

삼성 청년 SW 아카데미

Node.js

<알림>

본 강의는 삼성 청년 SW아카데미의 컨텐츠로
보안서약서에 의거하여
강의 내용을 어떠한 사유로도 임의로 복사,
촬영, 녹음, 복제, 보관, 전송하거나
허가 받지 않은 저장매체를
이용한 보관, 제3자에게 누설, 공개,
또는 사용하는 등의 행위를 금합니다.

7장. 웹 크롤러

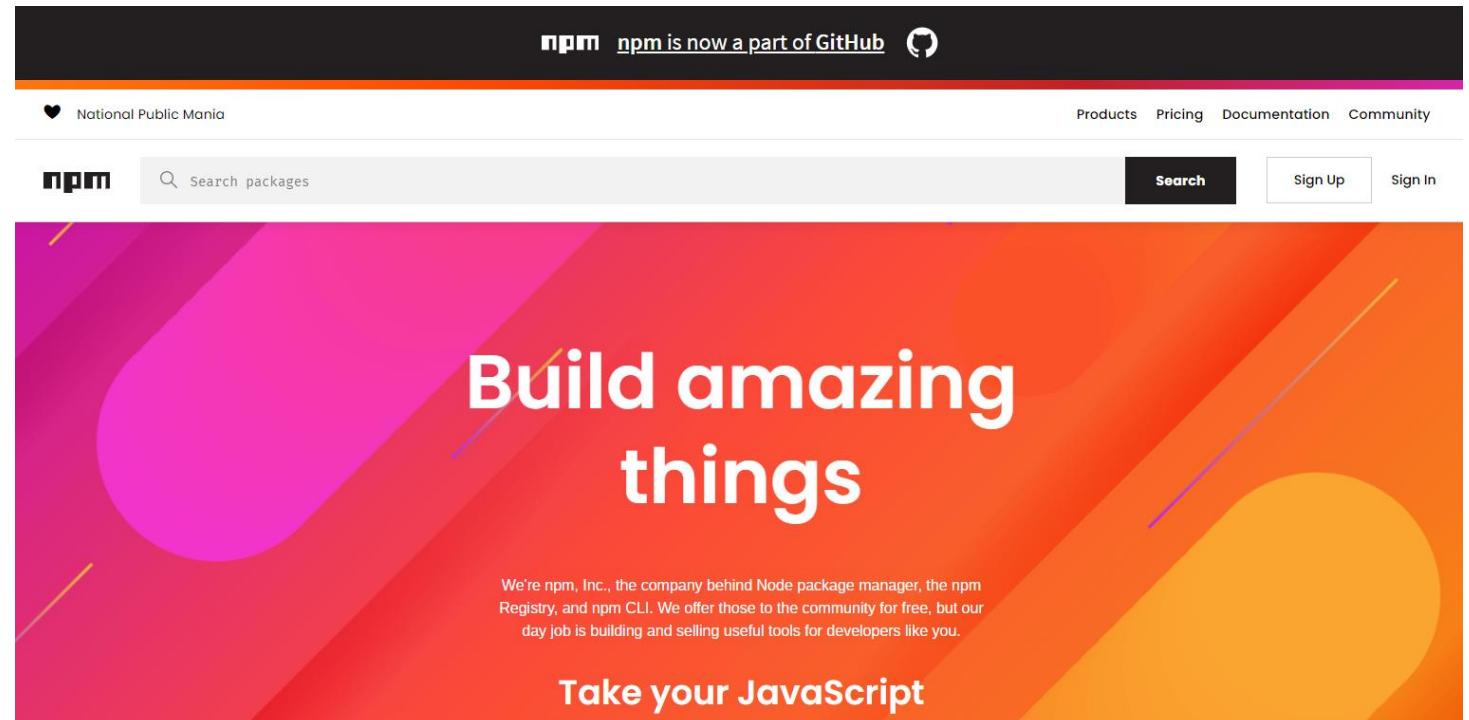
챕터의 포인트

- npm
- 웹 크롤러 개념
- cheerio
- 크롤링 응용

npm

node package manager

- 자바스크립트 프로그래밍 언어를 위한 패키지 관리자
- <https://www.npmjs.com/>



This website stores cookies on your computer. These cookies are used to collect information about how you interact with our website and allow us to remember you. We use this information in order to improve and customize your browsing experience and for analytics and metrics about our visitors both on this website and other media. To find out more about the cookies we use, see our [Privacy Policy](#).

If you decline, your information won't be tracked when you visit this website. A single cookie will be used in your browser to remember your preference not to be tracked.

Accept

Decline

npm init - 패키지 생성

```
C:\Users\mincoding\Desktop\node-test>npm init
npm WARN config global `--global`, `--local` are
deprecated. Use `--location=global` instead.
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 init` 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: (test) node-practice
version: (1.0.0)
description: for ssafy node.js class
entry point: (index.js)
test command:
git repository:
keywords:
author: Jon Jaryong Lee
license: (ISC)
About to write to C:\Users\mincoding\Desktop\node-test\package.json:
```

```
1  {
2      "name": "node-practice",
3      "version": "1.0.0",
4      "description": "for ssafy node.js class",
5      "main": "index.js",
6      ▷ Debug
7      "scripts": {
8          "test": "echo \\\"Error: no test specified\\\" && exit 1"
9      },
10     "author": "Jon Jaryong Lee",
11     "license": "ISC"
12 }
```

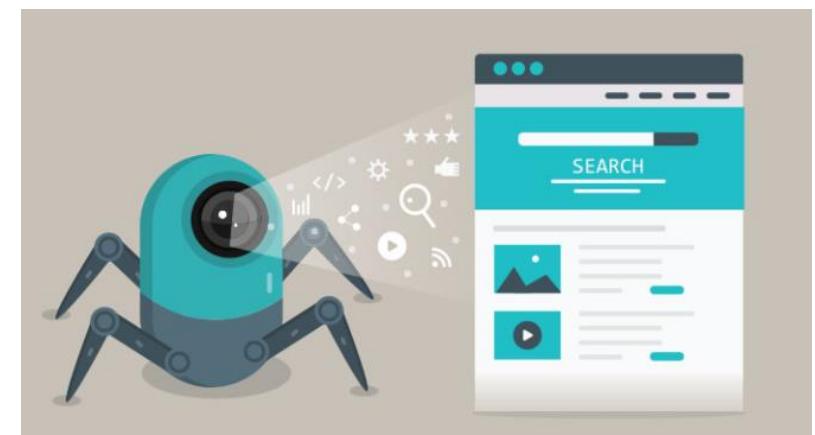
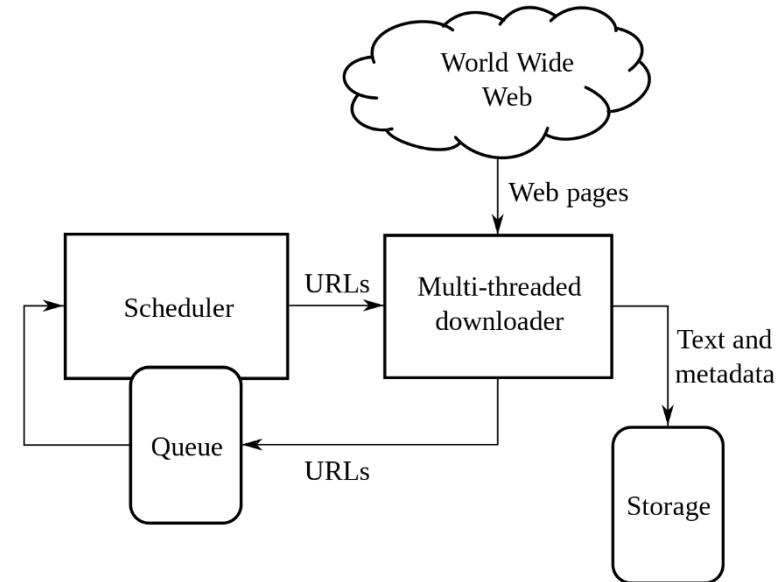
웹 크롤러 개념

Web Scraping

- 웹 사이트에서 데이터를 추출하는 프로그래밍 기술
- 사이트에 있는 정보를 가져오는 행동

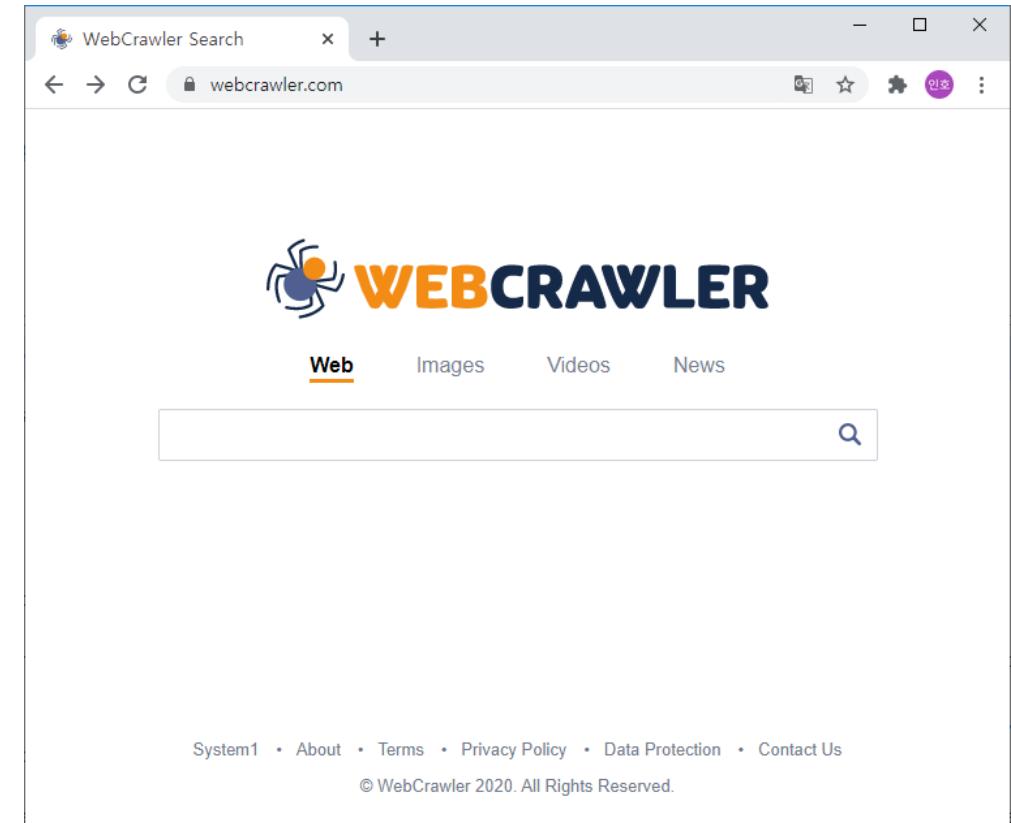
Web Crawler

- URL를 가져온 후, 또 다시 URL에 접속을 반복하여 웹을 탐색하며 Web Scraping을 하는 봇
(Spiderbot이라고도 함)
 - 재귀적으로 웹을 탐색하며 사이트를 가져오는 프로그램
-
- https://en.wikipedia.org/wiki/Web_scraping
 - https://en.wikipedia.org/wiki/Web_crawler



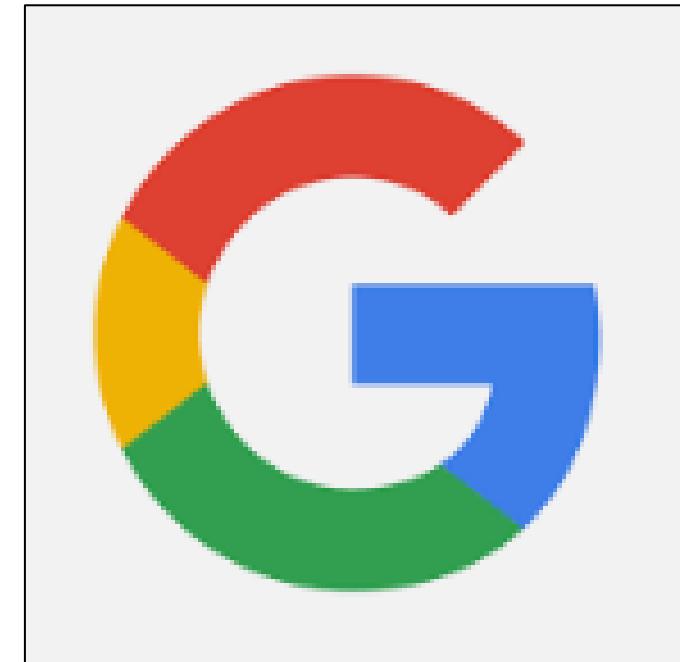
WEBCRAWLER.com

- 현대식 검색엔진의 시초
- 1994년 출시



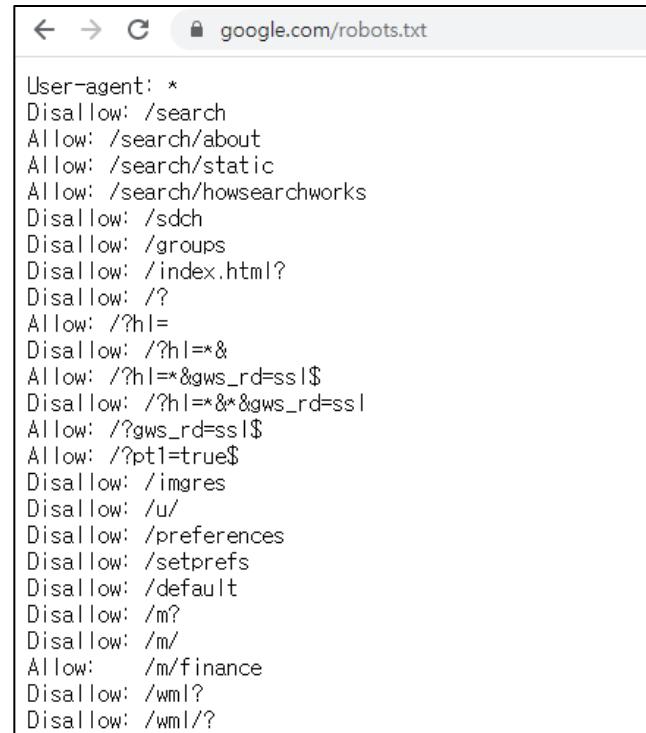
Google.com

- 1998년 시작
- 2006년 Youtube 인수
- 2006년 Google Korea 법인 설립
- 2010년 세계 1위 포털 사이트



robots.txt

- Robots exclusion standard 표준 규약
- 각 홈페이지 뒤에 robots.txt 를 붙이면 조회할 수 있다. (ex: <https://www.google.com/robots.txt>)
- 웹 크롤러들이 접근하지 못하도록 제한하는 규약
- 교육 목적 외 해당 robots로 접근을 막아놨는데도 불구하고
서비스 출시 및 영리적 목적으로 사용되는 경우
법적인 분쟁이 발생할 수 있다.



