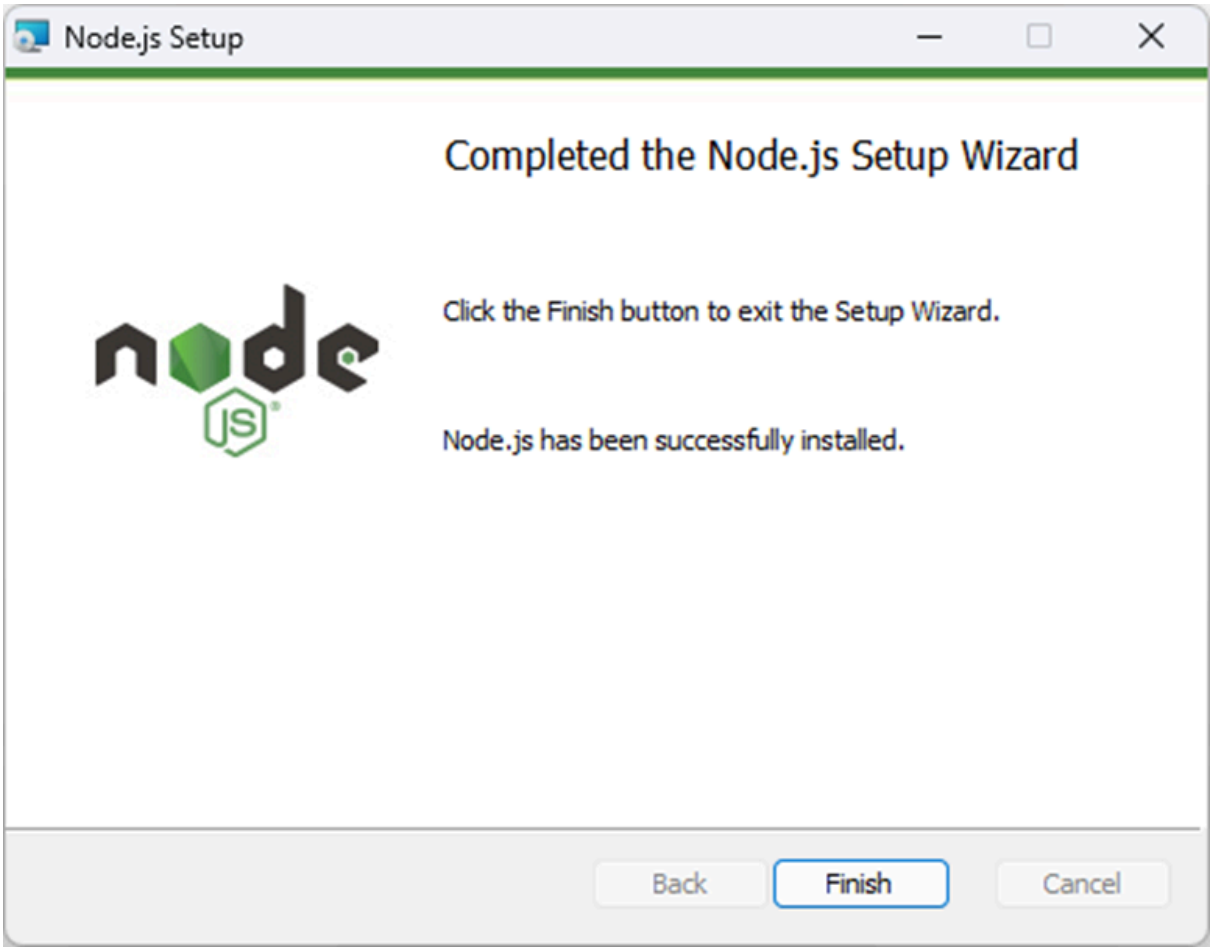
- Install 시작


Node.js Setup
Ready to install Node.js
Click Install to begin the installation. Click Back to review or change any of your installation settings. Click Cancel to exit the wizard.
Back
Install
Cancel

