[모음] 복습 도우미

목차

웹 기초 동작원리 - 서버/클라이언트/웹의 동작 개념
1주차 - HTML, CSS, Javascript
2주차 - Javascript, jQuery, Ajax
3주차 - Python, MongoDB
4주차 - Flaks , API 만들고 사용하기
5주차 - Flaks , API 만들고 사용하기 복습
6주차 - 클라우드 구입해 사용해보기
7주차 - 클라우드 서버에 mongoDB 설치하기, 포트 포워딩
8주차 - 도메인 연결하기, 웹사이트 정보 og 태그 넣기

▼ PDF로 다운받기

웹 기초 동작원리 - 서버/클라이언트/웹의 동작 개념

▼ 1. 웹을 데이터 관점으로 바라보기

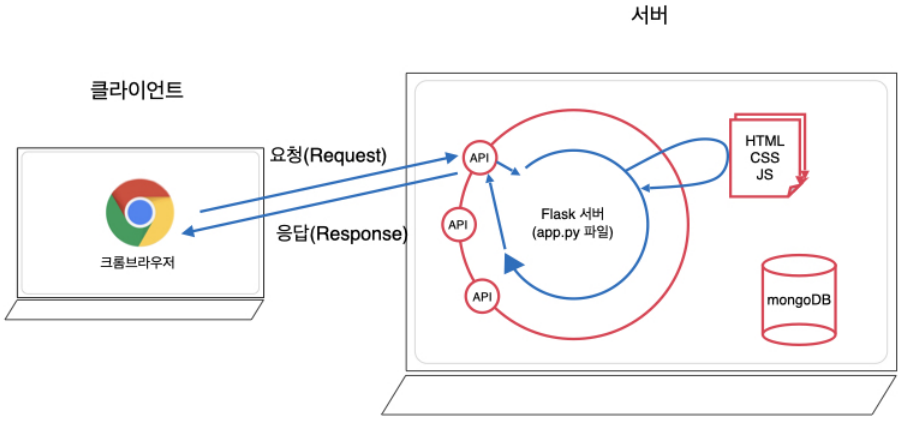
- IT라는 단어가 익숙하신가요? 바로 Information Technolgy 의 약자. 즉, 정보 기술 입니다.
- 웹은 정보를 빠르게 교환하기 위해 태어났어요. 각 웹 페이지 하나 하나는 정보를 담은 문서(Document)입니다. 결국, 화면은 정보(데이터)를 표현한 것이지요.
- 데이터가 우리 눈에 보이도록 화면을 만드는 것이 프론트엔드(Front-end)의 역할이고,
- 데이터를 관리(전송, 저장, 수집, 가공)하는 것이 백엔드(Back-end)의 역할입니다.
- 이제 구체적으로 웹의 기초 동작 원리를 살펴볼까요?

▼ 2. 웹 기초 동작원리 (HTML을 받는 경우)

네! 우리가 보는 웹페이지는 모두 서버에서 미리 준비해두었던 것을 "받아서", "그려주는" 것입니다. 즉, 브라우저가 하는 일은 1) 요청을 보내고, 2) 받은 화면 파일(HTML 파일)을 그려주는 일 뿐이죠.

근데, 1)은 어디에 요청을 보내냐구요? 좋은 질문입니다. 서버가 만들어 놓은 "**API**"에 미리 정해진 **약속**대로 요청을 보내는 것이랍니다.

예) https://naver.com/
→ 이것은 "naver.com"이라는 이름의 서버에 있는, "/" 창구에 요청을 보낸 것!



- 웹은 **HTTP**(HyperText Transfer **Protocol**)라는 통신 규약(**Protocol**, 서로 지키도록 협의한 **규칙**)을 따른답니다. url에서 `http://` 가 바로 HTTP 라는 통신 규약을 따른다는 표시에요.
- HTTP에서 **요청(Request)하는 쪽을 클라이언트**, 요청을 처리해 그에 맞게 **응답(Response)하는 쪽을 서버**라고 합니다.

