

프로젝트 기반 실무형 서비스

Chap03. Nodejs 콜백과 비동기

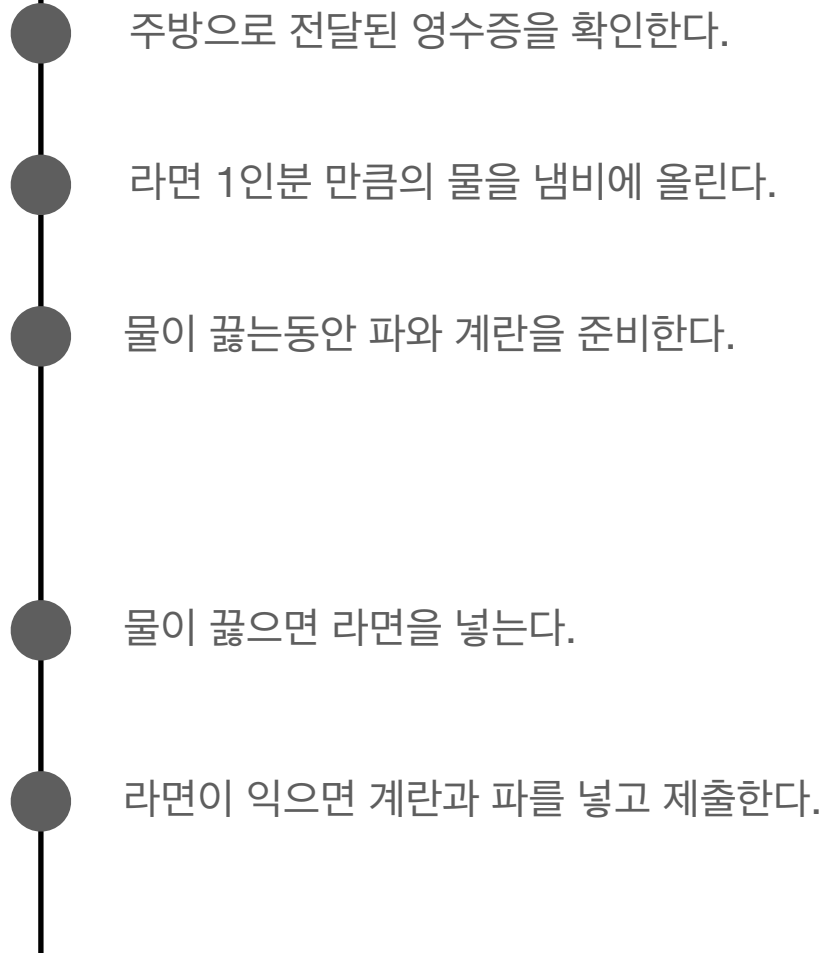
김진형

동기 / 비동기 프로그래밍

동기식 프로그래밍 예제

단순한 라면집 주방장

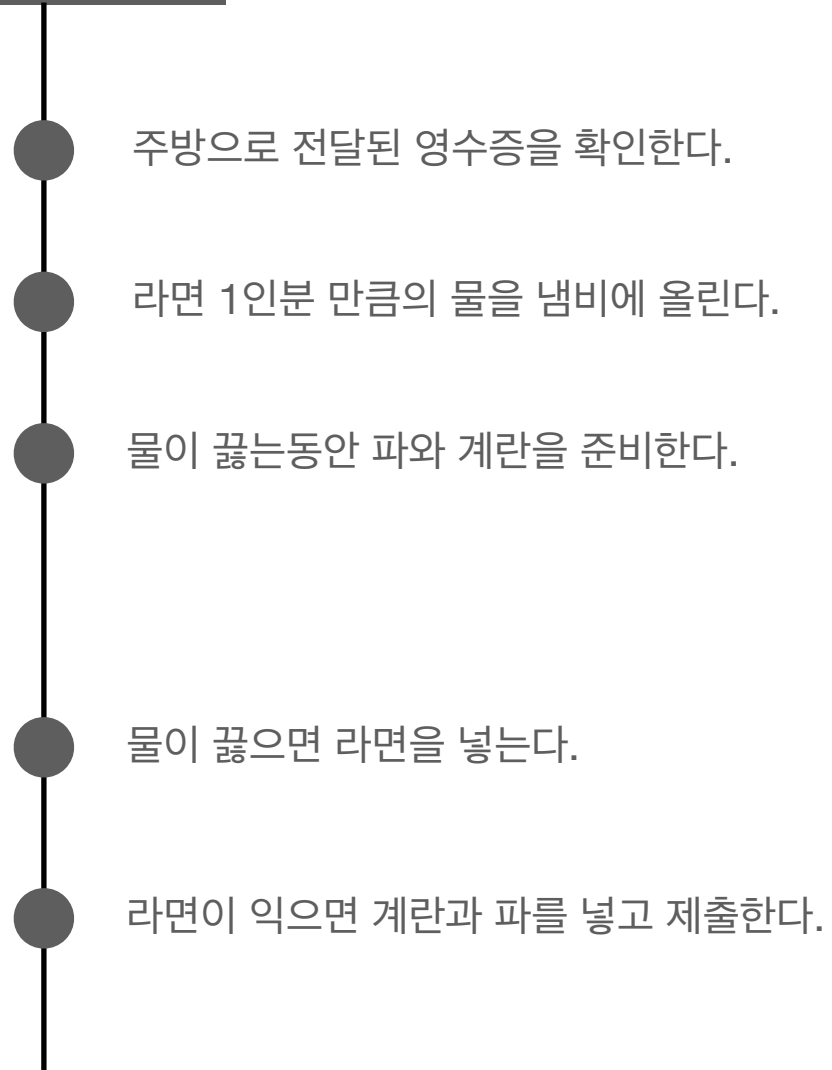
주문



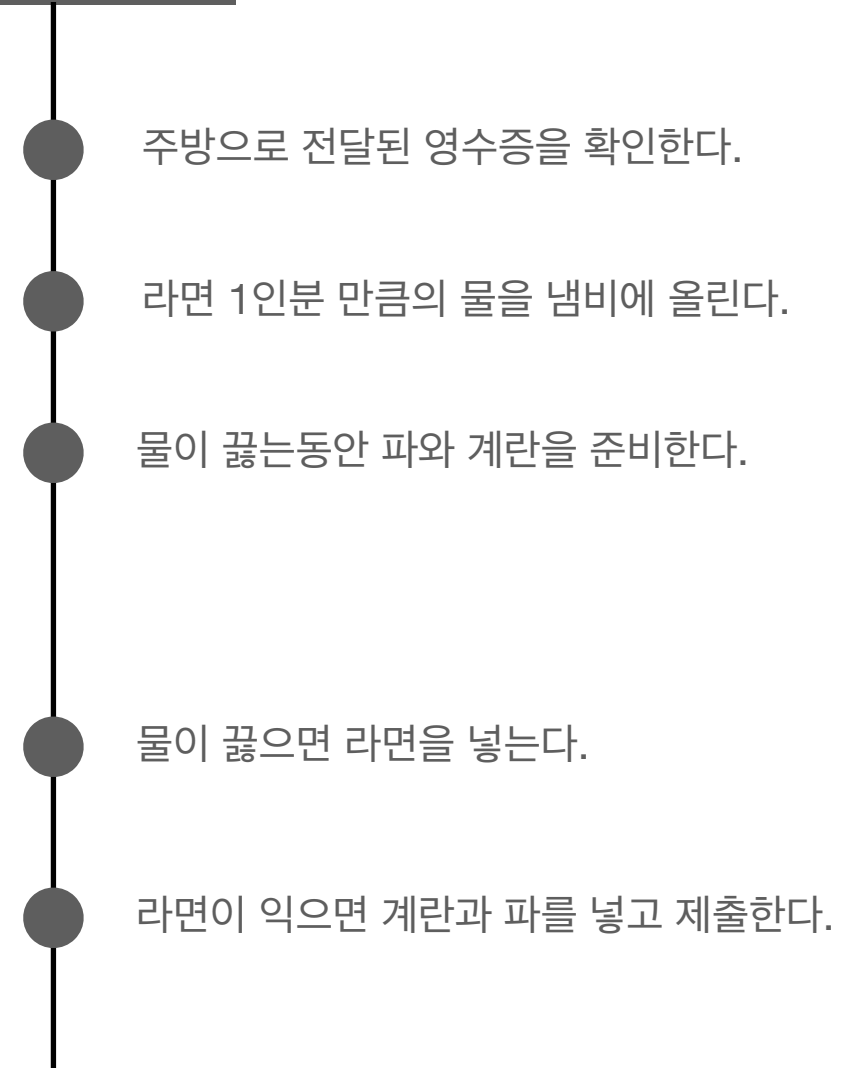
동기식 프로그래밍 예제

단순한 라면집 주방장

주문



주문



동기식 프로그래밍 예제

동기식 + 블로킹 프로그램 예제

```
let noodle = () => {  
  for(let i = 1; i <= 100; i++) {  
    console.log(`라면 ${i}개째...`);  
  }  
}
```

```
console.log("가게오픈");  
noodle2();  
console.log("가게닫기");
```