The screenshot shows a browser window with the URL "google.com/robots.txt" in the address bar. The page content is a plain text file containing the following robots.txt rules:

```
User-agent: *
Disallow: /search
Allow: /search/about
Allow: /search/static
Allow: /search/howsearchworks
Disallow: /sdch
Disallow: /groups
Disallow: /index.html?
Disallow: /?
Allow: /?hl=
Disallow: /?hl=&
Allow: /?hl=&gws_rd=ssl
Disallow: /?hl=&&gws_rd=ssl
Allow: /?gws_rd=ssl
Allow: /?pt1=true
Disallow: /imgres
Disallow: /u/
Disallow: /preferences
Disallow: /setprefs
Disallow: /default
Disallow: /m?
Disallow: /m/
Allow: /m/finance
Disallow: /wml?
Disallow: /wml/?
```

puppeteer

- 크롬 환경을 제어하기 위한 라이브러리
- 크롬의 기반인 Chromium 으로 작성되었기 때문에 가장 빠른 크롤링 라이브러리중 하나
- headless 지원
- pdf, 스크린샷 지원
- javascript를 통해 제어 가능
- SPA 크롤링 가능

Install

```
> npm i puppeteer
```

Repository

❖ github.com/puppeteer/puppeteer

Homepage

🔗 [github.com/puppeteer/puppeteer#read...](https://github.com/puppeteer/puppeteer#readme)

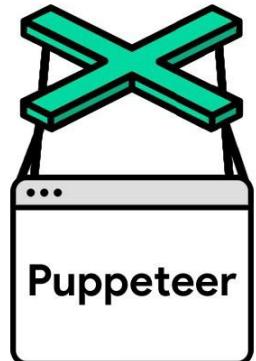
⬇ Weekly Downloads

3,527,096



headless browser

- 창이 없다는 의미
- 브라우저는 직접 띄우는것 대신 프로그램만으로 동작하는 브라우저



puppeteer

- **puppeteer.launch**

- 브라우저를 생성.
 - headless : false
 - 기본 값은 true이며 false로 지정시 브라우저가 보인다.

- **browser.newPage()**

- 해당 브라우저에 새 창을 띄운다.

- **page.goto('주소')**

- 해당 페이지를 해당 주소로 이동

```
const puppeteer = require('puppeteer')

const main = async() => {
  const browser = await puppeteer.launch({
    headless:false
  });
  const page = await browser.newPage();
  await page.goto('https://www.naver.com');

}

main();
```

puppeteer로 pdf 만들기

- **headless: true**
 - pdf는 headless:true 모드에서만 동작
- **page.pdf()**
 - test.pdf 의 A4 형식으로 pdf 생성
- **browser.close**
 - 브라우저 종료

```
const puppeteer = require('puppeteer')

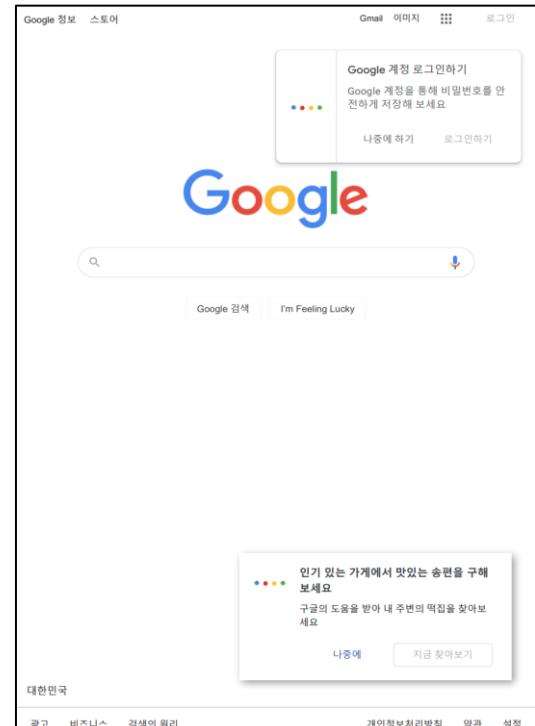
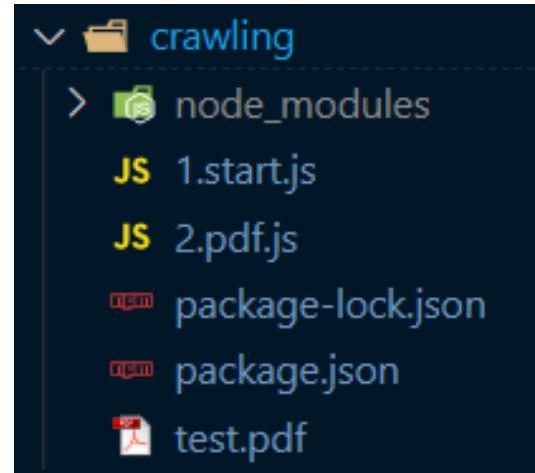
const main = async () => {

  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.google.co.kr');
  await page.pdf({ path: 'test.pdf', format: 'A4' });

  await browser.close();
}

main();
```



puppeteer로 screenshot 찍기

- **headless: true**
 - screenshot은 headless true/false 상관없이 동작한다.
- **page.screenshot**
 - path: 경로
 - fullpage
 - true 일시 전체 스크롤을 스크린샷 찍는다.

```
const puppeteer = require('puppeteer')

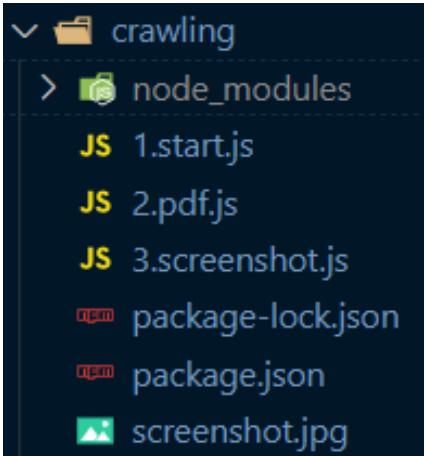
const main = async () => {

  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.google.co.kr');
  await page.screenshot({ path: 'screenshot.jpg', fullPage: true });

  await browser.close();
}

main();
```



네이버 웹툰 크롤링하기

- 월요 웹툰 전체의 제목과 링크를 가져와 보기

The screenshot shows the NAVER Webtoon homepage. At the top, there's a search bar and a login button. Below the header, there are tabs for 'Home' (홈), 'Webtoon' (웹툰), 'Best Doctor' (베스트 도전), 'Adventure Manhwa' (도전만화), and 'My Page' (마이페이지). A sidebar on the right lists 'Top Recommended Webtoons' (웹툰을리기 속) and 'Top Popular Webtoons' (인기금상승웹툰). The main content area features a 'Weekly Recommended Webtoon' section (월요 추천 웹툰) with three entries: 'Mal Bakwang' (말박왕), 'Changgyo' (창교), and 'Ip Sulim Ebeon Namja' (입술이 예쁜 남자). Below this is a 'Weekly Complete Webtoon' section (월요 전체 웹툰) displaying nine more webtoons like 'Lutti Gumivari' (뷰티풀 군비리) and 'Wondoreumie' (원드브레이커).

네이버 월요웹툰 크롤링하기

- 먼저 개발자 도구에서 querySelectorAll로 해당 선택자를 가져온다.

The screenshot shows the Naver Monday Webtoon page. It features two comic strips: one titled '참교육' (Changgyuk) and another titled '뷰티풀 군바리' (Beautiful Jungsabari). The developer tools console on the right displays the following code and output:

```
length: 80
[[Prototype]]: NodeList
> document.querySelectorAll("ul.img_list >li dl>dt")
< NodeList(80) [dt, dt, dt, dt, dt, dt, dt, dt, dt,
, dt, dt,
, dt] i
  ▶ 0: dt
  ▶ 1: dt
  ▶ 2: dt
  ▶ 3: dt
  ▶ 4: dt
  ▶ 5: dt
  ▶ 6: dt
```

네이버 웹툰 크롤링 하기

- 네이버 월요 웹툰으로 이동
- `page.evaluate`
 - evaluate안에서는 `document` 접근이 가능
(기존 node에서는 접근이 되지 않는다.)
 - `Array.from`
 - 배열로 만들어주는 함수
 - `document.querySelectorAll`에 담기는 배열은 배열 메서드를 사용할수 없다.
 - `for`문만 사용 가능

```
const main = async () => {  
  
    const browser = await puppeteer.launch({  
        headless: true  
    });  
    const page = await browser.newPage();  
  
    await page.goto('https://comic.naver.com/webtoon/weekdayList?week=mon');  
  
    const data = await page.evaluate(() => {  
        const webToonLists = document.querySelectorAll('ul.img_list > li dl > dt');  
  
        const titles = Array.from(webToonLists).map(li => li.textContent.trim());  
  
        return titles;  
    })  
  
    console.log(data);  
  
    await browser.close();  
}  
  
main();
```

네이버 웹툰 크롤링 하기

- 결과 확인
 - fullname의 형태로 나와있지 않다.

```
C:\Users\dhsdb\Desktop\자료\ssafy_c_class_web\실제 강의\1-웹\3주차(노드, AWS)\8기-node\code\crawling>node 4.naver-webtoon.js
[
  '최후의 금빛아이',  '경비실에서 안...',  '세번째 로망스',  '아마도',
  '파견체',          '그림자 신부',      '싸이코 리벤저',  '백호랑',
  '왕따협상',        '매지컬 급식:...',  '나만의 고막남친', '지옥연애환담',
  '악녀 18세 ...',   '칼가는 소녀',      '모노마니아',    '흔들리는 세계...',
  '또다시, 계약...',  '슈퍼스타 천대리',  '결혼공략',     '역주행!',
  '사막에 편 달',   '기사님을 지켜줘',  '헬로맨스',    '디나운스',
  '오로지 오로라',   '이제야 연애',     '남주서치',    '별을 쓸는 소...'
]
```

네이버 웹툰 크롤링하기

- **fullname 가져오기**
 - a tag의 attribute중에 title에 fullName이 담겨있는것을 확인

```
▼<dl>
  ▼<dt>
    <a href="/webtoon/list?titleId=795297&weekday=mon" title="신화급 귀속 아이템을 손에 넣었다" 신화급 귀속 ...
  </dt>
  ▶<dd class="desc">...</dd>
  ▶<dd>...</dd>
  ▶<dd class="more">...</dd>
</dl>
```

네이버 웹툰 크롤링 하기

- **fullName 가져오기**

```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://comic.naver.com/webtoon/weekdayList?week=mon');

  const data = await page.evaluate(() => {
    //.textContent로 가져오기
    //const webToonLists = document.querySelectorAll('ul.img_list >li dl>dt');

    //const titles = Array.from(webToonLists).map(li => li.textContent.trim())

    // fullname 가져오기
    const webToonLists = document.querySelectorAll('ul.img_list >li dl>dt>a');

    const titles = Array.from(webToonLists).map(li => li.getAttribute('title').trim())
  })
  return titles;
}

console.log(data);
await browser.close();
}

main();
```

네이버 웹툰 크롤링 하기

- 결과 확인
 - fullName의 형태로 가져왔다.

```
C:\Users\dhsdb\Desktop\자료\ssafy_c_class_web\실제 강의\1-웹\3주차(노드, AWS)\8기-node\code\crawling>node 4.naver-webtoon.js
[
  '참교육',
  '뷰티풀 군바리',
  '신의 탑',
  '퀘스트지상주의',
  '원드브레이커',
  '장씨세가 호위무사',
  '팔이피플',
  '호랑신랑뎐',
  '백수세끼',
  '소녀의 세계',
  '신화급 귀속 아이템을 손에 넣었다',
  '앵무살수',
  '버림받은 왕녀의 은밀한 침실',
```

Daum 뉴스 <IT> 섹션의 헤드라인 기사를 크롤링하자

- <https://news.daum.net/digital#1>
- 빨간색 박스 친 곳만 가져오기
 - 신문사를 제외한 내용만 가져오기

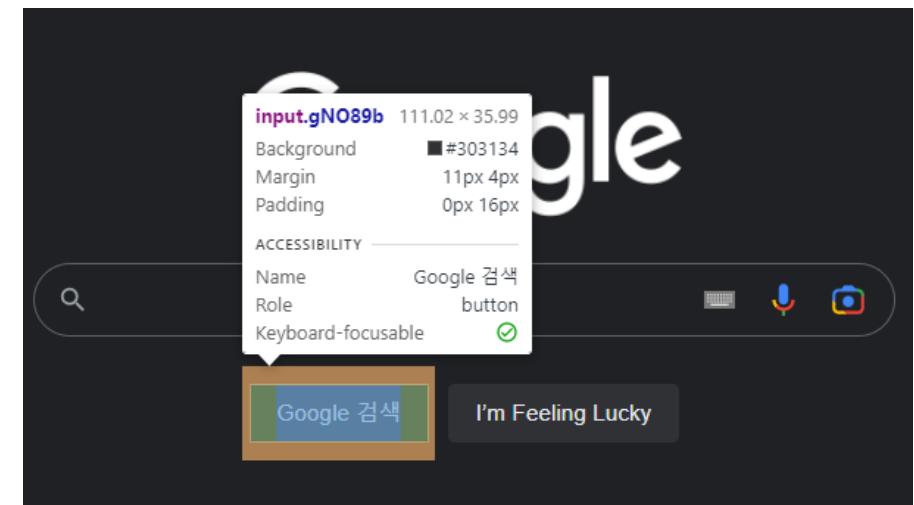
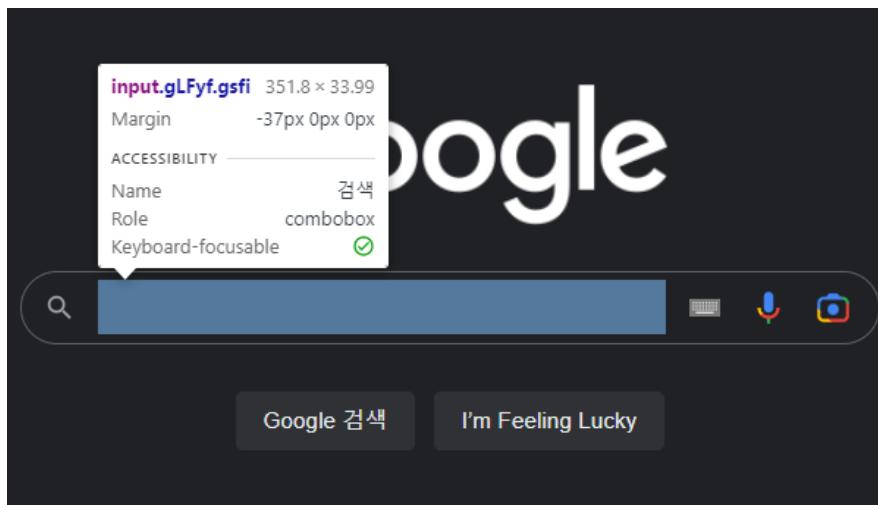
The screenshot shows the Daum News homepage with the 'IT' category selected. The 'IT 뉴스' section is highlighted with a red box, displaying headlines from multiple news sources. To the right, there are sections for stock prices (코스피, 코스닥) and a sidebar with a weather forecast for Seoul.