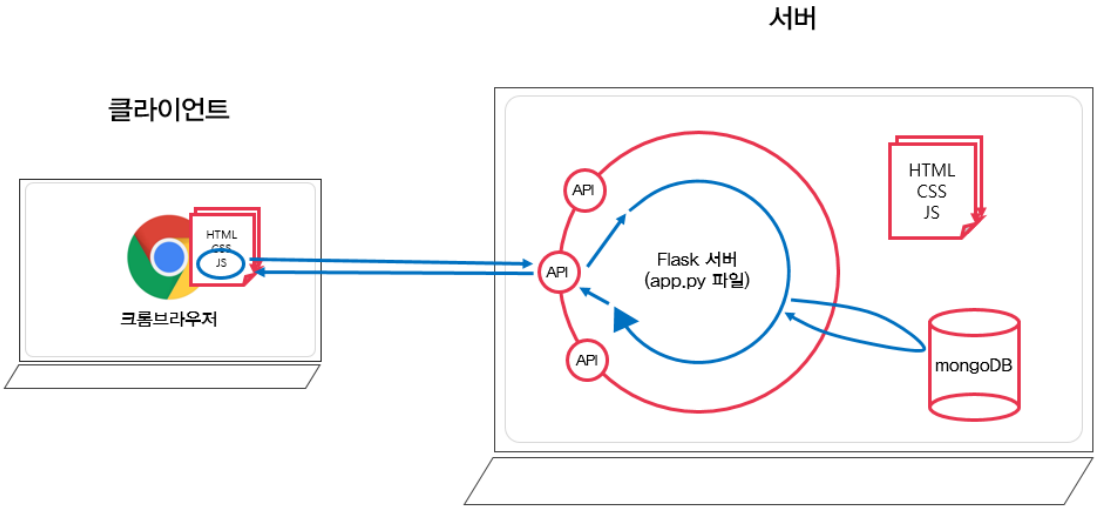
▼ 3. 웹 기초 동작원리 (데이터만 받는 경우)

앗, 그럼 항상 서버는 요청에 응답해서 이렇게 화면 파일(HTML 파일)만 보내주냐구요?
아뇨! 데이터만 보내줄 때가 더~ 많아요.

사실 화면 파일(HTML 파일)도 줄글로 쓰면 이게 다 '데이터' 아닌가요?

자, 콘서트 티켓을 예매하고 있는 상황을 상상해봅시다!
좌석이 찰 때마다 예매 페이지가 새로고침 되면 난감하겠죠ㅠ?

이럴 때! 서버에서 데이터만 받아서 데이터 내용만 바꿔주게 된답니다.



👉 데이터만 보내줄 경우는, 이렇게 생겼어요!
(소곤소곤) 이렇게 데이터를 표현하는 것을 **JSON** 형식이라고 한답니다.