비동기식 프로그래밍 예제

멀티플레이어 라면집 주방장

주문

- 주방으로 전달된 영수증을 확인한다.
- 라면 1인분 만큼의 물을 냄비에 올린다.
- 물이 끓는동안 파와 계란을 준비한다.
- ● 라면 1인분 만큼의 물을 냄비에 올린다.
- ● 물이 끓는동안 파와 계란을 준비한다.
- 물이 끓으면 라면을 넣는다.
- ● 라면이 익으면 계란과 파를 넣고 제출한다.
● 물이 끓으면 라면을 넣는다.
- ● 라면이 익으면 계란과 파를 넣고 제출한다.

비동기식 코드 예제

비동기 + 논블로킹 예제

```
const noodle = (message, time) => {  
  setTimeout(() => {  
    console.log(message);  
  }, time);  
};
```

```
noodle('라면1', 1000);  
noodle('라면2', 500);  
noodle('라면3', 200);
```

Callback

콜백 함수 예제

라면을 다 끓이고 서빙하는 예제

```
function noodle(message) {  
  console.log(`${message} 라면 끓이는중..`);  
  return `${message}라면`  
}  
function serve(number) {  
  console.log(`${number} 테이블에 서빙`);  
}  
  
serve(noodle("살안찌는"));
```

콜백 함수 예제

콜백 함수로 변경

```
function noodle(message, callback) {  
  console.log(`${message} 라면 끓이는중..`);  
  callback(`${message}라면`)  
}  
function serve(number) {  
  console.log(`${number} 테이블에 서빙`);  
}  
  
noodle("살안찌는", serve)
```

Callback 함수

특성

- 어떤 작업을 다른 객체에게 맡기고, 그 일이 끝나기를 기다리지 않고 알아서 그 일이 끝나면 수행해야 할 함수를 정의할 수 있다.
- 논블로킹 이며 비동기 방식 함수이다.
- Callback 지옥에 빠질 수 있으므로 주의해야한다.
- 코드의 가독성이 떨어진다.

Callback 일반적인 예제

3초라면 호출 예제

```
function order(menu, time, callback) {  
  console.log(menu + "주문대기");  
  callback(time);  
}
```

```
function cook(time, callback) {  
  setTimeout(() => {  
    callback();  
  }, time);  
}
```

```
order("라면", 3000, (time) => {  
  cook(time, () => {  
    console.log('조리완료')  
  })  
})
```

빈번한일...

```
function order(menu, time, callback) {  
  console.log(menu + "주문대기");  
  callback(time);  
}
```

```
function cook(time, callback) {  
  setTimeout(() => {  
    callback();  
  }, time);  
}
```

```
function serving(callback) {  
  setTimeout(() => {  
    callback();  
  }, 300);  
}
```

```
order("라면", 3000, (time) => {  
  cook(time, () => {  
    console.log('조리완료')  
    serving(() => {  
      console.log('식사...')  
    });  
  });  
})  
})
```

공공 API를 호출하는 예제

현재 상영중인 영화의 등장에는 배우의 소속사의 전화번호를 조회하는 api

```
getApi("http://movie.kr/상영중영화", function(result) {  
    getApi("http://movie.kr/영화조회", function(result) {  
        getApi("http://movie.kr/배우조회", function(result) {  
            getApi("http://movie.kr/소속사조회", function(result) {  
                getApi("http://movie.kr/전화번호조회", function(result) {  
                    // 코드...  
                })  
            })  
        })  
    })  
})
```

Promise

Promise

개요

- ES6 부터 등장한 흐름제어 패턴
- 프로그램의 가독성을 높여주고, 에러 처리를 원활히 수행

```
new Promise((resolve, reject) => {  
    // code  
});
```


callback 과 promise

```
let loading = (path, done) => {  
  console.log("경로 : " + path);  
  done(path + "text.png");  
}
```

```
loading("/Users/home/", (path) => {  
  console.log("완료 : " + path);  
}))
```

```
let loading = (path, done) => {  
  return new Promise((resolve, reject) => {  
    console.log("경로 : " + path);  
    resolve(path + "text.png");  
  })  
}
```

```
loading("/Users/home/")  
  .then((path) => {  
    console.log("완료 : " + path);  
  }).catch((err) => {  
    console.log("error : " + error);  
  })
```

