IT 기술실무
Node.js 6주차

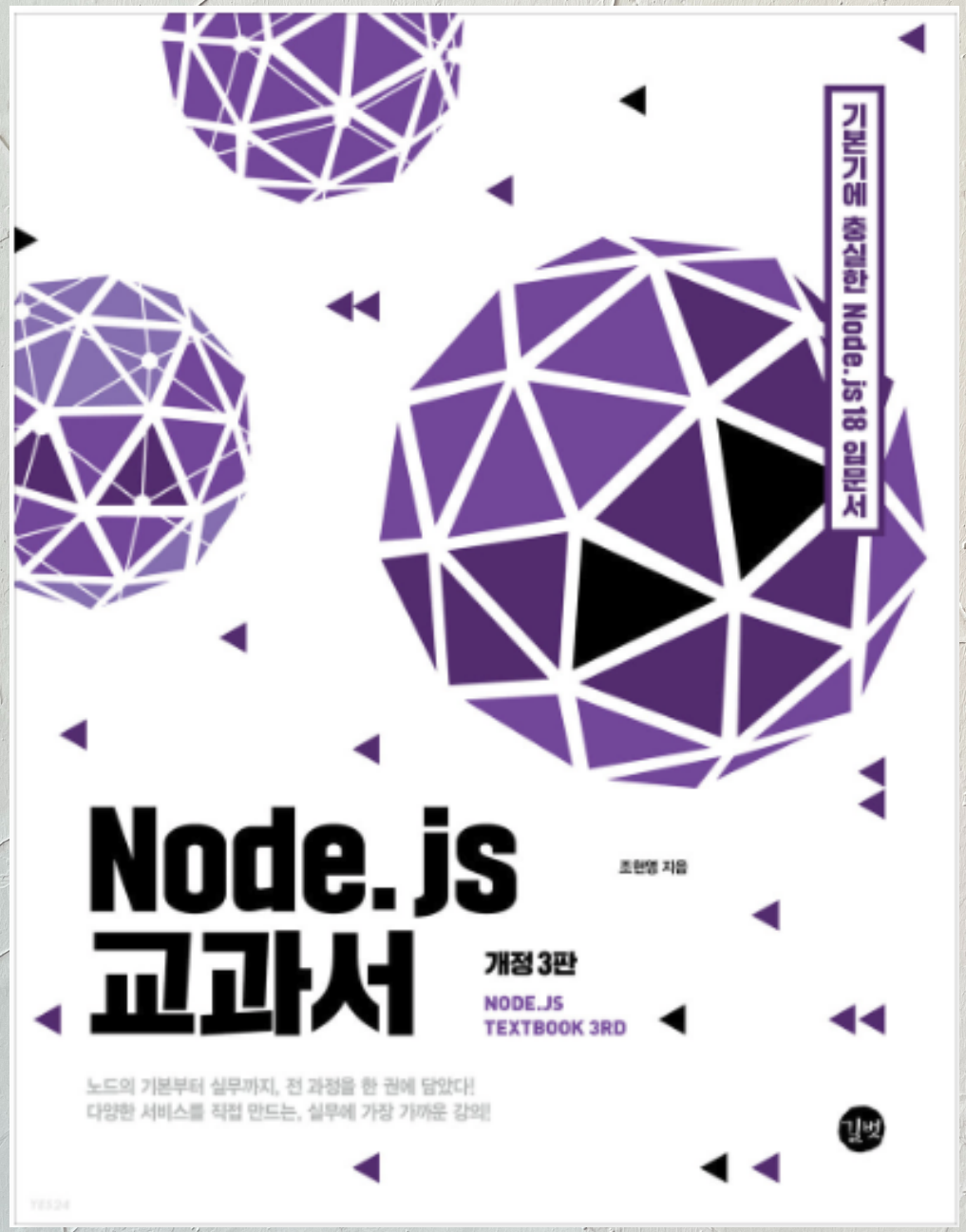
목표 : Package Manager에 대하여 이해할 수 있다.