openapi.seoul.go.kr:8088/6d4d7 ×

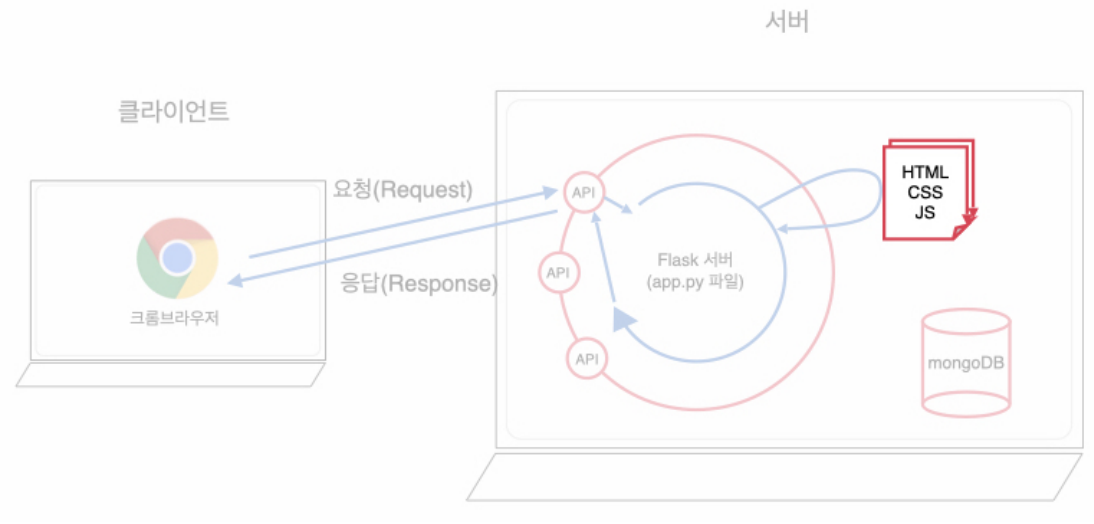
주의 요함 | openapi.seoul.go.kr:8088/6d4d776b466c656533356a4b4b5872/json/RealtimeCityAir/1/99


```
{
  - RealtimeCityAir: {
        list_total_count: 25,
      - RESULT: {
            CODE: "INFO-000",
            MESSAGE: "정상 처리되었습니다"
        },
      - row: [
          - {
                MSRDT: "202004241900",
                MSRRGN_NM: "도심권",
                MSRSTE_NM: "중구",
                PM10: 44,
                PM25: 20,
                O3: 0.039,
                NO2: 0.02,
                CO: 0.4,
                SO2: 0.003,
                IDEX_NM: "보통",
                IDEX_MVL: 59,
                ARPLT_MAIN: "PM10"
            },
          - {
                MSRDT: "202004241900",
```

▼ 4. 웹 기초 동작원리로 보는 1~5주차 학습 순서!

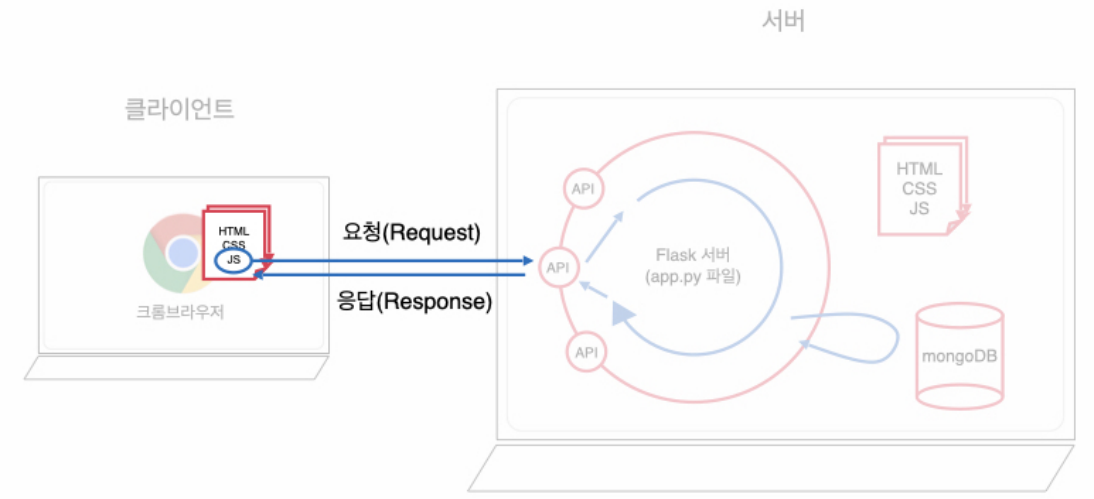
▼ 1주차: HTML, CSS, JavaScript(기초)

> HTML과 CSS를 배우는 날! 즉, 4주차에 클라이언트에 보내줄 HTML파일을 미리 만들어 두는 과정입니다. + 또, 2주차에 JavaScript(줄여서 JS)를 능숙하게 다루기 위해서, 문법을 먼저 조금 배워둘게요!