공공 API를 호출하는 예제

현재 상영중인 영화의 등장에는 배우의 소속사의 전화번호를 조회하는 api

```
getApi("http://movie.kr/상영중영화")
  .then((result) => {
    return getApi("http://movive.kr/영화조회");
  })
  .then((result) => {
    return getApi("http://movive.kr/배우조회");
  })
  .then((result) => {
    return getApi("http://movive.kr/소속사조회");
  })
  .then((result) => {
    return getApi("http://movive.kr/전화번호조회");
  })
  .then((result) => {
    // 코드...
  })
```

async / await

async / await

개요

- callback 과 promise의 가독성을 해결하기 위한 최신 문법
- 항상 async 함수 내에서 사용해야함

```
async function myFunction() {  
    let data = await secondFunction();  
}
```

promise 와 async/await

```
let loading = (path, done) => {  
  return new Promise((resolve, reject) => {  
    console.log("경로 : " + path);  
    resolve(path + "text.png");  
  })  
}
```

```
loading("/Users/home/")  
  .then((path) => {  
    console.log("완료 : " + path);  
  }).catch((err) => {  
    console.log("error : " + error);  
  })
```

```
let loading = async (path, done) => {  
  console.log("경로 : " + path);  
  return path + "text.png";  
}
```

```
let path = await loading("/Users/home/");  
console.log("완료 : " + path);
```

공공 API를 호출하는 예제

현재 상영중인 영화의 등장에는 배우의 소속사의 전화번호를 조회하는 api

```
let result = await getApi("http://movie.kr/상영중영화");  
result = await getApi("http://movie.kr/영화조회");  
result = await getApi("http://movie.kr/배우조회");  
result = await getApi("http://movie.kr/소속사조회");  
result = await getApi("http://movie.kr/전화번호조회");
```

module

nodejs module system

파일 분리

- 파일 하나에 모든 함수들을 넣으면 가독성이 떨어지고, 유지관리가 어려움
- javascript에서는 module 방식으로 파일을 분리해서 사용할 수 있음
- 소스들을 높은 응집도와 낮은 결합도를 유지하기 위하여 분리해야 유지관리가 쉬움

단일 file에 모든 함수들이 총 집합

```
async function sendNoti(message, userId) {  
    // userId 에게 알림 전송  
}
```

```
async function saveFile(image) {  
    // image에 온 이미지를 저장  
}
```

```
async function storeDatabase(message) {  
    // message 데이터를 DB에 저장  
}
```

```
async function createFeed(message, image) {  
    // 1. 올바른 데이터가 왔는지 검증  
    // 2. 이미지 저장  
    await saveFile(image);  
    // 3. 데이터베이스에 기록  
    await storeDatabase(message);  
    // 4. 멘션한 유저에게 새글 알림  
    await sendNoti(message, userId);  
}
```

module 분리

module을 이용한 단일 함수 분리

main.js

```
const sendNoti = require('./noti');
const saveFile = require('./file');

function storeDatabase(message) {
  // message 데이터를 DB에 저장
}

async function createFeed(message, image) {
  // 1. 올바른 데이터가 왔는지 검증
  // 2. 이미지 저장
  await saveFile(image);
  // 3. 데이터베이스에 기록
  await storeDatabase(message);
  // 4. 멘션한 유저에게 새글 알림
  await sendNoti(message, userId);
}
```

noti.js

```
function sendNoti(message, userId) {
  // userId 에게 알림 전송
}

module.exports = sendNoti
```

file.js

```
function saveFile(image) {
  // image에 온 이미지를 저장
}

module.exports = saveFile
```

module 분리

module을 이용한 여러 함수 분리

main.js

```
const sendNoti = require('./test');
const saveFile = require('./file');
const db = require('./db');

async function createFeed(message, image) {
  // 1. 올바른 데이터가 왔는지 검증
  // 2. 이미지 저장
  await saveFile(image);
  // 3. 데이터베이스에 기록
  await db.storeDatabase(message);
  // 4. 멘션한 유저에게 새글 알림
  await sendNoti(message, userId);
}
```

db.js

```
function storeDatabase(message) {
  // message 데이터를 DB에 저장
}

function deleteDatabase(id) {
  // 해당하는 id의 피드를 삭제
}

function updateDatabase(id, msg) {
  // 해당 id의 피드를 message로 업데이트
}

module.exports = {
  storeDatabase: storeDatabase,
  delete: deleteDatabase,
  update: updateDatabase
}
```