Headline	Source
[반도체 초격차 포럼 좌담회]은퇴자 교수로 활용..인력 양성 생태계도 시급	전자신문
JTBC 전면 디지털 혁신 개편, 기자들 우려에 잠정 보류	미디어오늘
"소비자 불편은 뒷전".카카오-구글 기싸움에 업데이트 막혀	동아일보
2022 공군 해커톤, 수상팀 늘리고 상금도 높여	전자신문
과기정통부, NTIS 서포터즈 출범..경진대회·공모전도 개최	연합뉴스
"마스크 벗으니 얼굴이 드러났다"..윤곽 필러에 탈모치료제, 명크림까지 '관심'	조선비즈
용산 대통령 집무실 앞에 대형 전광판이 설치된 이유는?	경향신문
'아웃링크' 카카오톡, 구글플레이에서 최신 버전 심사 거부	지디넷코리아
5G 주파수 추가 할당 받는 LGU+, 농어촌 품질격차부터 줍힌다	뉴시스
빅테크 갑질 막는다..국회 '갑질보호 TP' 발족	아시아경제

크롤링 응용

검색후 이동해서 캡쳐 찍기

- 먼저 필요한 selector 파악하기
 - input 입력에 필요한 selector input.gsf1
 - 검색 버튼을 위한 selector



검색후 이동해서 스크린샷 찍기

- 동작과정

- value=치킨
- input 버튼을 찾아서 클릭
- 스크린샷 찍기

```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.google.co.kr');
  await page.evaluate(()=>{
    document.querySelector("input.gsf").value="치킨";
    document.querySelector("input.gNO89b").click();
  })

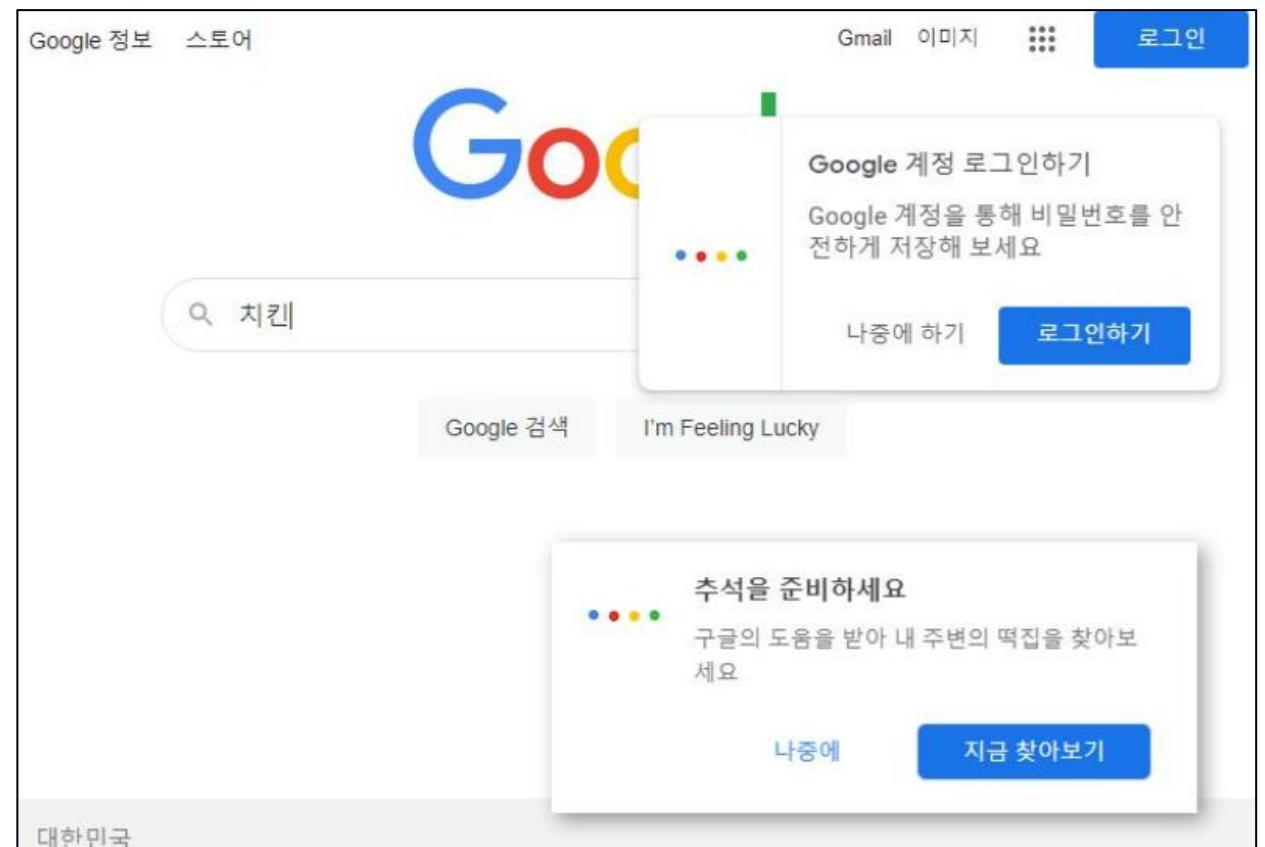
  await page.screenshot({ path: 'screenshot1.jpg', fullPage: true })

  await browser.close();
}

main();
```

결과 확인하기

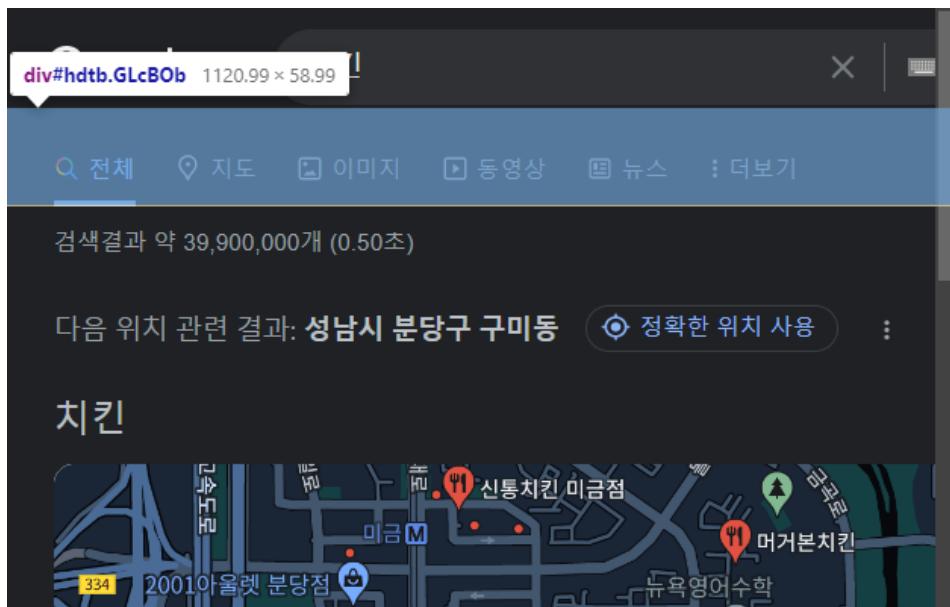
- 의도한 바와 다르게 치킨만 입력되고 동작하지 않는다.



검색 후 이동 해서 스크린샷 찍기

- **waitForSelector**

- 특정 선택자가 생성되기까지 대기
- 검색하면 네비게이션 부분이 생기기 때문에 해당 선택자가 생성되기까지 대기한다.



```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: false
  });
  const page = await browser.newPage();

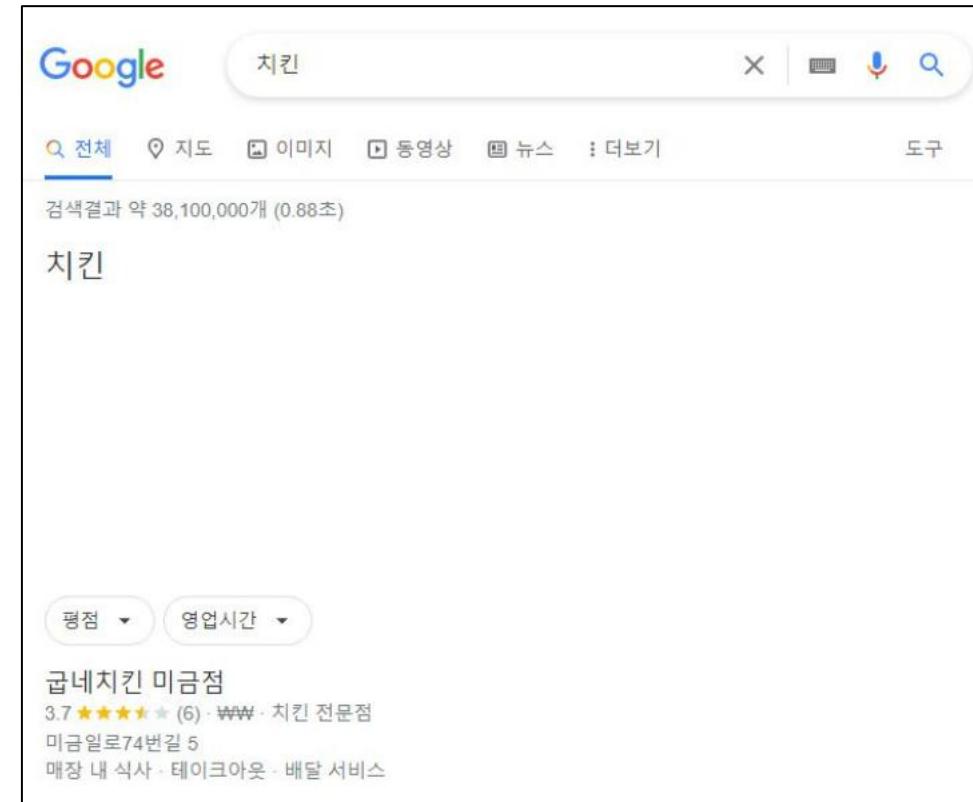
  await page.goto('https://www.google.co.kr');
  await page.evaluate(()=>{
    document.querySelector("input.gsf").value="치킨";
    document.querySelector("input.gNO89b").click();
  })
  // selector가 오키로 대기
  await page.waitForSelector(".GLcB0b")
  await page.screenshot({ path: 'screenshot1.jpg', fullPage: true })

  await browser.close();
}

main();
```

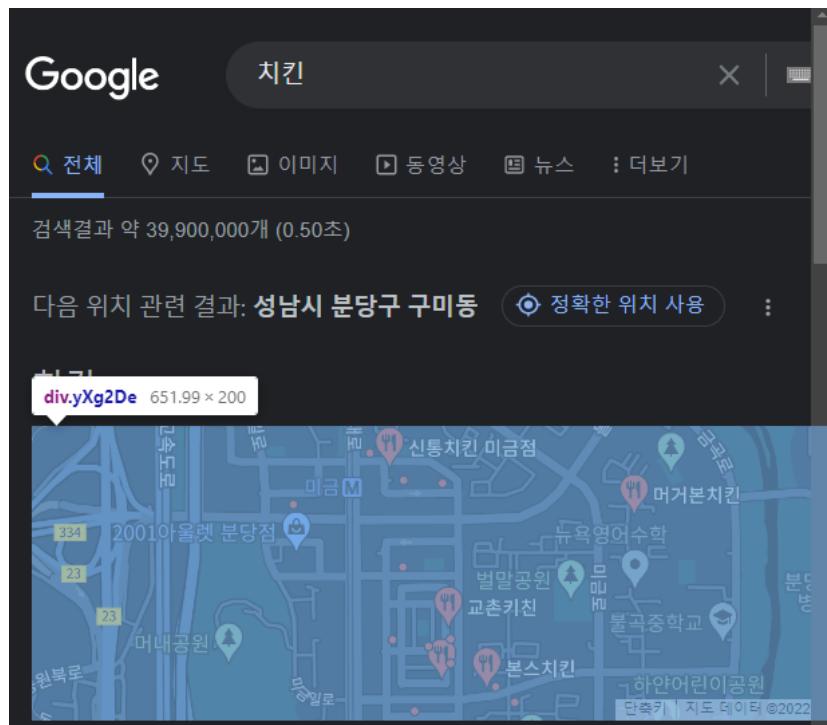
결과 확인하기

- 치킨 검색 후 이동까지는 나오게 되었다.
- 그 외 지도등 내용이 보이지 않는다.



검색후 이동해서 스크린샷 찍기

- 이번에는 해당 지도 부분이 나올수있도록 대기를 걸어보자.



```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.google.co.kr');
  await page.evaluate(()=>{
    document.querySelector("input.gsf").value="치킨";
    document.querySelector("input.gN089b").click();
  })
  // selector가 올바르게 등록되었는지 확인
  await page.waitForSelector(".yXg2De")
  await page.screenshot({ path: 'screenshot1.jpg', fullPage: true })

  await browser.close();
}

main();
```

결과 확인하기

- 모든 정보들을 잘 가져오게 되었다.

Google 치킨

전체 지도 이미지 동영상 뉴스 더보기 도구

검색결과 약 35,500,000개 (0.51초)

치킨

지도 데이터 ©2022 TMap Mobility

평점 ▾ 영업시간 ▾

굽네치킨 미금점
3.7 ★★★★☆ (6) · ₩₩₩ · 치킨 전문점
미금일로74번길 5
매장 내 식사 · 테이크아웃 · 배달 서비스

머거본치킨
5.0 ★★★★★ (1) · 닭 튀김
구미동 77
매장 내 식사 · 테이크아웃 · 배달 서비스

신풍치킨 미금점
4.0 ★★★★☆ (80) · ₩₩₩ · 음식점
금곡동 147
매장 내 식사 · 테이크아웃 · 배달 서비스

메가박스 크롤링

- <https://www.megabox.co.kr/movie>
- 로딩(비동기 통신)에 대한 비동기 데이터 크롤링
- 각 포스터 별 src 가져오기



메가박스 이미지 크롤링

- 필요한 선택자 파악하기

전체영화

박스오피스 상영예정작 특별상영 필름소사이어티

img.poster.lozad 230 x 331

15 헌트
예매율 16.5% | 개봉일 2022.08.10
1.6k 예매 Dolby CINEMA

12 육사오 (6/45)
예매율 11.3% | 개봉일 2022.08.24
457 예매

All 아라시 20주년 투어 콘서트
예매율 11% | 개봉일 2022.08.31
639 예매 Dolby CINEMA

```
> document.querySelectorAll(".movie-list img.poster")
< NodeList(20) [img.poster.lozad, img.poster.lozad, img.poster.lozad]
  ▷ 0: img.poster.lozad
  ▷ 1: img.poster.lozad
  ▷ 2: img.poster.lozad
  ▷ 3: img.poster.lozad
  ▷ 4: img.poster.lozad
  ▷ 5: img.poster.lozad
  ▷ 6: img.poster.lozad
  ▷ 7: img.poster.lozad
  ▷ 8: img.poster.lozad
  ▷ 9: img.poster.lozad
  ▷ 10: img.poster.lozad
  ▷ 11: img.poster.lozad
  ▷ 12: img.poster.lozad
  ▷ 13: img.poster.lozad
  ▷ 14: img.poster.lozad
  ▷ 15: img.poster.lozad
  ▷ 16: img.poster.lozad
  ▷ 17: img.poster.lozad
  ▷ 18: img.poster.lozad
  ▷ 19: img.poster.lozad
  length: 20
  [[Prototype]]: NodeList
```