강의개요

주	주차별 목표	강의소개	교재
1	강의소개 등	강의소개 및 필수사항 공유	주/보조
2	Abstract	핵심 개념, 서버로서의 노드, 서버 외의 노드, 개발 환경 구축(23~64)	주교재
3	ES2015+ : ECMAScript 6	ES2015+ 및 프런트엔드 자바스크립트(65~92)	주교재
4	Node's Functions	REPL, JS 파일 실행하기, 모듈로 만들기, 노드 내장 객체/모듈 등(93~178)	주교재
5	http Module	요청/응답, REST와 라우팅, 쿠키/세션, https/http2, cluster(179~216)	주교재
6	Package Manager	npm, package.json, 패키지 배포하기(217~240)	주교재
7	Express Web Server	미들웨어, req/res, 템플릿 엔진 사용하기(241~290)	주교재
8	중간시험	중간시험 및 웹어플리케이션 추가 강의	주/보조
9	Databases : MySQL	MySQL 설치, 데이터베이스/테이블, CRUD, 시퀄라이즈(291~364)	주교재
10	Databases : MongoDB	몽고디비 설치, 데이터베이스/컬렉션, CRUD, 몽구스(365~416)	주교재
11	Web API Server	API 서버, JWT 토큰 인증하기, 다른 서비스 호출하기, SNS API, CORS 등(475~520)	주교재
12	Node Service Test	테스트 준비하기, 유닛/커버리지/통합/부하(521~560)	주교재
13	Web Socket	웹 소켓 이해하기, ws 모듈, Socket.io, 미들웨어와 소켓, 채팅 구현하기(561~608)	주교재
14	CLI	간단한 콘솔 명령어, Commander, Inquirer 사용하기(649~676)	주교재
15	기말시험	기말시험 및 웹개발자 로드맵 추가 강의	주/보조

목차

table of contents



5.1 npm 알아보기

1. npm이란

Node Package Manager

- 노드의 패키지 매니저
- 다른 사람들이 만든 소스 코드들을 모아둔 저장소
- 남의 코드를 사용하여 프로그래밍 가능
- 이미 있는 기능을 다시 구현할 필요가 없어 효율적
- 오픈 소스 생태계를 구성중

- 패키지: npm에 업로드된 노드 모듈
- 모듈이 다른 모듈을 사용할 수 있듯 패키지도 다른 패키지를 사용할 수 있음
- 의존 관계라고 부름

5.2 package.json으로 패키지 관리하기

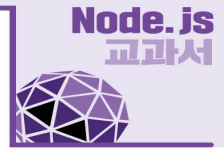
1. package.json

- 현재 프로젝트에 대한 정보와 사용 중인 패키지에 대한 정보를 담은 파일
  - 같은 패키지라도 버전별로 기능이 다를 수 있으므로 버전을 기록해두어야 함
  - 동일한 버전을 설치하지 않으면 문제가 생길 수 있음
  - 노드 프로젝트 시작 전 package.json부터 만들고 시작함(npm init)

콘솔

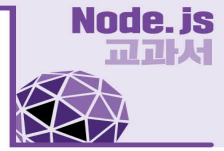
```
$ npm init
This utility will walk you through creating a package.json file.
It only covers the most common items, and tries to guess sensible defaults.

See `npm help json` for definitive documentation on these fields
and exactly what they do.

Use `npm install <pkg>` afterwards to install a package and
save it as a dependency in the package.json file.

Press ^C at any time to quit.
package name: (폴더명) [프로젝트 이름 입력]
version: (1.0.0) [프로젝트 버전 입력]
description: [프로젝트 설명 입력]
entry point: index.js
test command: [엔터 키 클릭]
git repository: [엔터 키 클릭]
keywords: [엔터 키 클릭]
author: [여러분의 이름 입력]
license: (ISC) [엔터 키 클릭]
About to write to C:\Users\zerocho\npmtest\package.json:

{
  "name": "npmtest",
  "version": "0.0.1",
  "description": "hello package.json",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "ZeroCho",
  "license": "ISC"
}

Is this ok? (yes) yes
```

2. package.json 속성들



- package name: 패키지의 이름입니다. package.json의 name 속성에 저장됩니다.
- version: 패키지의 버전입니다. npm의 버전은 다소 엄격하게 관리됩니다. 5.3절에서 다룹니다.
- entry point: 자바스크립트 실행 파일 진입점입니다. 보통 마지막으로 module.exports를 하는 파일을 지정합니다. package.json의 main 속성에 저장됩니다.
- test command: 코드를 테스트할 때 입력할 명령어를 의미합니다. package.json scripts 속성 안의 test 속성에 저장됩니다.
- git repository: 코드를 저장해둔 Git 저장소 주소를 의미합니다. 나중에 소스에 문제가 생겼을 때 사용자들이 이 저장소에 방문해 문제를 제기할 수도 있고, 코드 수정본을 올릴 수도 있습니다. package.json의 repository 속성에 저장됩니다.
- keywords: 키워드는 npm 공식 홈페이지(https://npmjs.com)에서 패키지를 쉽게 찾을 수 있게 해줍니다. package.json의 keywords 속성에 저장됩니다.
- license: 해당 패키지의 라이선스를 넣어주면 됩니다.