module 분리

module을 이용한 여러 함수 분리2

main.js

```
const sendNoti = require('./test');
const saveFile = require('./file');
const { storeDatabase } = require('./db');

async function createFeed(message, image) {
  // 1. 올바른 데이터가 왔는지 검증
  // 2. 이미지 저장
  await saveFile(image);
  // 3. 데이터베이스에 기록
  await storeDatabase(message);
  // 4. 멘션한 유저에게 새글 알림
  await sendNoti(message, userId);
}
```

db.js

```
function storeDatabase(message) {
  // message 데이터를 DB에 저장
}

function deleteDatabase(id) {
  // 해당하는 id의 피드를 삭제
}

function updateDatabase(id, msg) {
  // 해당 id의 피드를 message로 업데이트
}

module.exports = {
  storeDatabase: storeDatabase,
  delete: deleteDatabase,
  update: updateDatabase
}
```

module 분리

module을 이용한 여러 함수 분리3

main.js

```
const sendNoti = require('./test');
const saveFile = require('./file');
const { storeDatabase } = require('./db');

async function createFeed(message, image) {
  // 1. 올바른 데이터가 왔는지 검증
  // 2. 이미지 저장
  await saveFile(image);
  // 3. 데이터베이스에 기록
  await storeDatabase(message);
  // 4. 멘션한 유저에게 새글 알림
  await sendNoti(message, userId);
}
```

db.js

```
exports.storeDatabase = (message) => {
  // message 데이터를 DB에 저장
}

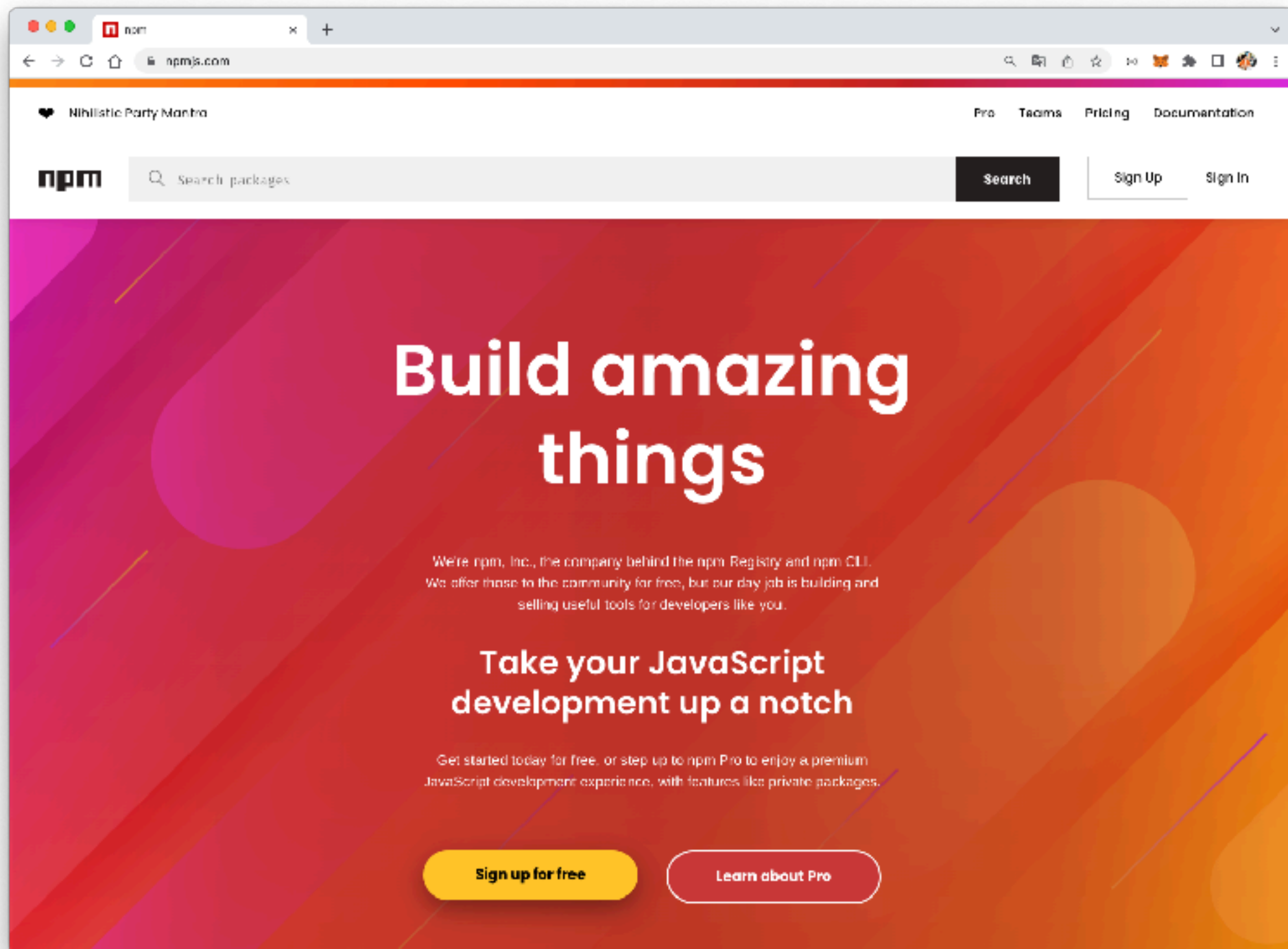
exports.deleteDatabase = (id) => {
  // 해당하는 id의 피드를 삭제
}

exports.updateDatabase = (id, msg) => {
  // 해당 id의 피드를 message로 업데이트
}
```