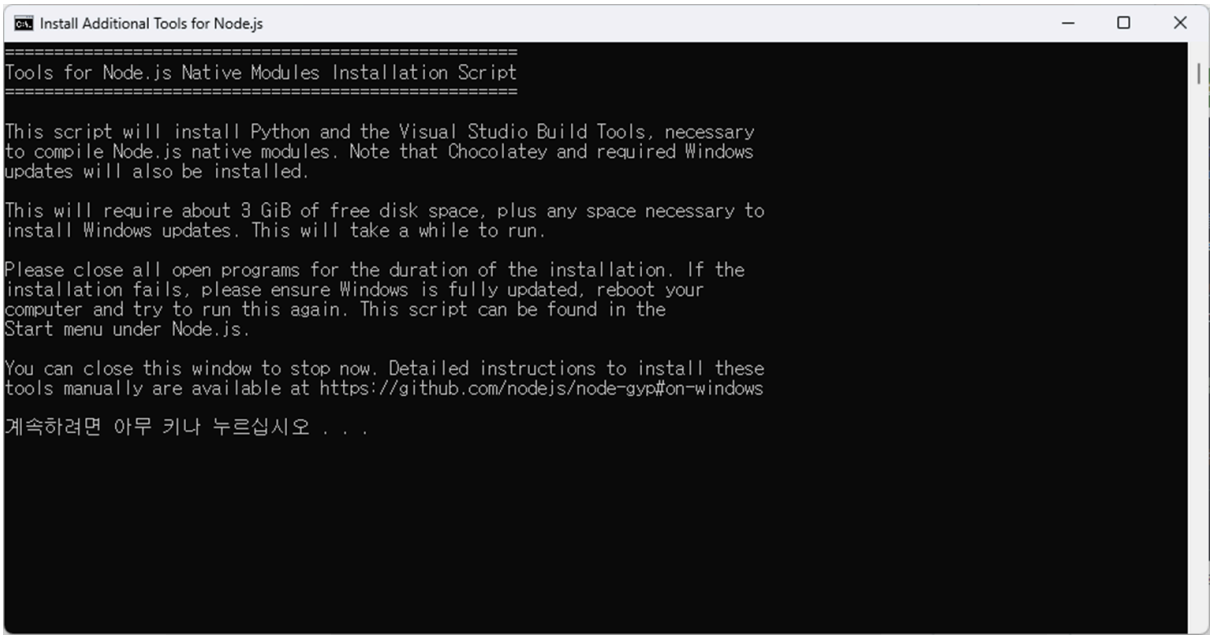
- Installing...




- Install Finish


Node.js Setup
Completed the Node.js Setup Wizard
Click the Finish button to exit the Setup Wizard.
Node.js has been successfully installed.
Back
Finish
Cancel

- 추가 툴 설치


Install Additional Tools for Node.js
Tools for Node.js Native Modules Installation Script
This script will install Python and the Visual Studio Build Tools, necessary
to compile Node.js native modules. Note that Chocolatey and required Windows
updates will also be installed.
This will require about 3 GiB of free disk space, plus any space necessary to
install Windows updates. This will take a while to run.
Please close all open programs for the duration of the installation. If the
installation fails, please ensure Windows is fully updated, reboot your
computer and try to run this again. This script can be found in the
Start menu under Node.js.
You can close this window to stop now. Detailed instructions to install these
tools manually are available at https://github.com/nodejs/node-gyp#on-windows
계속하려면 아무 키나 누르십시오 . . .

- 설치 확인

```
node -v
npm -v
```

# 3. Node.js를 활용한 애플리케이션 만들기

## 3.1 프로젝트 폴더 생성

```
mkdir hello-world-app
cd hello-world-app
```

## 3.2 프로젝트 초기화

```
npm init -y
```

- author 항목을 제외한 나머지 항목은 디폴트로 처리합니다.
- 이 명령어는 기본 설정으로 package.json 파일을 자동 생성합니다.

## 3.3 Express.js 설치

```
npm install express
```

- 더 간단하고 실용적인 방법을 사용하기 위해서 Express.js를 설치합니다.
- Express.js를 설치하고 나면 node_modules 폴더와 package-lock.json 파일이 생성됩니다.

## 3.4 app.js 서버 파일 생성

```
// Express 모듈 불러오기
const express = require('express');
const app = express();
const PORT = 3000;

// 루트 경로에 대한 GET 요청 처리
app.get('/', (req, res) => {
  res.send('<h1>Hello World!</h1>');
});

// 추가 라우트 예제
app.get('/api/hello', (req, res) => {
  res.json({ message: 'Hello World from API!' });
});

// 서버 시작
app.listen(PORT, () => {
  console.log(`Express 서버가 http://localhost:${PORT} 에서 실행 중입니다.`);
});
```

### 3.5 package.json 스크립트 추가 (선택사항)

```
{
  "scripts": {
    "start": "node app.js"
  }
}
```

- 필수사항이 아니고, 선택사항입니다.
- 스크립트를 추가해 두면 `npm run start` 명령을 이용해 애플리케이션을 실행할 수 있습니다.