▼ 2주차: JS(응용), jQuery, Ajax

> 서버가 이미 HTML파일을 보내주었다고 가정하고, JS로 서버에 데이터를 요청하고, 응답받은 데이터를 처리하는 방법을 배워볼거예요

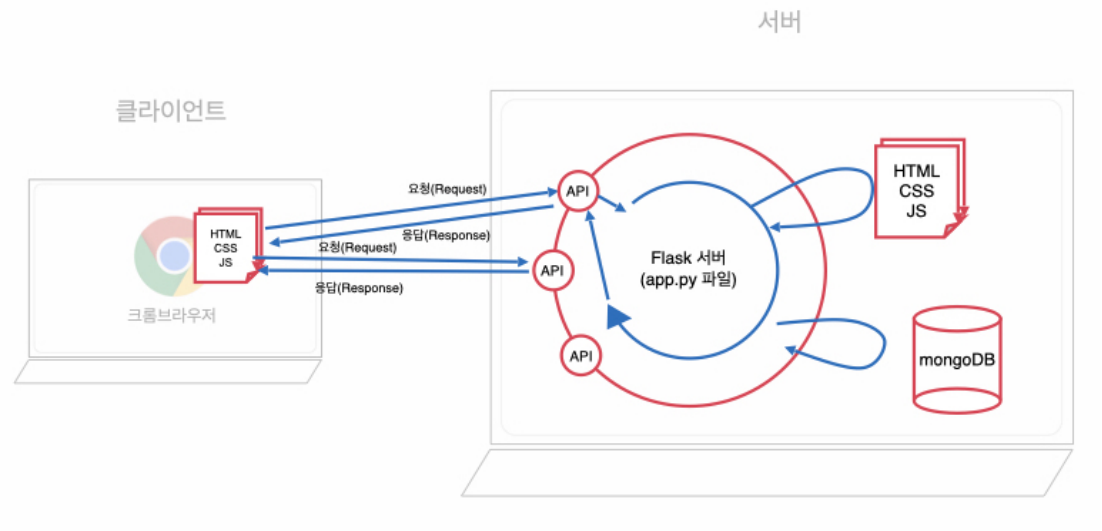


▼ 3주차: Python, 웹 스크래핑(크롤링), mongoDB

드디어 'Python'을 배울거예요. 먼저 문법을 연습하고, 라이브러리를 활용하여 네이버 영화목록을 쫙 가져와보겠습니다. (기대되죠!)
+
그리고, 우리의 인생 첫 데이터베이스. mongoDB를 다뤄볼게요!

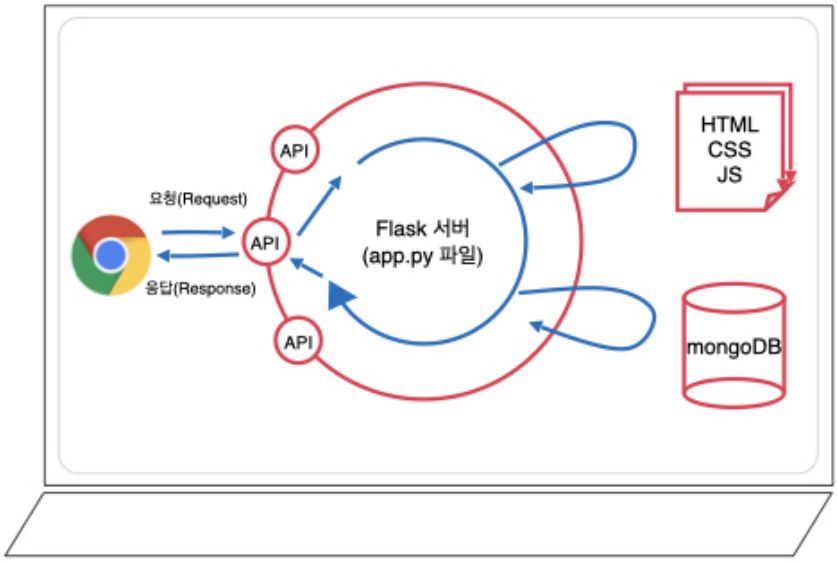
▼ 4주차: Flask, 미니프로젝트1 - 모두의책리뷰

서버(백엔드)를 만들어봅니다! HTML과 mongoDB까지 연동하여,
미니프로젝트1를 완성해보죠! 굉장히 재미있을 거예요!