Node Package Manager

npm

Don't reinvent the wheel



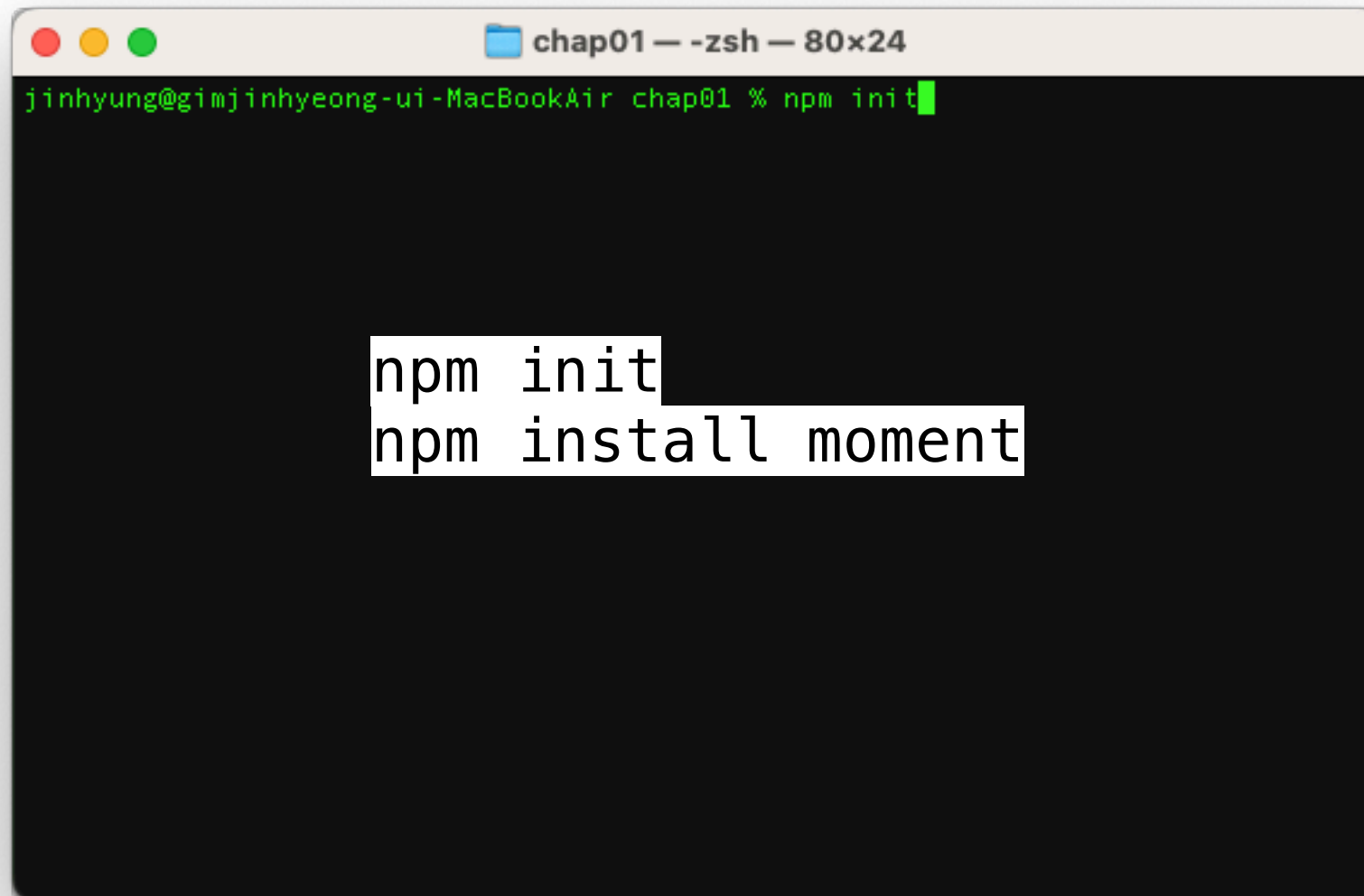
npm

package.json 기본 구조

```
{
  "name": "example-project",
  "version": "1.0.0",
  "description": "tukroea example",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "keywords": ["tukorea", "nodejs", "flutter"],
  "author": "jinhyung",
  "license": "ISC",
  "dependencies": {
    "axios": "^1.1.3",
    "dotenv": "^16.0.3",
    "moment": "^2.29.4",
    "mysql2": "^2.3.3"
  },
  "devDependencies": {
    "jest": "^29.2.1",
    "nodemon": "^2.0.20",
    "supertest": "^6.3.0"
  }
}
```


npm 활용하기

npm 초기화 및 moment 사용하기



```
chap01 — -zsh — 80x24
jinhyung@gimjinhyeong-ui-MacBookAir chap01 % npm init
npm init
npm install moment
```

A terminal window titled 'chap01 — -zsh — 80x24' showing the execution of 'npm init' and 'npm install moment' commands. The prompt is 'jinhyung@gimjinhyeong-ui-MacBookAir chap01 %'. The output of 'npm init' is 'npm init' and the output of 'npm install moment' is 'npm install moment'.

npm 활용하기

npmjs.com 내에 있는 moment 문서

The screenshot shows the npm package page for 'moment'. The browser address bar displays 'npmjs.com/package/moment'. The page header includes the npm logo, a search bar, and links for 'Pro', 'Teams', 'Pricing', and 'Documentation'. The package name 'moment' is shown with a TypeScript (TS) badge. Below it, the version '2.29.4' is listed as 'Public' and 'Published 6 months ago'. A navigation bar contains links for 'Readme', 'Code' (with a 'Beta' badge), '0 Dependencies', '61,236 Dependents', and '74 Versions'. The