메가박스 이미지 크롤링

- **waitForSelector 활용**
 - loading 후 해당 div가 생성
 - 해당 선택자가 생성되기를 대기

```
const puppeteer = require('puppeteer')

const main = async () => {
  const browser = await puppeteer.launch({
    headless: true
  });
  const page = await browser.newPage();

  await page.goto('https://www.megabox.co.kr/movie');
  await page.waitForSelector(".movie-list img.poster")
  const data = await page.evaluate(()=>{
    const images = document.querySelectorAll('.movie-list img.poster')
    const result = Array.from(images).map(li=> li.getAttribute('src'));
    return result
  })
  console.log(data);

  await browser.close();
}

main();
```

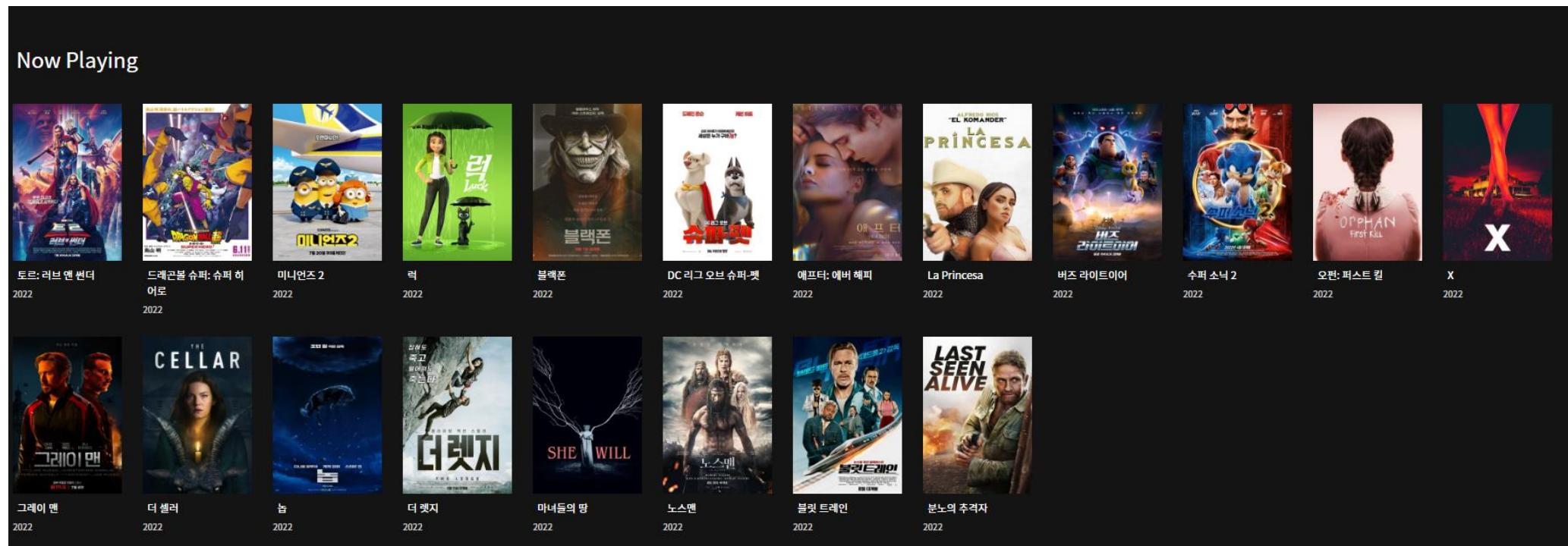
메가박스 이미지 크롤링

- 결과 확인

```
C:\Users\dhsdb\Desktop\자료\ssafy_c_class_web\실제 강의\1-웹\3주차(노드, AWS)\87|node\code\10_crawling>node 8.megabox.js
[
  'https://img.megabox.co.kr/SharedImg/2022/08/05/QDUC0cjm2bnWDCCQPYpQvelnoFe1CCfH_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/29/wMbAzWp0ahUuTxhD5hq8DzSXEjQD6gNY_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/05/09/6zfAYe6IrZ8BWnruqEfafwakt5cUjWgX_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/11/fcfDt05PrS4H5YusAg7BjUZ3JfhS5MRj_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/07/28/1ogGdwYxCNJ9MTnizlZLdZ6REjg6xX1z_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/29/oUqrNQTflUqvHUQG6kv1zF8SEhJSomfh_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/02/hQsvDEd41AY0pm0N6fAhJ55ouwS5K3wb_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/04/oPoYTP4zSq5iWCCGYj4SQ3tuSU5J89Rl_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/03/B5P9rlhS5BwMjvPi35pyH60I5Dv75kDm_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/03/ypNwU8S72y0b03rgLEc01ih8uG6mcZgm_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/29/v1Y6EQN1rpaADkfQhCPwnSxFgGYZwfB0_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/03/HJQ3kqIL5kUJBmKQSc3HOxZV7SUpHWGx_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/07/04/wo9hN6dpVFzHp4d3MpGmPEGTC0CE2yt_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/06/07/S3GJQZbpshoIx0Lelerscl9rlI14FHqK_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/29/CyFJcEEUDngBC07vzJra2mSocDg1Trmk_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/07/06/KnVXVmorgtWJC8EauyxVAUGEGLuDx95_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/10/sdYAZCCnn0gvx119La32kjjInsNB9CuB_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/19/Wx0F5W0amgT0s5fmLXzMuiSQvttecJMLc_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/08/26/0yefBJeFcMey7RXhnYWLAeGPBTWMQiUx_420.jpg',
  'https://img.megabox.co.kr/SharedImg/2022/02/17/djm7aYuL9bQGZRxyUH75wATz9ub9ouk_420.jpg'
]
```

도전 - 영화 페이지 크롤링

- <https://loy124.github.io/vue-movie/> 사이트 접속
 - 해당 사이트는 Vue.js로 생성되어 있다.
 - 해당 포스터 src, 제목 배열, 객체 형식으로 가져오기



도전 - 영화 페이지 크롤링

- 목표 결과

```
[  
  {  
    title: '토르: 러브 앤 썬더',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//bZLrpWM065h5bu1msUcPmLFsHBe.jpg'  
  },  
  {  
    title: '드래곤볼 슈퍼: 슈퍼 히어로',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//uohymzBVaIYjbnoQstbnlia6ZPJ.jpg'  
  },  
  {  
    title: '미니언즈 2',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//1heBUD8o0sgdqLWyeXkyLR2POKb.jpg'  
  },  
  {  
    title: '럭',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//nN2iHej7TyISjXC6Jl577aqpDHJ.jpg'  
  },  
  {  
    title: '블랙폰',  
    imageSrc: 'https://image.tmdb.org/t/p/w300//eJPIkRiYGnvQaPCZtTRTDmhMJVK.jpg'  
  },  
]
```

8장. express

챕터의 포인트

- 간단한 express 웹서버
- 요청과 응답
- 클라이언트와 CORS
- Routing

간단한 express 웹서버

nodemon 이란?

- 서버 코드 수정시마다 매번 서버를 재부팅해야하는 불편함
- nodemon 은 코드가 수정되면 자동으로 서버를 재시작



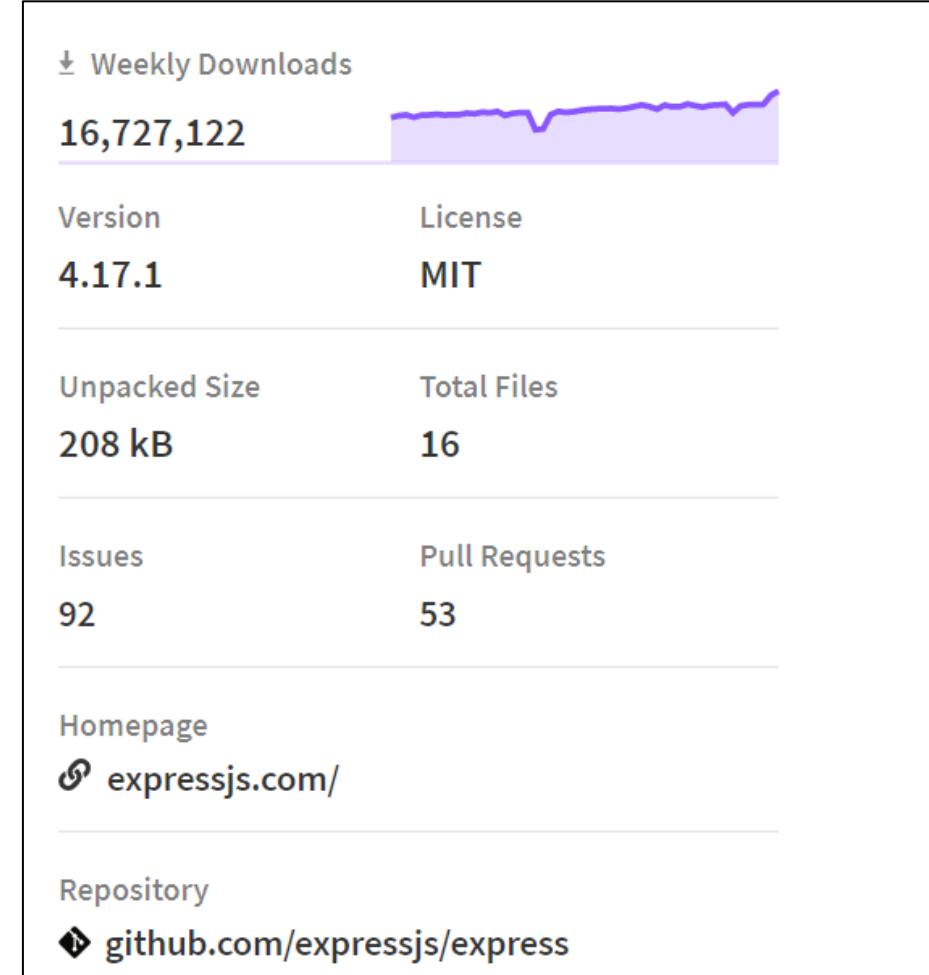
global install (전역 설치)

- package.json 이 위치한 프로젝트 디렉터리 이외에도,
 - 어떤 경로에서든 패키지 호출 가능
 - 커멘드라인에서 실행하는 프로그램은 전역 설치가 기본
-
- **npm install --location=global nodemon**
 - nodemon 전역 설치
 - **npm list --location=global --depth=0**
 - 전역 패키지 확인

```
C:\Users\mincoding\Desktop\node-test>npm list --location=global --depth=0
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.
C:\Users\mincoding\AppData\Roaming\npm
└─ nodemon@2.0.16
```

Node.js의 대표 웹 프레임워크

- http와 Connect 컴포넌트를 기반으로 만들어짐
- Node.js의 API들을 단순화
- 쉬운 확장성



전역설치가 아니라, 지역 설치해보자.

package.json 이 위치한 경로로 이동 후,

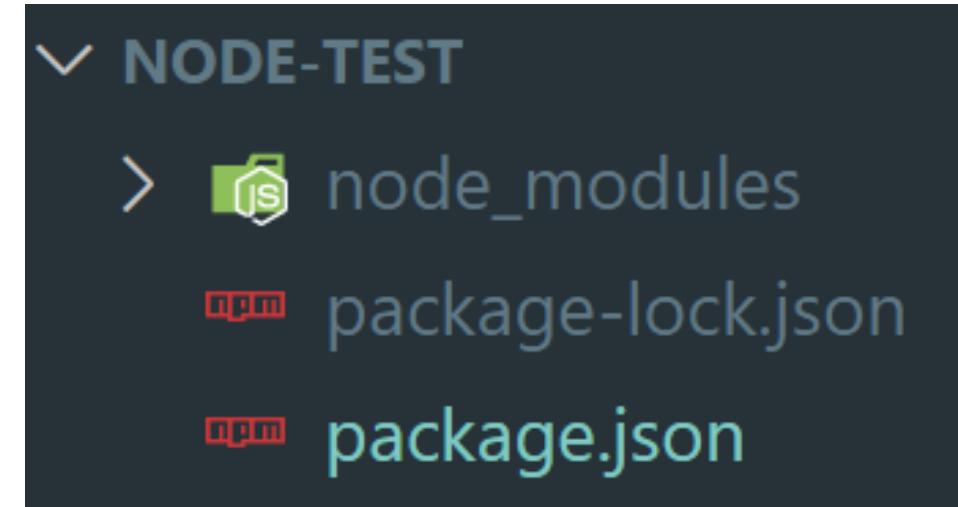
npm install express // 현재 프로젝트에 express 설치

```
C:\Users\mincoding\Desktop\node-test>npm i express
```

install 의 약어인 i 를 써도 된다.

각각의 개념들이 무엇을 의미하는지 숙지하기

- **node_modules**
 - 실제 설치된 패키지
- **package.json**
 - 명세서
 - 프로젝트에 관련된 데이터가 담긴다
 - 프로젝트 정보 및 빌드 방식
 - 필요한 패키지 목록 정리
- **package-lock.json**
 - 상세 명세서
 - 각 패키지 별 정확한 버전명이 명시 되어 있음



package.json 안에 dependencies 가 추가됨

- **dependencies**

- 한국어로 의존성
- 우리가 만든 패키지(node-practice) 를 실행시키기 위해 필요한 "다른 패키지" 목록
- node-practice 패키지는 express 가 설치되어 있지 않으면 작동 하지 않는다.
즉, 의존(depend)하고 있다

```
"dependencies": {  
    "express": "^4.18.1"  
}
```

package-lock.json

- 열어보면 1000 줄이 넘어가는 파일
 - package.json 만으로 충분하지 않다.
 - package.json 요약 명세서
 - package-lock.json 상세 명세서
 - express 역시도 하나의 패키지이므로, "다른 패키지들" 을 dependencies 로 가지고 있다.
 - 다른 개발 PC 또는 서버 컴퓨터로 우리가 만든 패키지(node-practice) 를 옮겼을 때,
 - 원래 개발 PC 와 완벽하게 동일한 환경에서 우리 패키지를 동작시키기 위함
 - 즉, 협업 시엔 package-lock.json 까지 공유

```
1015      "integrity": "sha512-BNGbL  
1016    }  
1017  }  
1018 }  
1019 |
```

[참고] express 의 dependencies 를 확인해보자

Confidential

- npm list --depth=0

- 현재 node-practice 패키지에서 확인되는 dependencies 는 express 하나밖에 없다.

```
node-practice@1.0.0
└── express@4.18.1
```