3. npm 스크립트



- npm init이 완료되면 폴더에 package.json이 생성됨

package.json

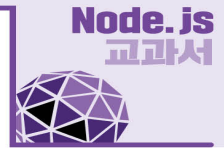
```
{
  "name": "npmtest",
  "version": "0.0.1",
  "description": "hello package.json",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "author": "ZeroCho",
  "license": "ISC"
}
```

- npm run [스크립트명]으로 스크립트 실행

콘솔

```
$ npm run test
> npmtest@0.0.1 test C:\Users\zerocho\npmtest
> echo "Error: no test specified" && exit 1

"Error: no test specified"
npm ERR! Test failed.  See above for more details.
...
```

4. 패키지 설치하기



express 설치하기

콘솔

```
$ npm install express
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN npmtest@0.0.1 No repository field.

+ express@4.17.1
added 285 packages from 163 contributors and audited 2401 packages in 6.314s
```

package.json에 기록됨(dependencies에 express 이름과 버전 추가됨)

package.json

```
{
  "name": "npmtest",
  ...
  "license": "ISC",
  "dependencies": {
    "express": "^4.17.1",
```

5. node_modules

- npm install 시 node_modules 폴더 생성
  - 내부에 설치한 패키지들이 들어 있음
  - express 외에도 express와 의존 관계가 있는 패키지들이 모두 설치됨
- package-lock.json도 생성되어 패키지 간 의존 관계를 명확하게 표시함

그림 5-3 node_modules 폴더 안의 패키지

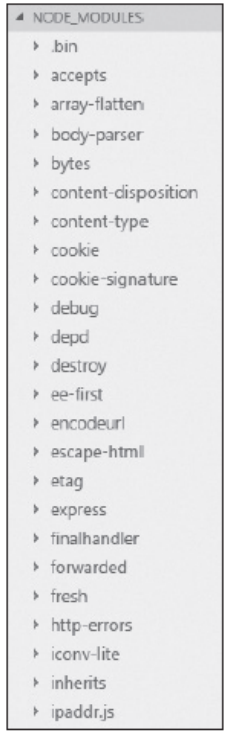
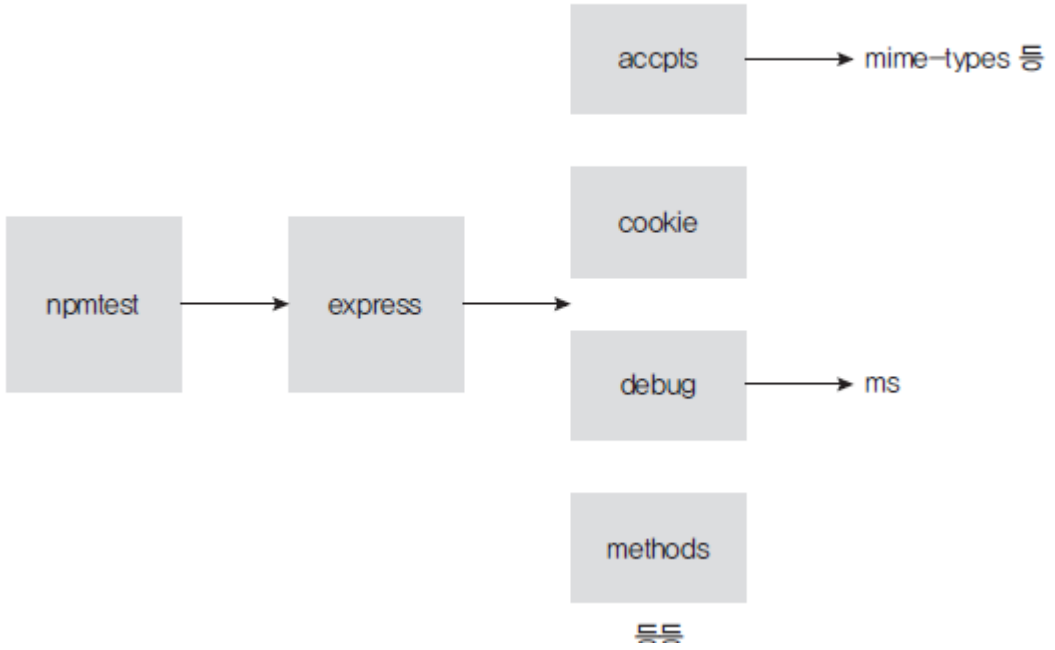


그림 5-4 의존 관계



6. 여러 패키지 동시 설치하기

» npm install 패키지1 패키지2 패키지3 ...