나중에 또 이야기하겠지만 헷갈리면 안되는 것!
우리는 컴퓨터가 한 대 잖아요... 그래서 같은 컴퓨터에다 서버도 만들고, 요청도 할 거예요. 즉, 클라이언트 = 서버가 되는 것이죠.

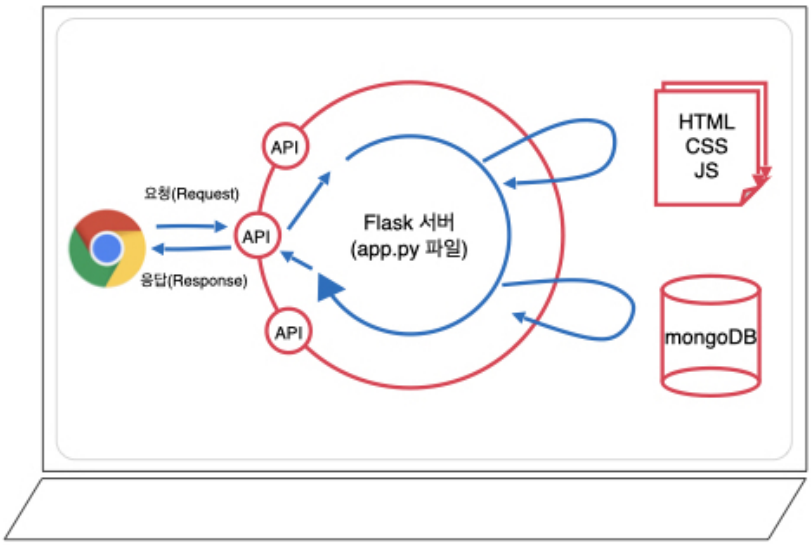
이것을 바로 "로컬 개발환경"이라고 한답니다! 그림으로 보면, 대략 이렇습니다.



▼ 5주차: 미니프로젝트2-나홀로 메모장, 미니프로젝트3 - 페이보릿무비스타

아직 익숙해지지 않았을 여러분을 위해! 같은 난이도의 유사한 두 개의 프로젝트를 더 진행하며 머릿속의 퍼즐을 맞출 예정입니다.

여기까지 배웠다면, 이제 6~8주차 프로젝트 준비 완료!!



x 2개 더!