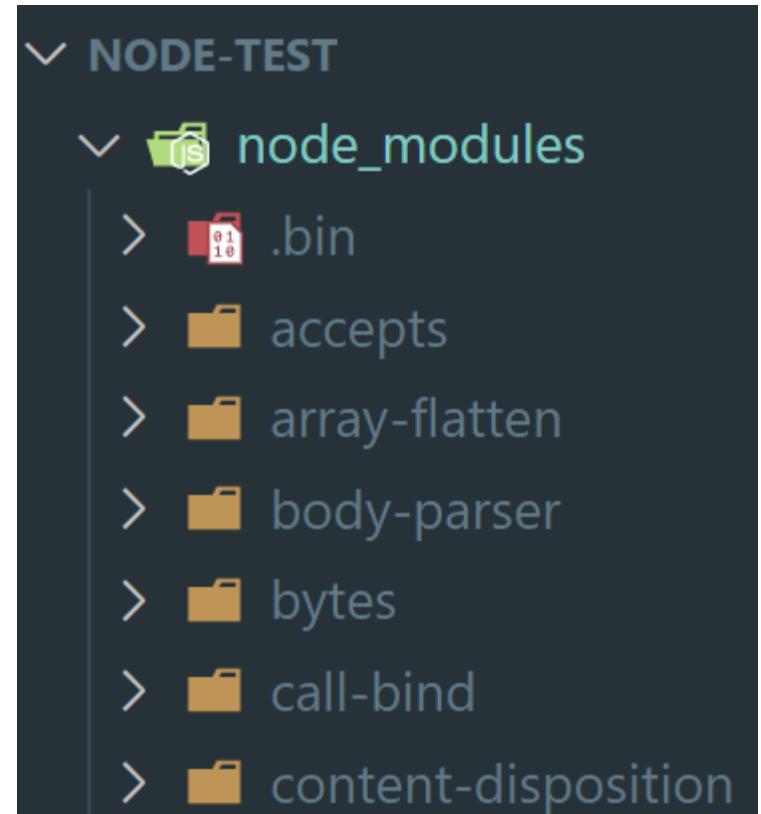
- npm list --depth=1

- express 패키지의 dependencies 출력됨
- 즉, --depth 숫자가 늘어날수록 패키지 안에 패키지 안에 패키지...의 dependencies 확인 가능

```
node-practice@1.0.0 C:\Users\min
└── express@4.18.1
    ├── accepts@1.3.8
    ├── array-flatten@1.1.1
    ├── body-parser@1.20.0
    ├── content-disposition@0.5.4
    ├── content-type@1.0.4
    ├── cookie-signature@1.0.6
    ├── cookie@0.5.0
    ├── debug@2.6.9
    └── depd@2.0.0
```

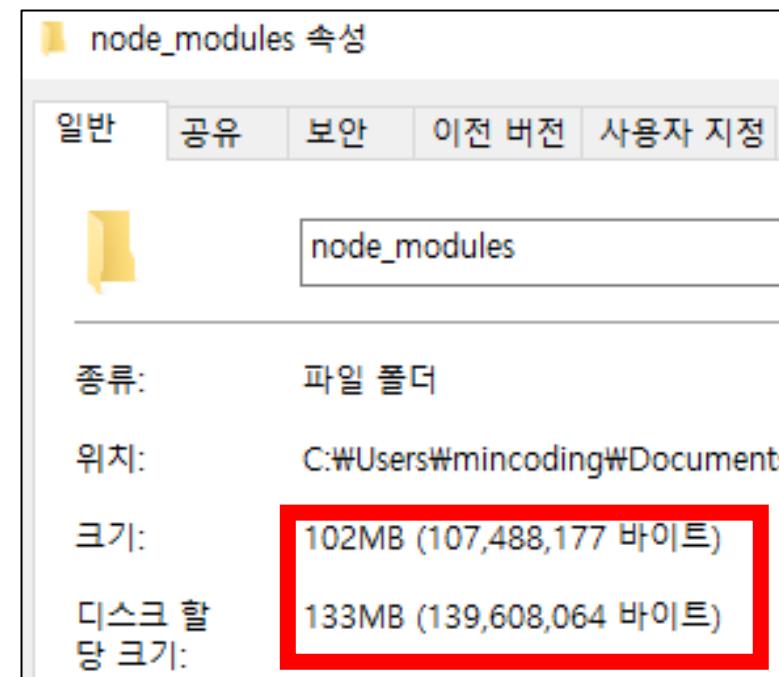
node_modules

- package-lock.json 에 기입된대로 설치된 실제 dependencies
- 협업할때나, github 업로드 시 node_modules 은 제외하고 공유
 - package.json, package-lock.json 두 개의 명세서만 있으면 된다.
 - 위 명세로 dependencies 설치가 가능하다.
 - node_modules 의 용량은 어마어마하다.
- 간단한 테스트:
 - node_modules 디렉터리를 삭제한 후,
 - 커멘드라인에 npm i 를 입력해보자.



node_modules 크기 확인하기

- todo list 를 만들기위한 vue.js 프로젝트의 node_modules 크기



100MB 는 기본이다.

샘플 코드 작성하기

- 간단한 express code를 작성하기.
- index.js 생성



```
1 const express = require("express");
2 const app = express();
3 const PORT = 8080;
4
5 app.get("/api/info", (req, res) => {
6   return res.json({
7     name: "jony",
8     job: "tutor",
9   });
10 });
11
12 app.listen(PORT, () => console.log(`this server listening on ${PORT}`));
```

```
1 // express 모듈 가져옴
2 const express = require("express");
...  
3 // express 함수 호출 후 app 객체 생성
4 const app = express();
5 // PORT 상수. 8080 사용 예정
6 const PORT = 8080;
```

코드 분석

- 사용자가 /api/info 로 http 요청을 보내면,
JSON Format 으로 name 과 job 을 응답
- **get**
 - http 요청의 한 종류
 - REST API 챕터에서 배움
- **"/"**
 - API 라우트 경로
다음 장 Routing 에서 배움
- **req, res**
 - 요청객체 (request), 응답객체 (response)
- **return res.json()**
 - JSON Format 으로 응답 객체 res 에 name, job 을 실어서 보낸다.

```
8  app.get("/api/info", (req, res) => {  
9    return res.json({  
10      name: "jony",  
11      job: "tutor",  
12    });  
13  });
```

마지막 줄에서, 서버를 작동시킨다.

- **listen**
 - 서버 요청 대기 상태
- **PORT 8080**
 - 즉, express 서버는 실행과 동시에 8080 포트에서 클라이언트 요청(request) 대기중
- **두번째 파라미터 콜백함수**
 - listen 성공 시 실행. 터미널에 어느 포트에서 서비스되고있는지 안내하는 용도

```
12     app.listen(PORT, () => console.log(`this server listening on ${PORT}`));
```

express 서버 실행하기

- `nodemon ./index.js`

```
C:\Users\mincoding\Desktop\node-test>nodemon index.js
[nodemon] 2.0.16
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
this server listening on 8080
```

8080 포트에 작동중이라는 안내문 확인

구글에서 크롬 웹스토어 검색

The screenshot shows a Google search results page with a dark theme. The search bar at the top contains the query "크롬 웹스토어". Below the search bar, there are several search filters: 전체 (selected), 쇼핑, 동영상, 이미지, 뉴스, and 더보기. To the right of the filters is a "도구" button. The main search results area displays a single result from the Chrome Web Store:

<https://chrome.google.com/webstore> ▾
Chrome 웹 스토어
Chrome 웹 스토어. 확장 프로그램, 테마, 앱으로 데스크톱 컴퓨터에서 Chrome을 맞춤설정합니다. Chrome에 새로운 기능 추가. Chrome에 확장 프로그램을 설치하여 새 ...

테마
브라우저를 맞춤설정할 수 있는 맞춤 브라우저 스킨입니다.

로그인
Chrome에 사용할 유용한 앱, 게임, 확장 프로그램 및 테마를 찾아보 ...

다크 앤 블랙 테마
Chrome에 사용할 유용한 앱, 게임, 확장 프로그램 및 테마를 찾아보 ...

모두 보기
Chrome에 사용할 유용한 앱, 게임, 확장 프로그램 및 테마를 찾아보 ...

[google.com 검색결과 더보기 »](#)

JSON 검색 후, JSON Formatter 설치

The screenshot shows the Chrome Web Store search results for the query "JSON". The search bar at the top left contains "JSON". To the right of the search bar, there is a user profile icon with the email "jaryong.lee@mincoding.co.kr". Below the search bar, a list of extensions is displayed:

- json
- jsonview
- json formatter
- json viewer
- jsonview hateoas
- json beautifier
- json editor

The extension "json formatter" is highlighted with a green ribbon banner at the bottom left of its card. The card for "json formatter" includes the following details:

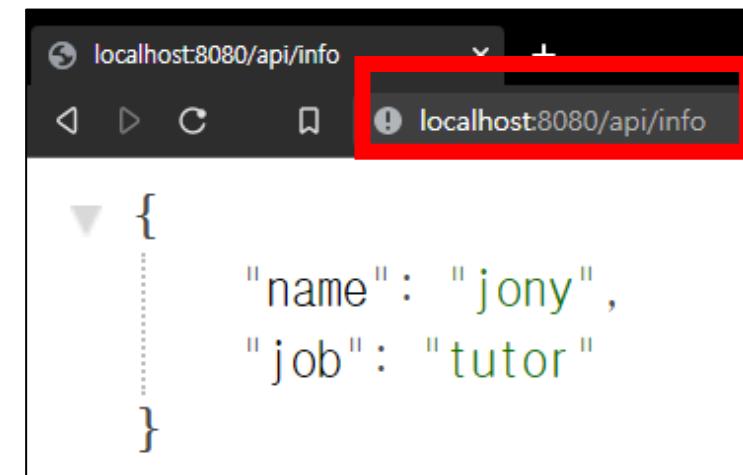
- JSON Formatter**
- Developer: callumlocke.com
- Version: 추천
- Description: Makes JSON easy to read. Open source.
- Rating: ★★★★★ 1,810 개발자 도구

- localhost:8080/api/info

- 접속 시, 작성해둔 객체 리턴
- localhost 내 (local) 컴퓨터 (host)
- /api/host 코드에서 지정한 경로

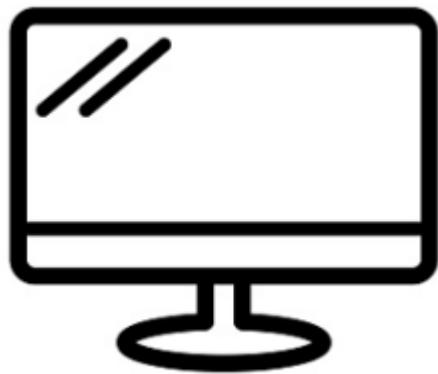
- 이렇게 활용되는 서버를 API 서버라고 한다.

- 활용: 사용자 요청을 받아 SQL DB 접근 후
- 결과를 JSON Format으로 리턴
- (Node.js 수업의 **최종 목표**)

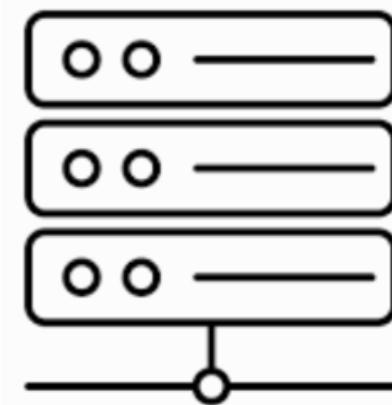
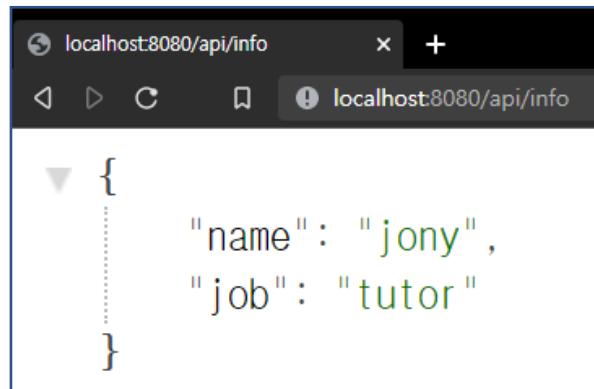
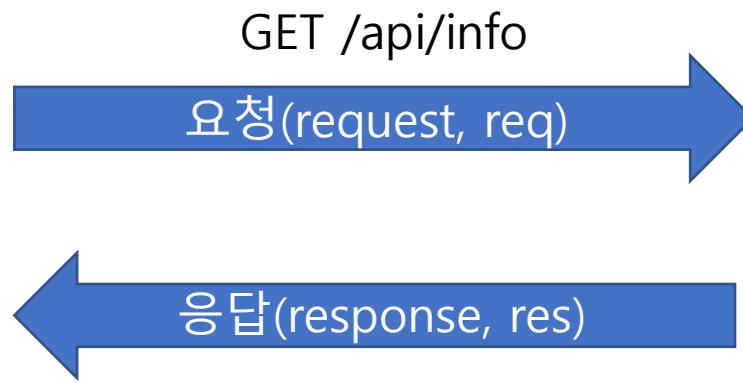


요청과 응답

요청과 응답은 통신의 기본이다.



Client (Frontend)



Server (backend)

express의 req, res 알아보기

- **req, res**

- 각각 요청(req)과 응답(res)에 대한 정보 및, 활용 가능한 메서드를 가진 **거대한 객체**
- 각각 console.log 를 사용해서, 터미널에 출력해보자
- 오른쪽 이미지는 req 객체의 아주 작은 일부분
- 우린 어떻게 활용했는가?
 - res 객체 안에, json 함수를 사용해,
 - 객체를 JSON Format 으로 바꾼 후, 클라이언트로 보냄

```
app.get("/api/info", (req, res) => {
  console.log(req);
  // console.log(res);
  return res.json({
    name: "jony",
    job: "tutor",
  });
});
```

```
<ref *2> IncomingMessage {
  _readableState: ReadableState {
    objectMode: false,
    highWaterMark: 16384,
    buffer: BufferList { head: null,
      length: 0,
      pipes: [],
      flowing: null,
      ended: false,
      endEmitted: false,
      reading: false,
      constructed: true,
      sync: true,
      needReadable: false,
      emittedReadable: false,
      readableListening: false,
      resumeScheduled: false,
      errorEmitted: false,
      emitClose: true,
      autoDestroy: true,
      destroyed: false,
      errored: null,
      closed: false,
    }
  }
}
```

req 객체

클라이언트와 CORS

API 서버에 접속하는 클라이언트를 만들어보자

- 단, 현재 작업 디렉터리 이외의 공간에 만들 것
 - ex) 바탕화면

```
<body>
  <h1>API TEST</h1>
  <button>서버의 데이터 가져오기</button>
  <script src="https://cdn.jsdelivr.net/npm/axios@0.27.2/dist/axios.min.js"></script>
  <script>
    const btn = document.querySelector("button");
    btn.addEventListener("click", async () => {
      try {
        const response = await axios.get("http://localhost:8080/api/info");
        console.log(response.data);
        if (response.data) {
          console.log(response.data);
        }
      } catch (error) {
        console.log(error);
      }
    });
  </script>
</body>
```

axios CDN
(jsdelivr)

버튼 누를 시 동
작

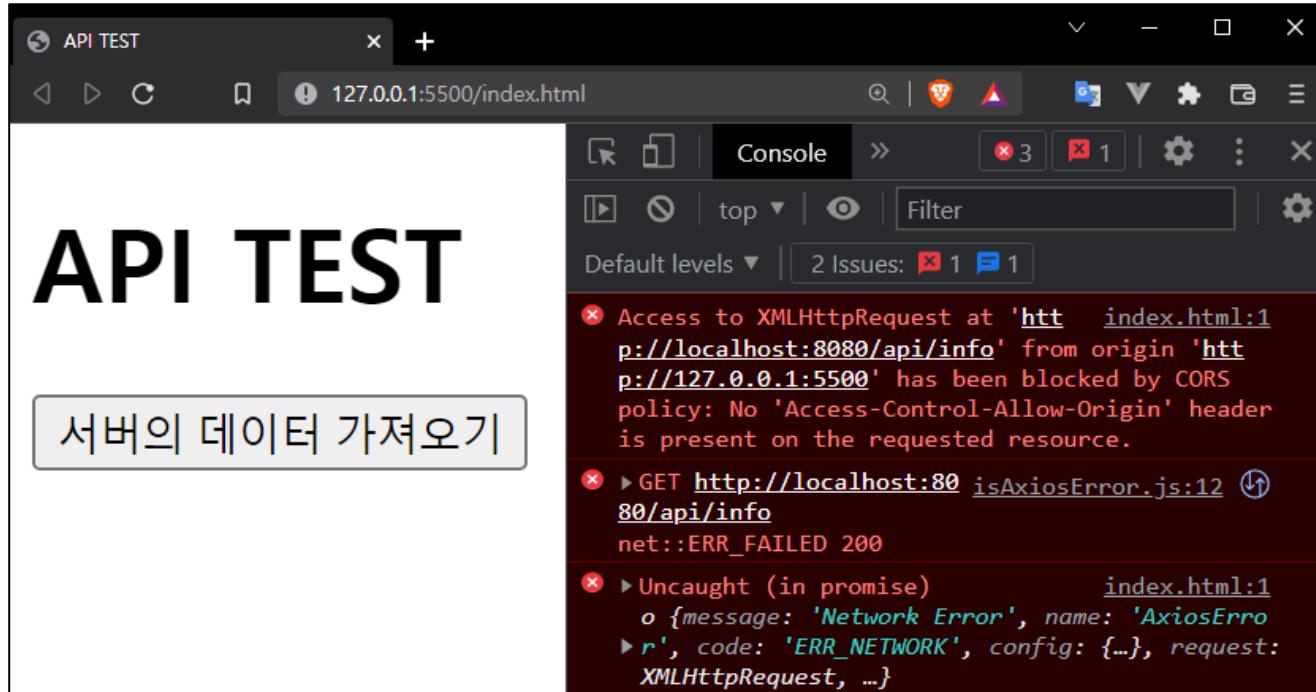
해당 경로에, GET 요청
보냄

- 클라이언트 (프론트엔드)
 - 코드는 바탕화면에 위치
- 서버 (백엔드)
 - 코드는 우리가 만든 node.js 패키지에 위치
- 핵심은, 두 코드가 다른 곳에 위치한다는 것!



