

# NODE JS와 EXPRESS JS

## Node.js설치

프로젝트의 규모가 커짐에 따라 React, Angular같은 프레임워크 또는 라이브러리를 도입하거나 Babel, Webpack, ESLint 등 여러가지 도구를 사용할 필요가 있다.

이때 Node.js와 npm이 필요하다.

- Node.js = 크롬 V8 자바스크립트 엔진으로 빌드된 **자바스크립트 런타임 환경**(브라우저 이외의 환경에서 동작시킬 수 있는 자바스크립트 실행 환경)이다.
- **npm(node package manager)**는 자바스크립트 패키지 매니저이다. node.js에서 사용할 수 있는 **모듈을 패키지화해서 모아둔 저장소 역할**과 **패키지 설치 및 관리**를 위한 **CLI(command line interface)**를 제공한다

<https://nodejs.org/ko>

Node.js®는 Chrome V8 JavaScript 엔진으로 빌드된 JavaScript 런타임입니다.

Security releases now available

다운로드 - Windows (x64)

18.18.2 LTS

안정적, 신뢰도 높음

20.8.1 현재 버전

최신 기능

[다른 운영 체제](#) | [변경사항](#) | [API 문서](#)   [다른 운영 체제](#) | [변경사항](#) | [API 문서](#)

LTS 일정은 [여기서 확인하세요.](#)

- **LTS버전** = 장기적으로 **안정된 지원**이 보장된 버전(실제 개발환경에서는 이버전을 설치하면 됨)
- **Current버전** = **최신기능을 제공**하지만 **안정적이지 않을 수 있다.**(학습을 위해서는 이버전을 설치)



설치가 완료되면 **터미널(명령 프롬프트)**에서 버전출력(**node -v, npm -v**)



node명령어로 자바스크립트 코드를 실행해 볼 수 있다.(ctrl+c = 종료)

## boiler-plate 디렉터리 생성

컴퓨터 프로그래밍에서 보일러플레이트는 최소한의 변경으로 여러곳에서 재사용되며, 반복적으로 비슷한 형태를 띄는 코드를 말한다.

### package.json 생성

프로젝트 디렉토리에서 아래 명령을 실행해야 한다.

```
npm init -yes
```

boiler-plate 프로젝트 폴더에 package.json 파일이 생성된다.

```
1 {
2   "name": "ex",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "test": "echo \"Error: no test specified\" && exit 1"
7   },
8   "keywords": [],
9   "author": "",
10  "license": "ISC",
11  "description": ""
12 }
```