콘솔

```
$ npm install morgan cookie-parser express-session
npm WARN npmtest@0.0.1 No repository field.

+ morgan@1.10.0
+ express-session@1.17.1
+ cookie-parser@1.4.5
added 8 packages from 5 contributors in 3.115s
```

package.json

```
{
  "name": "npmtest",
  ...
  "dependencies": {
    "cookie-parser": "^1.4.5",
    "express": "^4.17.1",
    "express-session": "^1.17.1",
    "morgan": "^1.10.0"
  }
}
```

7. 개발용 패키지

» npm install --save-dev 패키지명 또는 npm i -D 패키지명

- devDependencies에 추가됨

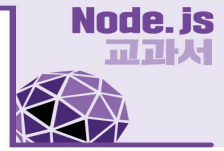
콘솔

```
$ npm install --save-dev nodemon
+ nodemon@1.17.3
added 227 packages from 134 contributors in 24.735s
```

package.json

```
{
  ...
  "devDependencies": {
    "nodemon": "^2.0.3"
  }
}
```

8. Peer Dependencies



- A 라이브러리의 peerDependencies가 다음과 같은 경우
  - A 라이브러리는 jQuery@3이 설치되었다고 간주하고 코딩한 것

package.json

```
{
  ...
  "peerDependencies": {
    "jQuery": "^3.0.0"
  }
}
```

- ERESOLVE unable to resolve dependency tree
  - npm i --force로 모든 버전 설치하기
  - npm i --legacy-peer-deps로 peerDependencies 무시하기

9. npm ci

- npm i를 할 때마다 package.json과 package-lock.json이 변할 수 있음
  - 배포 시에는 npm ci로 배포하기
- node_modules는 git 같은 버전 관리 시스템에 커밋할 필요가 없음
  - npm i나 npm ci를 하면 동일하게 복구됨

10. 글로벌(전역) 패키지

- npm install --global 패키지명 또는 npm i -g 패키지명
  - 모든 프로젝트와 콘솔에서 패키지를 사용할 수 있음
  - 예제는 rm –rf(리눅스의 삭제 명령)를 흉내내는 rimraf 패키지의 글로벌 설치
  - npx로 글로벌 설치 없이 글로벌 명령어 사용 가능

콘솔

```
$ npm install --global rimraf
C:\Users\zerocho\AppData\Roaming\npm\rimraf -> C:\Users\zerocho\AppData\Roaming\npm\
node_modules\rimraf\bin.js

+ rimraf@3.0.2
added 12 packages from 4 contributors in 1.015s
```

콘솔

```
$ rimraf node_modules
```

콘솔

```
$ npm install --save-dev rimraf
$ npx rimraf node_modules
```

5.3 패키지 버전 이해하기

1. SemVer 버저닝

» 노드 패키지의 버전은 SemVer(유의적 버저닝) 방식을 따름

- Major(주 버전), Minor(부 버전), Patch(수 버전)

- 노드에서는 배포를 할 때 항상 버전을 올려야 함
- Major는 하위 버전과 호환되지 않은 수정 사항이 생겼을 때 올림
- Minor는 하위 버전과 호환되는 수정 사항이 생겼을 때 올림
- Patch는 기능에 버그를 해결했을 때 올림

그림 5-5 SemVer

2. 버전 기호 사용하기

» 버전 앞에 기호를 붙여 의미를 더함

- ^1.1.1: 패키지 업데이트 시 minor 버전까지만 업데이트 됨(2.0.0버전은 안 됨)
- ~1.1.1: 패키지 업데이트 시 patch버전까지만 업데이트 됨(1.2.0버전은 안 됨)
- >=, <=, >, <는 이상, 이하, 초과, 미만.
- @latest는 최신 버전을 설치하라는 의미
- 실험적인 버전이 존재한다면 @next로 실험적인 버전 설치 가능(불안정함)
- 각 버전마다 부가적으로 알파/베타/RC 버전이 존재할 수도 있음(1.1.1-alpha.0, 2.0.0-beta.1, 2.0.0-rc.0)