클라이언트 실행하기

- live server 로 통신 해보기작동시켜 보았으나... 통신이 되지 않는다.
- 이것은 **CORS** 때문에 발생한다.



현재 live server 포트는 5500 번

다른 출처 리소스 공유

CORS (Cross-Origin Resource Sharing) Policy

- 기본적인 웹 Policy (정책)
- IP 뿐만 아니라 PORT 까지 일치해야만 리소스 공유 가능하다는 정책(규칙)
- 여기서 리소스? 우리 상황에선 JSON
- 즉, JSON을 공유하고자 하는 클라이언트와 서버
 - IP와 PORT가 일치해야 정확하게 통신이 가능하다는 규칙
- 현재 상태 정리
 - 클라이언트
 - Live Server - 5500번 포트
 - 서버
 - Express - 8080번 포트
 - 둘다 localhost인건 동일하나
서로 포트가 다르기 때문에 정책에 위배

- 왜 CORS Policy 를 정해서 우릴 힘들게 할까?

- 북한이나 중국에서 내 웹서버에 요청을 한다면, 신뢰할 수 있는가?
- 서버 개발자는 통신을 허용할 범위를 정해야 한다!

- 해결책

- 서버 코드인 Express 에서, CORS 패키지 설치
 - 특정 IP및 포트를 개방시키기

npm 패키지 cors 설치하기

- 여기서 다운받는 CORS는 정책이 아니라, NPM 패키지 이름이다

cors DT

2.8.5 • Public • Published 4 years ago

[Readme](#) [Explore BETA](#) [2 Dependencies](#) [10,510 Dependents](#) [34 Versions](#)

cors

npm v2.8.5 downloads 36M/month build passing coverage 100%

CORS is a node.js package for providing a [Connect/Express](#) middleware that can be used to enable [CORS](#) with various options.

Follow me (@troygoode) on Twitter!

- Installation
- Usage
 - Simple Usage
 - Enable CORS for a Single Route
 - Configuring CORS
 - Configuring CORS Asynchronously
 - Enabling CORS Pre-Flight

Install

```
> npm i cors
```

Repository github.com/expressjs/cors

Homepage github.com/expressjs/cors#readme

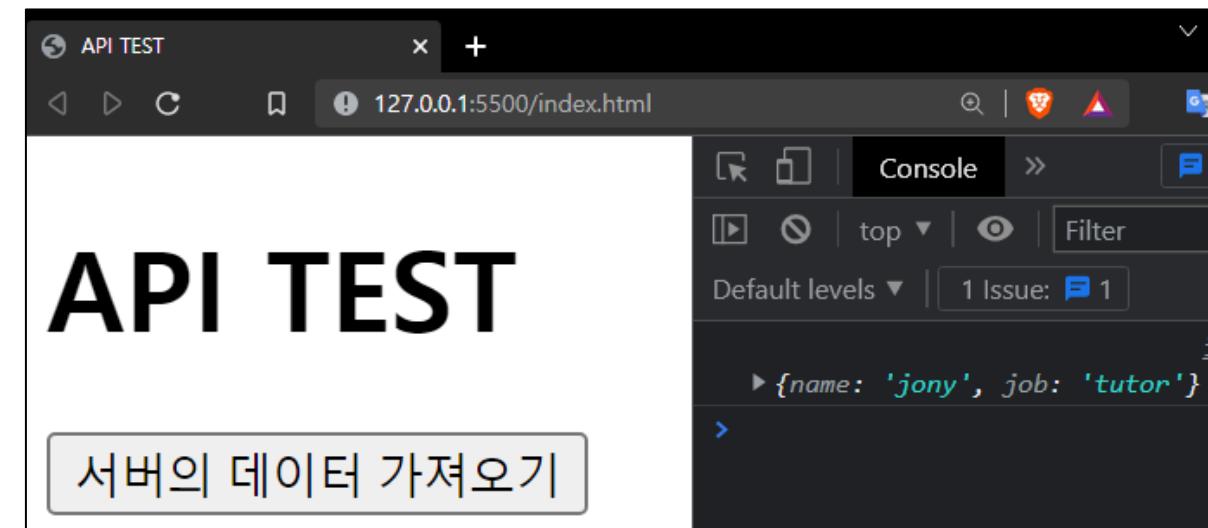
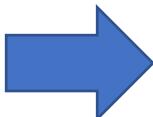
Weekly Downloads 8,223,991

Version 2.8.5 License MIT

기존 코드에 cors 관련 코드 추가하기

- `const cors = require("cors");`
 - cors 패키지 가져오기
- `app.use(cors());`
 - 해당 cors를 사용
 - 바로 app.use로 cors()를 사용하게 되면 모든 접속이 허용되는 방식
 - 추가 옵션을 부여해서 제어하는 것 또한 가능

```
1 const express = require("express");
2 const app = express();
3 const PORT = 8080;
4
5 const cors = require("cors");
6 app.use(cors());
```

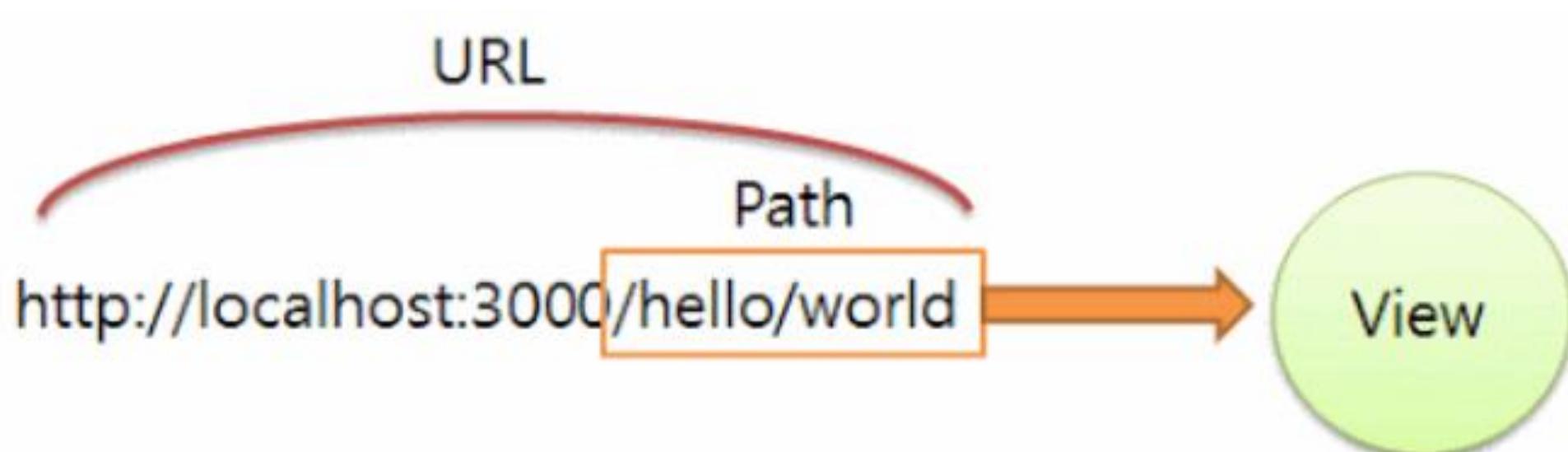


cors 적용 후 통신에 성공

Routing

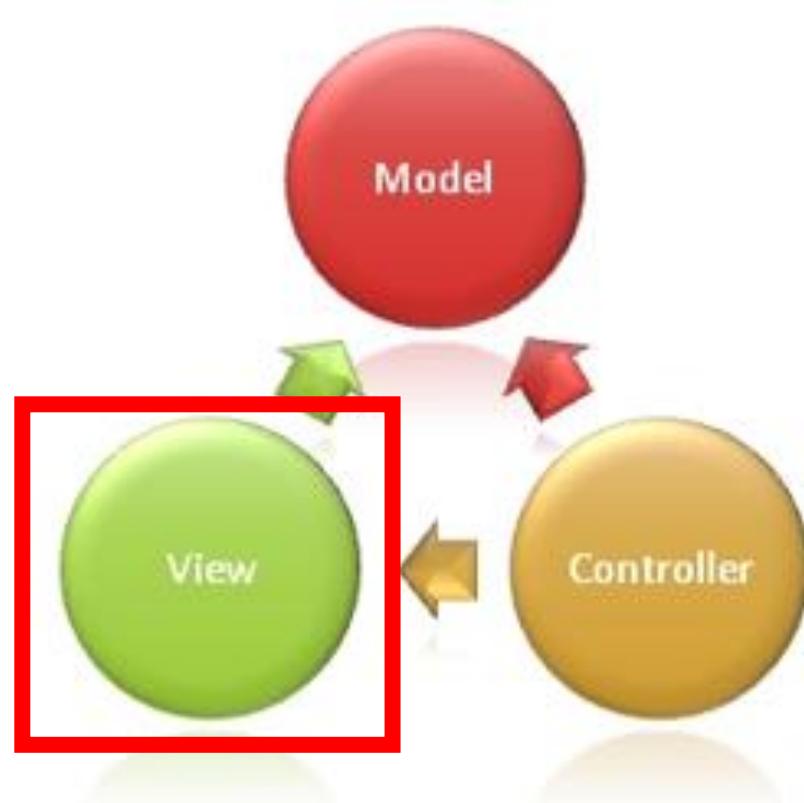
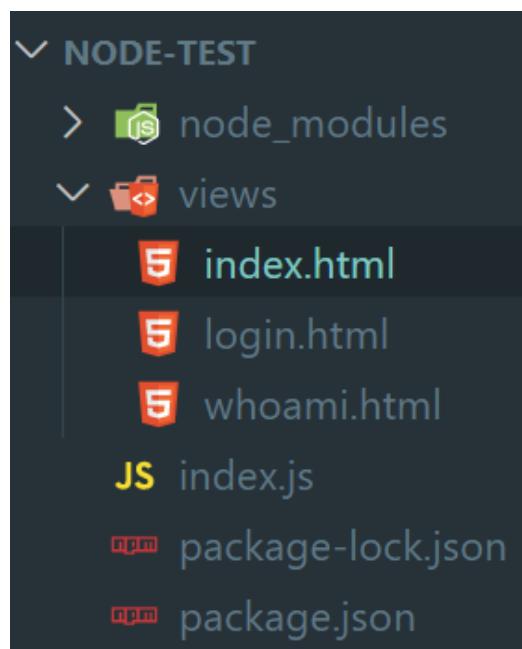
URL 라우팅

사용자가 접근한 경로에 따라서 그에 맞는 메소드를 호출해주는 기능



node.js에서 view 디렉토리 생성

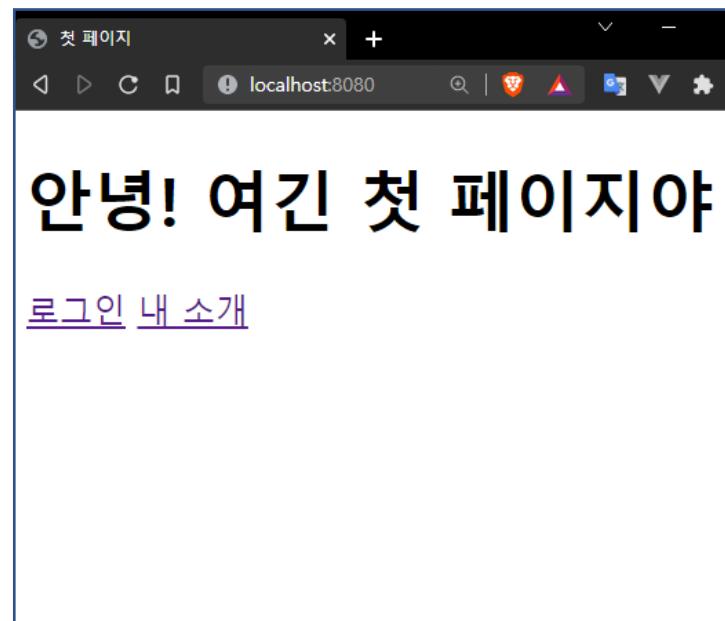
- 클라이언트 코드가 들어가는 디렉토리는 관습적으로 views 라고 이름 짓는다.