🧩1주차 - HTML, CSS, Javascript

▼ 👨‍🏫수업 목표

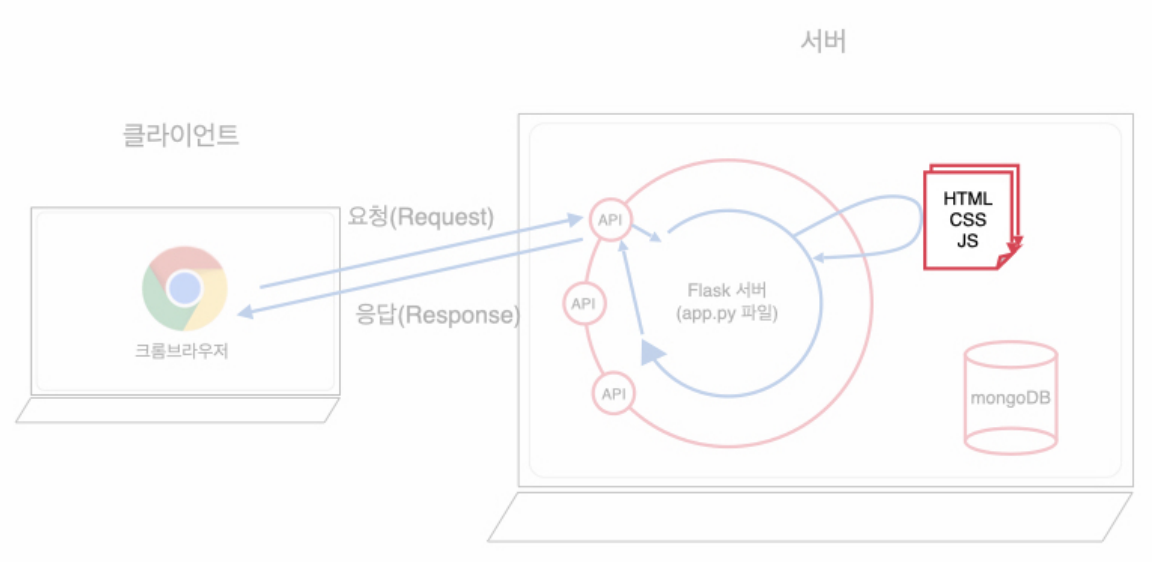
1. 기초 튼튼! 웹 기초 동작 원리, 서버와 클라이언트의 역할 이해하기
2. 프론트엔드(화면 만들기) 기초 다지기

    1) HTML, CSS의 기초 지식을 이해한다. Bootstrap을 가져다 쓸 줄 안다!

    2) Javascript 기본 문법을 익힌다.

▼ 🗺웹 기초 동작 원리

- 클라이언트의 요청(Request)에 서버가 응답(response)해서 데이터를 보내줍니다.
- 보내줄 응답(Response) 데이터인 HTML,CSS 를 배우고, Javscript 기초를 조금 배워봤습니다.



▼ 🧩전반부 주요 키워드

- 웹 기초 동작 원리 그림
- HTTP: 웹은 HTTP라는 규약(규칙)을 따릅니다. url에서 `http://` 가 바로 HTTP 라는 규약을 따른다는 표시에요.
- 클라이언트 : HTTP에서 요청을 하는 쪽
- 서버 : HTTP에서 요청을 받아 응답하는 쪽
- HTML, CSS, JS(Javascript) 는 화면을 구성합니다.
- HTML 은 뼈대, CSS 는 예쁘게 꾸미는 디자인을 담당합니다

▼ 🧩후반부 주요 키워드

- 부트스트랩(Bootstrap): 웹사이트 화면을 쉽게 만들 수 있도록 도와주는 일종의 디자인 키트
- JS(JavaScript, 자바스크립트): 프로그래밍 언어로, 웹 브라우저가 해석할 수 있는 언어입니다. JS 의 더 많은 기능은 2주차에 배웁니다.
- JS 프로그래밍 언어 문법 : 변수, 자료형, 함수(요기까지 배웠습니다), 조건문, 반복문