5.4 기타 npm 명령어

1. 기타 명령어

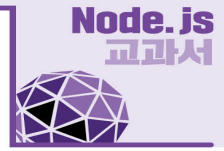
- **npm outdated**: 어떤 패키지에 기능 변화가 생겼는지 알 수 있음

그림 5-6 npm outdated

Package	Current	Wanted	Latest	Location
nodemon	1.19.4	1.19.4	2.0.1	npmtest
rimraf	2.6.3	2.7.1	3.0.0	npmtest

- **npm update**: package.json에 따라 패키지 업데이트
- **npm uninstall 패키지명**: 패키지 삭제(npm rm 패키지명으로도 가능)
- **npm search 검색어**: npm 패키지를 검색할 수 있음(npmjs.com에서도 가능)
- **npm info 패키지명**: 패키지의 세부 정보 파악 가능
- **npm login**: npm에 로그인을 하기 위한 명령어(npmjs.com에서 회원가입 필요)
- **npm whoami**: 현재 사용자가 누구인지 알려줌
- **npm logout**: 로그인한 계정을 로그아웃

2. 기타 명령어



- **npm version 버전**: package.json의 버전을 올림(Git에 커밋도 함)

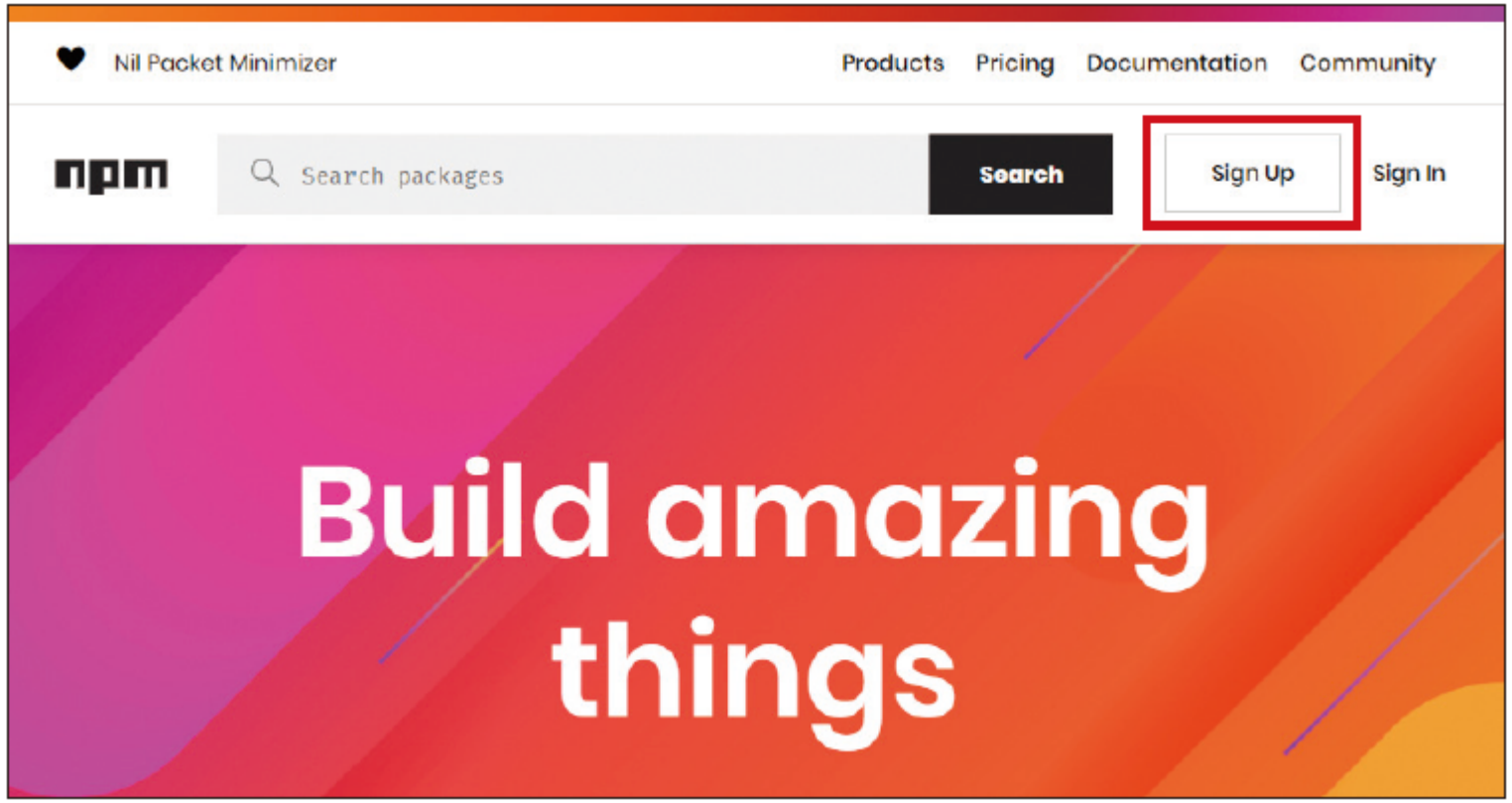
```
npm version 5.3.2, npm version minor
```