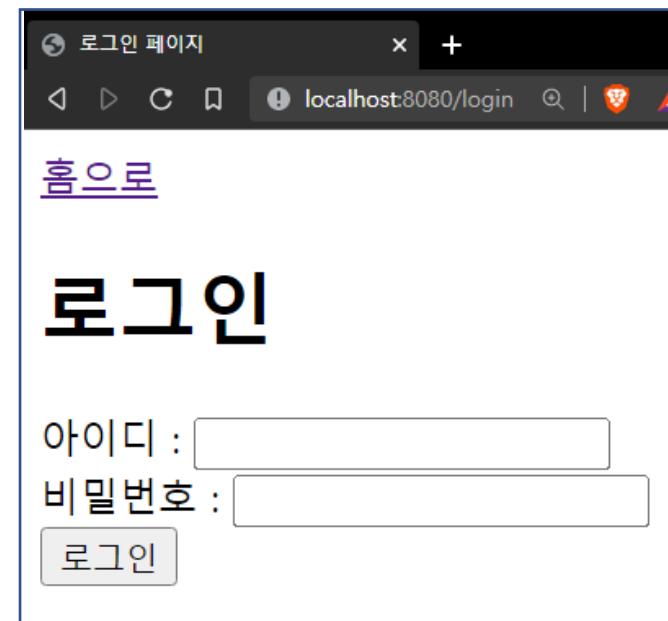
페이지 생성하기

Confidential

index.html



login.html



whoami.html



```
<a href="/login">로그인</a>
<a href="/whoami">내 소개</a>
```

```
<a href="/">홈으로</a>
```

주요 메서드 정리

- **sendFile**
 - res 에서, 파일 전송 시 사용하는 메서드. 여기선 html 을 보낼 용도
 - 상대 경로가 아닌 절대 경로로만 동작한다.
- **_dirname**
 - 현재 기기에서 작동중인 서버의 경로를 의미하는 상수
 - 왜? 각각의 기기마다 서버의 경로가 전부 다르다.(또한 sendFile은 절대 경로로만 동작)
 - 따라서 현재 서버의 경로 + 파일의 위치 조합으로 절대경로를 만들어 조합.
 - __dirname 과, 각각의 html 파일 경로를 합쳐 사용

```
app.get("/", (req, res) => {
  return res.sendFile(__dirname + "/views/index.html");
});

app.get("/login", (req, res) => {
  return res.sendFile(__dirname + "/views/login.html");
}

app.get("/whoami", (req, res) => {
  return res.sendFile(__dirname + "/views/whoami.html");
})
```

app.get("/라우터주소", 콜백함수)

- 해당 라우터 주소에 get 방식으로 도착을 하면 콜백 함수를 실행
- /에 도착하면 index.html 파일을 보여준다.
- /login에 도착하면 login.html파일을 보여준다.
- /whoami에 도착하면 whoami.html을 보여준다.

```
app.get("/", (req, res) => {
  return res.sendFile(__dirname + "/views/index.html");
});

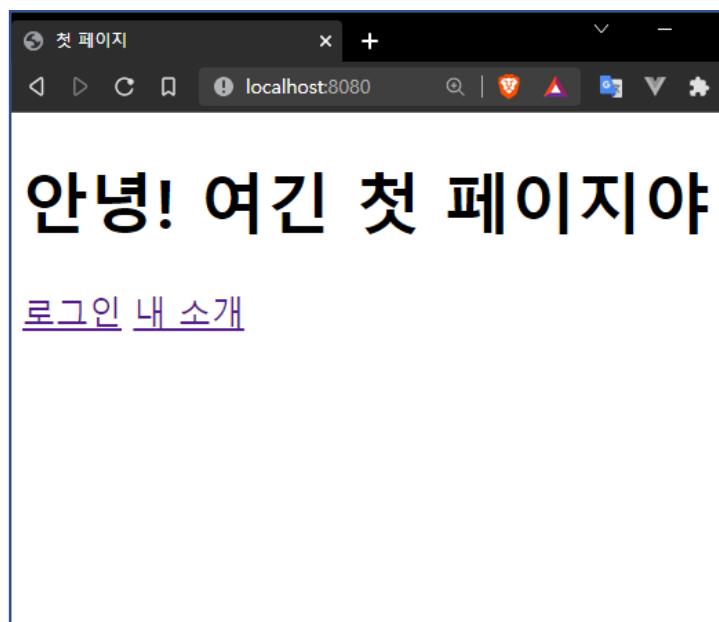
app.get("/login", (req, res) => {
  return res.sendFile(__dirname + "/views/login.html");
}

app.get("/whoami", (req, res) => {
  return res.sendFile(__dirname + "/views/whoami.html");
})
```

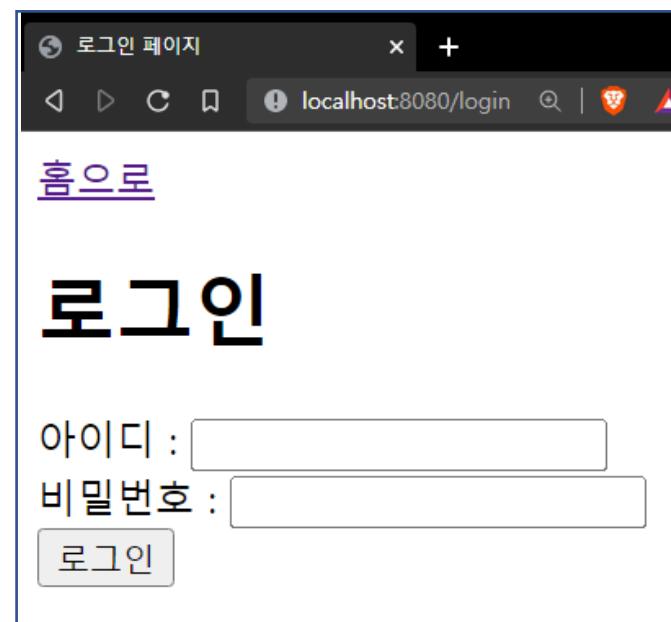
유의사항

- url에 써준 내용들은 파일의 경로가 아니라 요청의 경로
- /로 요청을 보내면 app.get("/", 콜백함수) / 부분에 요청을 보내는것
- /login으로 요청을 보내면 app.get("/login", 콜백함수) /login 부분에 요청을 보내는것

/



/login



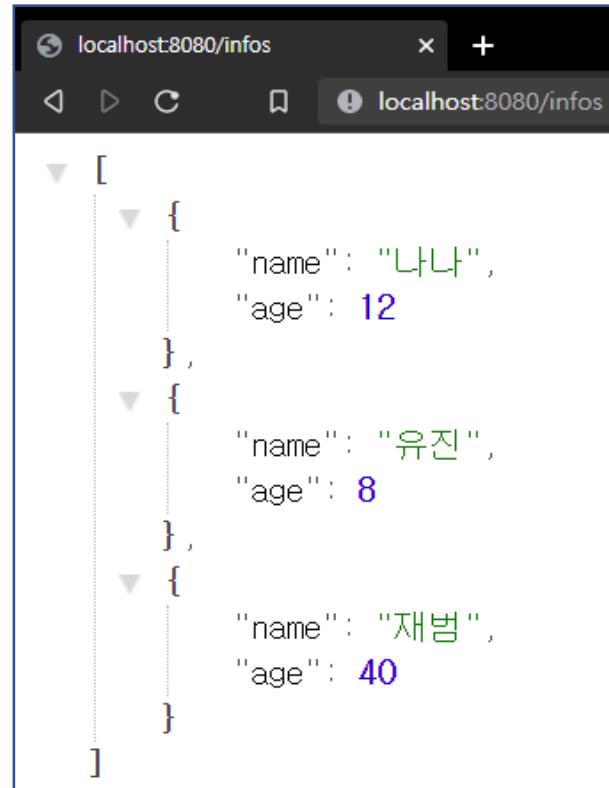
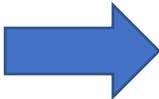
/whoami



이번엔 파일이 아니라 JSON을 리턴해보기

- /infos에 요청을 보낼시에 return res.json으로 배열을 리턴

```
13 app.use(express.json());
14
15 const infos = [
16   {
17     name: "나나",
18     age: 12,
19   },
20   {
21     name: "유진",
22     age: 8,
23   },
24   {
25     name: "재범",
26     age: 40
27   },
28 ]
29
30 app.get("/infos", (req, res) => {
31   return res.json(infos);
32 })
```



배열 안에 동일한 타입의 객체를 정의

/infos 접속 시, res.json() 을 사용해
배열을 JSON 형식으로 리턴

해당 경로 접속시 명시된 리턴을 수행

- **/infos/names**
 - infos 중 이름만 파싱한 배열을 JSON 리턴
 - map 활용해보기
 - `Array.map((element) => { return element.name })`
- **/infos/ages**
 - infos 중 나이만 파싱한 배열을 JSON 리턴
- **/infos/0**
 - infos 배열의 0번째 리턴
- **/infos/1**
 - infos 배열의 1번째 리턴
- **/infos/2**
 - infos 배열의 2번째 리턴

```
const infos = [  
  {  
    name: "나나",  
    age: 12,  
  },  
  {  
    name: "유진",  
    age: 8,  
  },  
  {  
    name: "재범",  
    age: 40,  
  },  
];
```

```
[  
  "나나",  
  "유진",  
  "재범"  
]
```

/infos/names

```
{  
  "name": "유진",  
  "age": 8  
}
```

/infos/1

9장. Express 와 MySQL 연동

챕터의 포인트

- MySQL 설치 (AWS)
- mysql2 패키지 설치

MySQL 설치 (AWS)

DB 서버 구축하기

- 이번엔 Windows 가 아니라,
AWS Instance 에 DB 서버를 구성하여
MySQL 원격 접속을 시도해보자.



MobaXterm에 접속

- AWS 서버에 ssh 접속
- \$ sudo apt update
- \$ sudo apt install mysql-server
- \$ mysql --version

```
ubuntu@ip-172-31-45-229:~$ mysql --version
mysql Ver 14.14 Distrib 5.7.38, for Linux (x86_64) using EditLine wrapper
```

\$ sudo mysql

- 새로운 관리자 계정을 만들기 위해 mysql root 계정으로 로그인

```
ubuntu@ip-172-31-40-222:~$ sudo mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 8
Server version: 5.7.38-0ubuntu0.18.04.1 (Ubuntu)

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> 
```

\$ 명령어 입력

- **create user 'ssafy'@'%' IDENTIFIED BY 'ssafy1234';**
 - 아이디 ssafy 비밀번호 ssafy1234로 계정 생성
- **grant all privileges on *.* to 'ssafy'@'%' with grant option;**
 - ssafy 계정에 권한 부여
 - ON *.* TO 에 * 은 DB 이름, 뒤에 * 은 권한 내용
 - % 는 모든 외부 IP 를 뜻한다. 즉, 전세계에서 연결 허용

```
mysql> create user 'ssafy'@'%' IDENTIFIED BY 'ssafy1234';
Query OK, 0 rows affected (0.02 sec)

mysql> grant all privileges on *.* to 'ssafy'@'%' with grant option;
Query OK, 0 rows affected (0.00 sec)

mysql>
```

관리자 계정 로그인

- mysql에서 exit 입력후 ubuntu shell로 돌아가기
- 방금 만든 관리자 계정으로 로그인
 - sudo mysql -u 아이디 -p
 - 작성후 비밀번호
 - 아이디 ssafy
 - 비밀번호 ssafy1234

```
ubuntu@ip-172-31-43-222:~$  
ubuntu@ip-172-31-43-222:~$ mysql -u ssafy -p  
Enter password:  
Welcome to the MySQL monitor. Commands end with ; or \g.  
Your MySQL connection id is 12  
Server version: 8.0.32-0ubuntu0.20.04.2 (Ubuntu)  
  
Copyright (c) 2000, 2023, Oracle and/or its affiliates.  
  
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.  
  
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.  
mysql> █
```

DB 생성하기

- **CREATE SCHEMA 디비이름 DEFAULT CHARACTER SET utf8mb4**

- CREATE SCHEMA jony DEFAULT CHARACTER SET utf8mb4;
- DB 를 생성하는데, 한글이 안 깨지도록 utf8 인코딩을 적용한 DB 생성
- DB 생성 후 show DATABASES로 DB가 생성되었는지 조회
- use jony를 사용해서 해당 데이터베이스를 사용하도록 명시 (workbench의 데이터베이스 더블클릭과 동일하다)

```
mysql> CREATE SCHEMA jony DEFAULT CHARACTER SET utf8mb4;
Query OK, 1 row affected (0.00 sec)

mysql> show schemas;
+-----+
| Database |
+-----+
| information_schema |
| jony |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.00 sec)
```

테이블 생성하기

- menus 테이블 생성

- CREATE TABLE menus (
menu_id INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
menu_name VARCHAR(20) NOT NULL,
menu_description TEXT NOT NULL
);

- 더미데이터 집어넣기

- INSERT INTO menus
(menu_name, menu_description)
VALUES
("복숭아 아이스티", "내 입안 복숭아향 가득");
 - 위 방식으로 두개 추가 ("카페라떼", "Latte is horse");
 - ("아이스 아메리카노", "여름엔 아아가 진리");

데이터가 잘 들어갔는지 확인해보자

- **show tables;**
- **select * from menus;**
- 확인 되었으면 **exit**로 DB 접속 종료

```
mysql> show tables;
+-----+
| Tables_in_jony |
+-----+
| menus          |
+-----+
1 row in set (0.00 sec)

mysql> select * from menus;
+-----+-----+-----+
| menu_id | menu_name           | menu_description      |
+-----+-----+-----+
| 1       | 아이스 아메리카노   | 여름엔 아아가 진리  |
| 2       | 카페라떼             | Latte is horse      |
| 3       | 복숭아 아이스티     | 내 입안 복숭아향 가득|
+-----+-----+-----+
3 rows in set (0.01 sec)
```

Listen IP 대역폭 변경

- 현재 MySQL 은 오로지 localhost 에서만 사용하도록 설정되어있다.
- 모든 IP 에서 원격접속할 수 있도록 바꿔보자.
- `sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf`

```
ubuntu@ip-172-31-40-222:~$ sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf █
```

mysqld.cnf 변경내용

- **bind-address = 127.0.0.1**
 - bind-address = 0.0.0.0로 변경
 - 127.0.0.1(localhost)에서만 접근허용 됐던 DB를 모든 곳에서 접속이 가능하게 변경시켜주는것
 - 수정 후 Ctrl + O Enter
 - Ctrl + X로 저장후 나가기
 - cat /etc/mysql/mysql.conf.d/mysqld.cnf | grep bind
 - 명령어로 수정된 bind-address 파트만 조회

```
#  
# Instead of skip-networking the default is now to listen only on  
# localhost which is more compatible and is not less secure.  
# bind-address          = 127.0.0.1  
bind-address          = 0.0.0.0
```

```
ubuntu@ip-172-31-30-8:~$ cat /etc/mysql/mysql.conf.d/mysqld.cnf | grep bind  
bind-address          = 0.0.0.0
```