### 3.6 app.js 실행 및 결과 확인

```
# 1.1 app.js 실행
node app.js

# 1.2 js 생략 가능
node app



# 2.1 package.json 스크립트 작성 시
npm run start

# 2.2 run 생략 가능
npm start
```

### 3.7 결과 확인

- 웹 브라우저에서 `http://localhost:3000`으로 접속하면 **"Hello World!"** 메시지가 화면에 출력됩니다.
- 웹 브라우저에서 `http://localhost:3000/api/hello`으로 접속하면 **{ message: 'Hello World from API!' }** JSON 데이터를 받아옵니다.

# 4. npm, npx

## 4.1 npm 개념

- Node Package Manager
- 패키지 관리자 역할을 하는 도구입니다.
- Node.js의 기본 패키지 관리자입니다. Node.js 설치 시 함께 설치됩니다.
- 패키지 설치, 버전 관리, 업데이트 등을 담당합니다.

## 4.2 npm 명령어

▼ `npm <command> help`

- command에 대해서 확인하는 명령입니다.
- `npm install help`, `npm init help` 등과 같이 확인하려는 command를 npm help 사이에 작성합니다.

▼ **`npm install <package>`**

- package를 설치하는 명령입니다.
- 옵션 -g : 글로벌 패키지에 package를 추가합니다.
  - 글로벌 패키지에 설치된 package는 모든 프로젝트에서 사용할 수 있습니다.
  - 글로벌 패키지 폴더 위치는 npm root -g 명령으로 확인할 수 있습니다.
    - `Windows` : C:\Users\<계정>\AppData\Roaming\npm
    - `macOS` : /usr/local/lib/node_modules
- `npm install <url>` 명령으로 지정한 주소(url)에 있는 package를 설치할 수 있습니다.
  - 주로 github에 저장된 패키지를 설치할 때 활용합니다.
- 축약 명령 : `npm i <package>`

▼ **`npm ci`**

- package-lock.json 파일을 기반으로 패키지를 설치하는 명령입니다.
- `npm install` 명령은 설치 시 세부 버전이 달라질 수도 있으므로 버전을 고정하고자 한다면 `npm ci` 명령을 사용합니다.

▼ **`npm update <package>`**

- 설치한 package를 업데이트 하는 명령입니다.
- 옵션 --save 사용으로 package.json 파일의 버전도 함께 수정 가능합니다.
  - 정확한 패키지 버전은 package-lock.json 파일에서 관리하므로 옵션 --save는 필수가 아닙니다.

▼ **`npm uninstall <package>`**

- 설치된 package를 제거하는 명령입니다.
- 글로벌 패키지에 설치한 pagkage는 옵션 -g를 추가해서 제거해야 합니다.
- 축약 명령 : `npm un <package>`, `npm r <package>`

▼ **`npm run <script>`**

- npm run 명령어는 package.json 파일의 scripts 섹션에 정의된 커스텀 스크립트를 실행하는 명령어입니다.

```
"scripts": {
  "start": "node server.js",
  "build": "webpack --mode production",
  "test": "jest",
  "dev": "webpack serve --open"
}
```

- `npm run <스크립트명>` 형식으로 실행합니다.
- 예시: `npm run build`, `npm run dev`
- 예외적으로 start, test는 많이 사용하기 때문에 `npm start`, `npm test`처럼 `run`을 생략할 수 있습니다.

## 4.3 npx

- Node Package eXecute
- npm 5.2.0 버전 이후 npm 설치 시 함께 설치됩니다.
- 패키지 설치 없이 npm 레지스트리에서 원하는 패키지를 실행할 수 있는 도구입니다.
- npm package runner이므로 아직 설치되지 않은 패키지를 자동으로 설치합니다.
- 필요한 패키지가 로컬이나 글로벌에 설치되어 있으면 그것을 실행하고, 없으면 인터넷의 npm 레지스트리에서 최신 버전을 임시로 다운로드 받아 실행 후 자동 삭제합니다.
- 일회성 실행(temporary execution)이 가능해, 예를 들어 `npx create-react-app` 처럼 프로젝트 생성 명령을 설치 없이 바로 실행할 때 유용합니다.
- 정리하면 npm은 패키지를 설치하고 관리하는 도구이고, npx는 패키지 설치 없이 패키지를 실행할 때 사용하는 도구입니다.