- **npm deprecate [패키지명][버전] [메시지]**: 패키지를 설치할 때 경고 메시지를 띄우게 함(오류가 있는 패키지에 적용)
- **npm publish**: 자신이 만든 패키지를 배포
- **npm unpublish --force**: 자신이 만든 패키지를 배포 중단(배포 후 24시간 내에만 가능)
  - 다른 사람이 내 패키지를 사용하고 있는데 배포가 중단되면 문제가 생기기 때문
- 기타 명령어는 https://docs.npmjs.com의 CLI Commands에서 확인

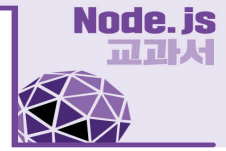
5.5 패키지 배포하기

1. npm 회원가입

» npmjs.com에 접속해서 회원가입



2. 배포할 패키지 작성



- package.json과 main 부분과 배포할 파일 경로명이 일치해야 함
  - "main": "index.js"

index.js

```
module.exports = () => {
  return 'hello package';
};
```

3. 배포 시도하기

npm publish 입력

콘솔

```
$ npm publish
npm notice
// notice 생략
npm ERR! code E403
npm ERR! 403 403 Forbidden - PUT https://registry.npmjs.org/npmtest - You do not have
permission to publish "npmtest". Are you logged in as the correct user?
npm ERR! 403 In most cases, you or one of your dependencies are requesting
npm ERR! 403 a package version that is forbidden by your security policy.

npm ERR! A complete log of this run can be found in:
npm ERR!     C:\Users\speak\AppData\Roaming\npm-cache\_logs\2020-04-
24T05_52_25_852Z-debug.log
```

npmtest란 이름을 누가 사용중

- 이름이 겹치면 안 되므로 다른 것으로 바꿔서 배포

4. 배포 시도하기

» 이름을 변경한 후 npm publish 입력

콘솔

```
$ npm publish
// notice 생략
+ npmtest-1234@0.0.1
$ npm info npmtest-1234
npmtest-1234@0.0.1 | ISC | deps: none | versions: 1
hello package.json
// 중략
maintainers:
- zerocho <zerohch0@gmail.com>

dist-tags:
 latest: 0.0.1

 published 51 seconds ago by zerocho <zerohch0@gmail.com>
```

5. 버전 올리기

- 버전을 올리지 않으면 E403 에러
- npm version 명령어로 버전 올리기

콘솔

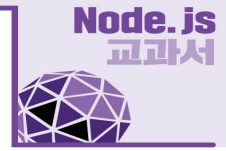
```
$ npm publish
// notice 생략
npm ERR! code E403
npm ERR! 403 403 Forbidden - PUT https://registry.npmjs.org/npmtest-1234 - You cannot
publish over the previously published versions: 0.0.1.
```

이 에러 메시지가 보인다면 이미 출시한 버전이라는 뜻입니다. 따라서 이보다 더 높은 버전을 출시해야 합니다. 버전을 올리기 위해 npm version 명령어를 사용합니다.

콘솔

```
$ npm version patch
v0.0.2
$ npm publish
// notice 생략
+ npmtest-1234@0.0.2
```

6. 배포 취소하기



» 24시간 내에 npm unpublish 패키지명 --force입력

콘솔

```
$ npm unpublish npmtest-1234 --force
npm WARN using --force I sure hope you know what you are doing.
- npmtest-1234
$ npm info npmtest-1234
npm ERR! code E404
npm ERR! 404 Unpublished by zerocho on 2020-04-28T08:51:10.506Z
...
```

감사합니다.

단순하게 설명할 수 없다면
제대로 이해하지 못한 것이다.

아인슈타인