main content area features the 'Moment.js' logo, a series of status badges (npm, v2.29.4, downloads, license, build, coverage, license scan), and a 'SemVer compatibility' note. A description states it's a JavaScript date library. The 'Project Status' section notes it's a legacy project in maintenance mode. The 'Resources' section lists links for 'Documentation', 'Changelog', and 'Stack Overflow'. The 'License' section states it's under the MIT license. On the right sidebar, the 'Install' section shows the command 'npm i moment'. The 'Repository' is 'github.com/moment/moment'. The 'Homepage' is 'momentjs.com'. A 'Weekly Downloads' chart shows 17,368,051 downloads. A table lists 'Version' (2.29.4), 'License' (MIT), 'Unpacked Size' (4.23 MB), 'Total Files' (533), 'Issues' (194), and 'Pull Requests' (53). The 'Last publish' date is '6 months ago'.

moment **TS**
2.29.4 • Public • Published 6 months ago

Readme Code **Beta** 0 Dependencies 61,236 Dependents 74 Versions

Moment.js

npm v2.29.4 downloads 68M/months license MIT build passing coverage 88% license scan passing

SemVer compatibility

A JavaScript date library for parsing, validating, manipulating, and formatting dates.

Project Status

Moment.js is a legacy project, now in maintenance mode. In most cases, you should choose a different library.

For more details and recommendations, please see [Project Status](#) in the docs.

Thank you.

Resources

- [Documentation](#)
- [Changelog](#)
- [Stack Overflow](#)

License

Moment.js is freely distributable under the terms of the [MIT license](#).