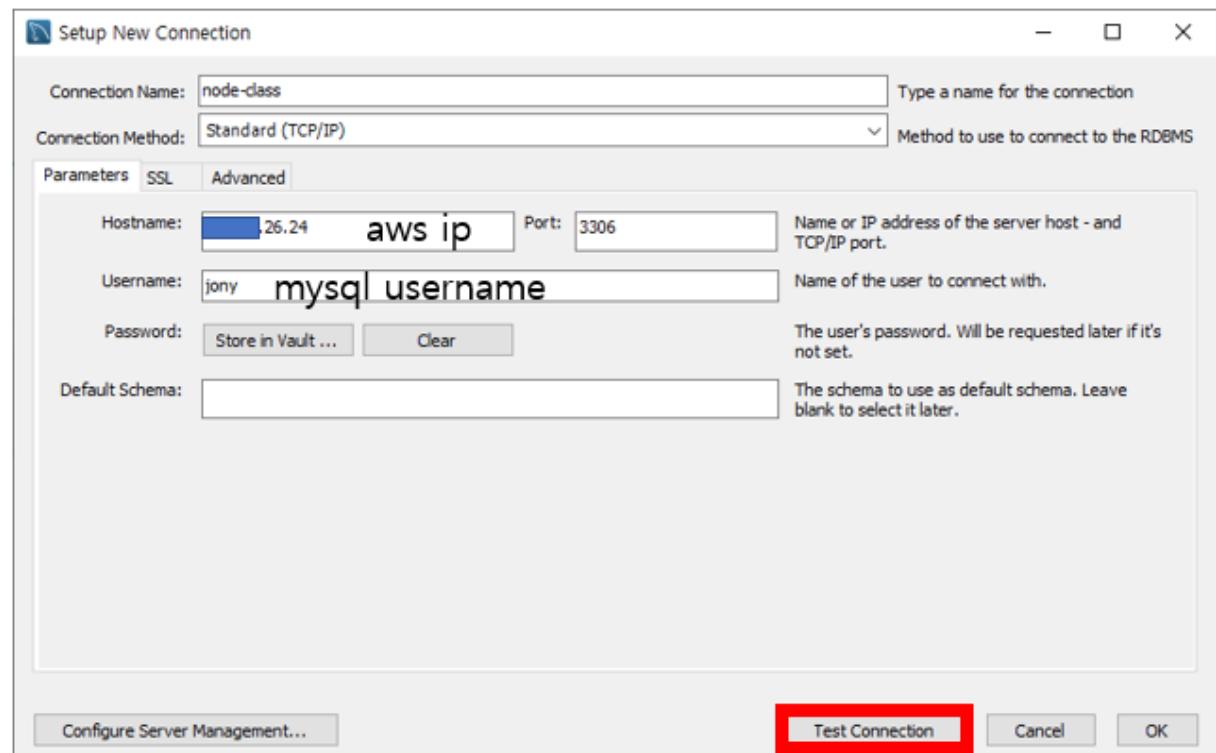
설정 내용 적용하기

- 변경된 내용을 적용하기 위해 MySQL 서버를 재시작하는 작업이 필요하다
- `sudo service mysql restart`
- 재시작 까지 진행하면 이제 외부에서 DB 접속을 위한 작업은 끝이난다.

```
ubuntu@ip-172-31-40-222:~$ sudo service mysql restart
ubuntu@ip-172-31-40-222:~$ █
```

MySQL Workbench 기동

- DB 외부 접속 셋팅이 끝났기 때문에 Window Workbench에서도 접근이 가능
 - hostname : AWS의 IP 주소
 - username 외부 접속을 위해 가입한 아이디
 - 아이디 ssafy
 - 비밀번호 ssafy_8_A



눌렀을 때 비밀번호 창 뜨면 성공

AWS DB서버 접속 완료

- 방금 만든 데이터베이스 jony 와, menus 더미데이터 확인하기
- 이로서, 원격에서 MySQL 을 외부에서 접근하는 법을 마쳤다.

The screenshot shows the MySQL Workbench interface with a successful connection to an AWS database. The left sidebar displays the Navigator with the Schemas section expanded, showing the 'jony' schema containing a 'menus' table, along with 'Views', 'Stored Procedures', and 'Functions'. Below the schemas is the 'sys' schema. The middle section shows the 'Query 1' tab with the query `SELECT * FROM jony.menus;` and its results. The results grid displays four rows of menu data:

	menu_id	menu_name	menu_description	menu_img_link
▶	1	아이스 아메리카노	여름엔 아아가 진리	/menus/ice-americano.jpg
▶	2	카페라떼	Latte is horse	/menus/cafe-latte.jpg
▶	3	복숭아 아이스티	내 입안 복숭아향 가득	/menus/peach-icetea.jpg
*	NULL	NULL	NULL	NULL

mysql2 패키지 설치

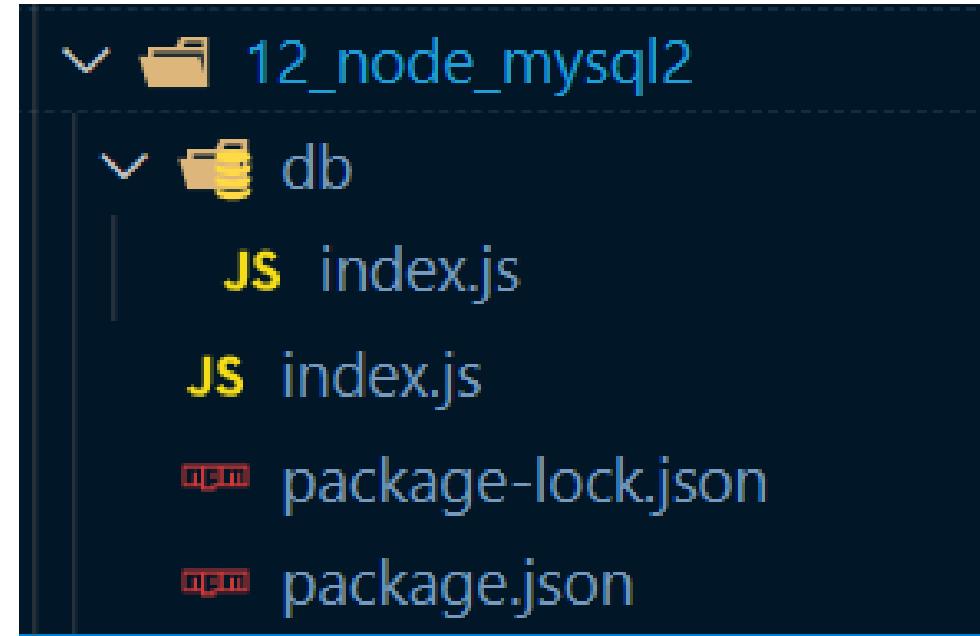
mysql2를 사용하는 이유

- node에서는 기본적으로 mysql에 접근하기 위해서는 라이브러리 사용이 필요
- mysql2라는 라이브러리를 사용한다.
 - 왜 mysql1 이 아니라 2를 사용할까?
 - mysql2부터 promise 방식이 사용 가능 및 개편된 버전
 - npm i mysql2

```
"dependencies": {  
    "cors": "^2.8.5",  
    "express": "^4.18.1",  
    "mysql2": "^2.3.3"  
}
```

폴더 구조 확인하기

- npm init 명령어 실행
- npm i mysql2 express cors
- db 폴더 생성 후 그 안에 index.js 생성



db 폴더 내 index.js를 수정

- **host**: 접속하기 위한 ip 주소 작성
 - localhost의 db에 접근하기 위해서는 localhost 나 127.0.0.1 을 작성
 - 외부 DB를 생성했기 때문에 AWS의 주소를 작성
- **user**
 - DB에 접근하기 위해 생성해둔 ID
 - ssafy
- **password**
 - DB에 접근하기 위해 생성해둔 PASSWORD
 - ssafy_8_A
- **database**
 - 접근하기 위한 DB(스키마) 의 이름

```
const mysql = require("mysql2/promise");

const pool = mysql.createPool({
    // aws ip
    host: "3.39.183.206",
    // mysql username
    user: "ssafy",
    // mysql user password
    password: "ssafy_8_A",
    // db name
    database: "jony",
    waitForConnections: true,
    connectionLimit: 10,
    queueLimit: 0,
});

module.exports = { pool };
```

DB 접속 코드 작성

- db 폴더에는 index.js로 서버 접속을 위한 셋팅이 되어있다.
- 해당 부분에서 pool 부분만 사용하기 위해 가져온 코드

- const { pool } = require('./db')
 - db 폴더의 index.js 의 객체에서 pool 내용만 따로 가져와서 사용하겠다.
 - const db = require('./db');
 - const pool = db.pool 과 같다

```
1  const express = require("express");
2  const { pool } = require("./db");
3
4  const app = express();
5  const PORT = 8080;
```

DB 접속 코드 작성

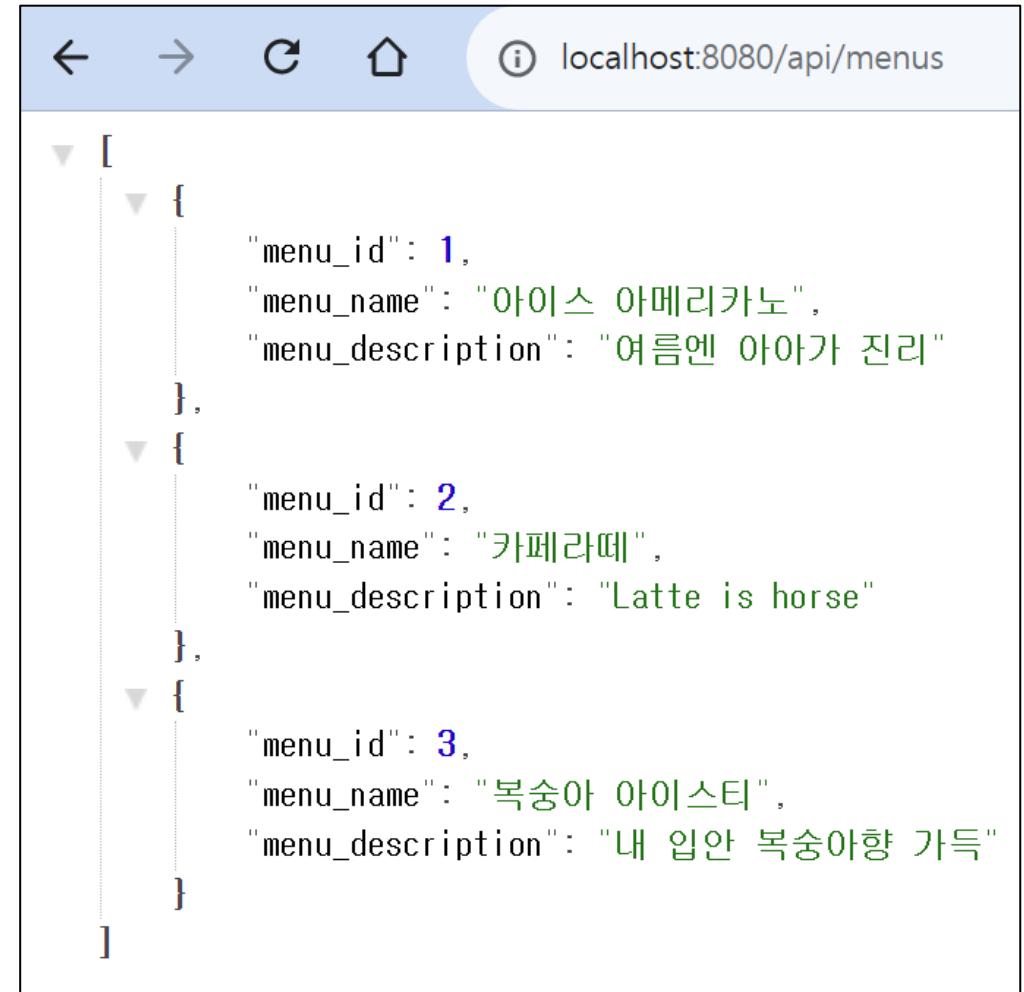
- **async await 활용**

- mysql2는 promise 형식으로 값을 리턴하기 때문에 async/await 활용이 가능
- pool.query 안에 SQL 쿼리문을 작성한다.
- 작성후 첫번째 배열을 리턴

```
app.get("/api/menus", async (req, res) => {
  try {
    const data = await pool.query("SELECT * FROM menus");
    if (data[0]) {
      return res.json(data[0]);
    }
  } catch (error) {
    return res.json(error);
  }
});
```

결과 확인

- localhost:8080/api/menus 접근하기
- 해당 정보를 리턴한다면 성공

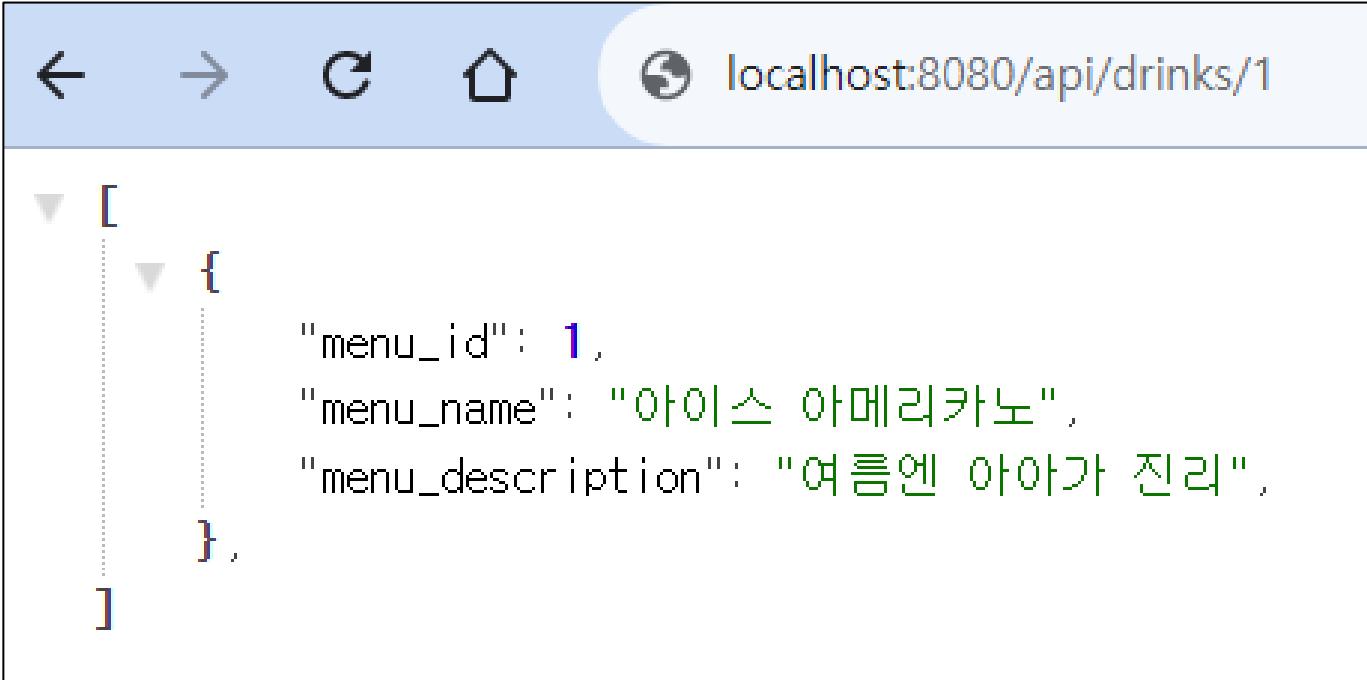


A screenshot of a web browser window displaying a JSON array of menu items. The URL in the address bar is "localhost:8080/api/menus". The JSON data is as follows:

```
[{"menu_id": 1, "menu_name": "아이스 아메리카노", "menu_description": "여름엔 아아가 진리"}, {"menu_id": 2, "menu_name": "카페라떼", "menu_description": "Latte is horse"}, {"menu_id": 3, "menu_name": "복숭아 아이스티", "menu_description": "내 입안 복숭아향 가득"}]
```

DB 연동 다시 진행하기

- 처음부터 DB 연동까지 다시 진행한다.
- app.get("/api/drinks/1") 요청시 id가 1인 데이터값만 가져올수 있도록 하기



A screenshot of a web browser window. The address bar shows "localhost:8080/api/drinks/1". The page content displays a JSON object:

```
[{"menu_id": 1, "menu_name": "아이스 아메리카노", "menu_description": "여름엔 아아가 진리"}, ]
```

내일
방송에서
만나요!

삼성 청년 SW 아카데미