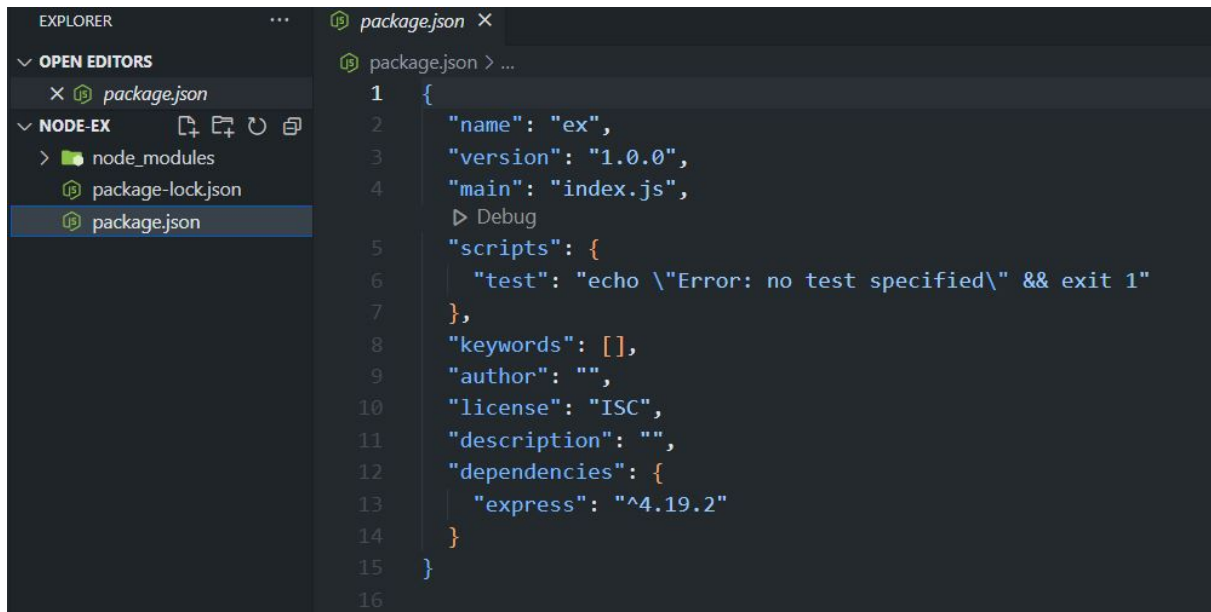
## express 라이브러리 설치

- 웹서버를 쉽게 구현할 수 있는 외부 모듈

- express 모듈은 http 모듈에 여러 기능을 추가해서 쉽게 사용할 수 있게 만든 모듈이다.

```
npm install express --save
```

설치가 완료되면 package.json에 dependencies로 express가 설치된 것을 확인할 수 있다.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows the file structure with 'package.json' selected under the 'NODE-EX' folder. The main editor window displays the contents of 'package.json'. The file contains a JSON object with the following properties: 'name' (ex), 'version' (1.0.0), 'main' (index.js), 'scripts' (test: echo \"Error: no test specified\" && exit 1), 'keywords' (empty array), 'author' (empty string), 'license' (ISC), 'description' (empty string), and 'dependencies' (express: ^4.19.2).

```
1 {
2   "name": "ex",
3   "version": "1.0.0",
4   "main": "index.js",
5   "scripts": {
6     "test": "echo \"Error: no test specified\" && exit 1"
7   },
8   "keywords": [],
9   "author": "",
10  "license": "ISC",
11  "description": "",
12  "dependencies": {
13    "express": "^4.19.2"
14  }
15 }
```

## 기본적인 express js 앱 만들기

<https://expressjs.com/en/starter/hello-world.html> ← 공식문서

아래 코드를 index.js를 생성하여 복사 붙여넣기 합니다.

## Hello world example

Embedded below is essentially the simplest Express app you can create. It is a single file app — *not* what you'd get if you use the [Express generator](#), which creates the scaffolding for a full app with numerous JavaScript files, Jade templates, and sub-directories for various purposes.

```
1 const express = require('express' 4.18.2 )
2 const app = express()
3 const port = 3000
4
5 app.get('/', (req, res) => {
6   res.send('Hello World!')
7 })
8
9 app.listen(port, () => {
10   console.log(`Example app listening on port ${port}`)
11 })
```

Save on RunKit

Node 10 ↕

help

URL: <https://b2qax0rpm9um.runkit.sh>

This app starts a server and listens on port 3000 for connections. The app responds with "Hello World!" for requests to the root URI (/) or **route**. For every other path, it will respond with a **404 Not Found**.

## index.js

```
const express = require('express') //express 모듈을 가져오는
구문
const app = express() //새로운 express 앱을 만들
const port = 5000 //port 경로 (port는 자유롭게 변경해도 됨)

//express앱에 get메서드를 연결해서 서버에 요청 ('/' = root 디렉터
리)
app.get('/', (req, res) => {
  res.send('Hello World!') //'Hello World!'를 출력
})

//listen으로 서버를 실행
app.listen(port, () => {
  console.log(`Example app listening on port ${port}`)
})
```

## package.json

```
{
  "name": "ex",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "start": "node index.js",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": "",
  "dependencies": {
    "express": "^4.19.2"
  }
}
```

package.json에 scripts 부분에 "start": "node index.js" 을 추가

```
# 노드 앱을 실행
npm run start
```