Install

```
> npm i moment
```

Repository
github.com/moment/moment

Homepage
momentjs.com

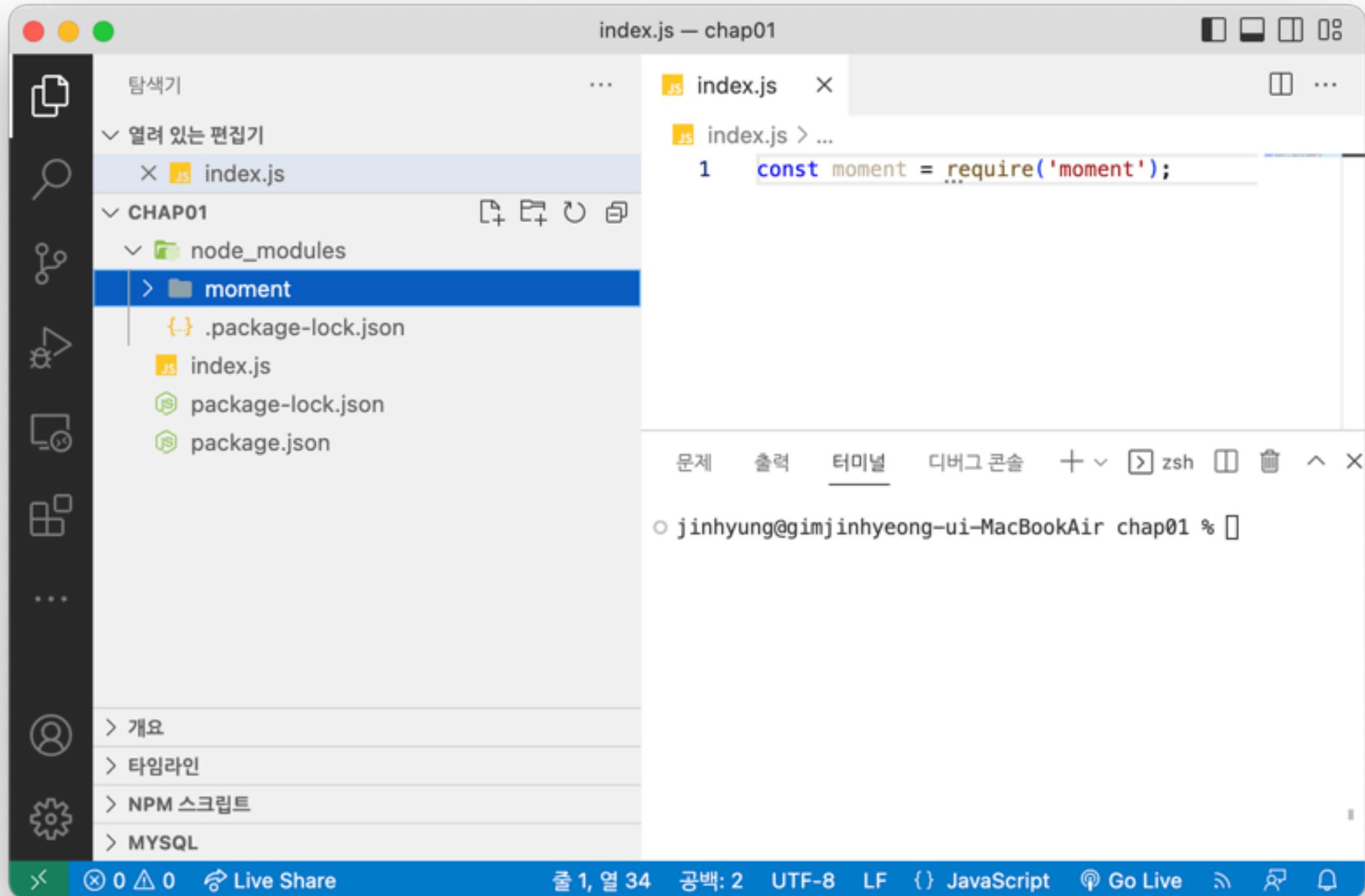
Weekly Downloads
17,368,051

Version	License
2.29.4	MIT
Unpacked Size	Total Files
4.23 MB	533
Issues	Pull Requests
194	53

Last publish
6 months ago

npm 활용하기

설치된 moment 확인 및 사용할 준비



moment 라이브러리 활용하기

간단한 날짜 출력

```
const moment = require('moment');  
require("moment/locale/ko");  
  
let now = moment().format("YYYY-MM-DD");  
console.log(`현재 날짜는 ${now} 입니다.`);  
  
let endClass = moment("2023-01-27", "YYYY-MM-DD").fromNow();  
console.log(`종강날짜 : ${endClass}`);
```

연습문제

moment 활용하기

mydate.js 파일에 SNS 에서 사용할 만한 함수 정의하기

```
/**
 * 오늘 날짜의 글일경우 N분전 또는 N시간전 등으로 표기
 * 오늘 이전의 날짜의 경우엔 YYYY-MM-DD 형식으로 표기
 * @param {string} date 'YYYY-MM-DD HH:mm:ss' 형식의 문자
 * @return {string}
 */
exports.dateFromNow = (date) => {

}

/**
 * 현재 등록된 글이 새 글인지 판단해주는 함수
 * 글을 작성한지 10분 이내의 글은 true를, 이후면 false를 반환
 * @param {string} date 'YYYY-MM-DD HH:mm:ss' 형식의 문자
 * @return {bool}
 */
exports.isNewFeed = (date) => {

}
```