🧩2주차 - Javascript, jQuery, Ajax

▼ 👨‍🏫수업 목표

1. 프론트엔드 - 웹 페이지에 동작을 줄 수 있는 Javascript 기초 익히기

   1) Javascript 기초 문법 - 조건문, 반복문

   2) jQuery (Javascript 편하게 쓰기 위한 라이브러리) 로 HTML 조작하기

2. API 사용하기

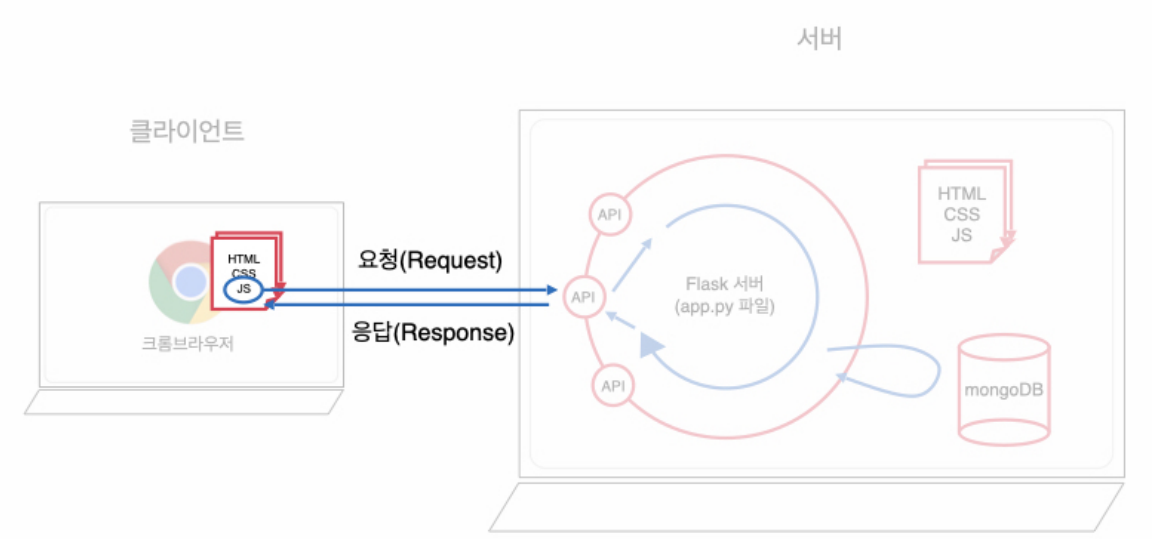
    1) 데이터 표현방식 JSON 이해하기

    2) GET / POST 이해하기

    3) Ajax로 서버 API(약속)에 데이터를 주고, 결과를 받아온다.

▼ 웹 기초 동작 원리

- HTTP: 웹은 HTTP라는 규약(규칙)을 따릅니다. url에서 `http://` 가 바로 HTTP 라는 규약을 따른다는 표시에요.
- 클라이언트 : HTTP에서 요청을 하는 쪽
- 서버 : HTTP에서 요청을 받아 응답하는 쪽
- HTML, CSS, JS(Javascript) 는 화면을 구성합니다.



- Ajax를 이용해 Javascript로 서버에 데이터를 요청하고, 서버가 응답한 데이터를 받아 조작하는 방법을 배웠습니다. 이 때 사용한 것이 API 였습니다.

▼ 전반부 주요 키워드

▼ 후반부 주요 키워드

- GET 요청(Request)
    - 클라이언트가 요청할 때, "방식(Type)"이라는 것이 존재합니다. GET 방식은 통상적으로 데이터 조회(Read)를 요청할 때 사용합니다.
    - GET 요청에서 데이터를 추가로 넘겨주려고 할 때 URL 에 `?` 를 사용해 덧붙이고 key=value 형태로 표시합니다.
        - `https://movie.naver.com/movie/bi/mi/basic.nhn?code=161967`
        - `https://movie.naver.com/movie/bi/mi/basic.nhn?code=30688`
        - `https://google.com/search?q=해리포터&sourceid=chrome&ie=UTF-8`

- JSON : 데이터 표현방식. API로 요청하면 JSON형태로 데이터를 전달해줍니다. Key:Value 로 이루어져 있습니다. 자료형 Dictionary와 아주- 유사하죠.
- Ajax : 비동기 서버 통신방식. jQuery를 사용해 Ajax를 사용해봤습니다.

  1) 웹 페이지 새로고침 없이 서버에 요청(Request)

  2) 서버로부터 데이터를 받고 작업을 수행 할 수 있습니다.
- API 를 사용할 땐, 미리 정해둔 **약속** 을 따라야 작동합니다. 약속들은 API 페이지(문서)에 적혀있습니다.
  - 오늘 수업시간에 사용한 API 페이지 (서울시 권역별 실시간 대기환경 현황 / 서울시 공공자전거 실시간 대여정보 / 랜덤 고양이 사진 API Doc)

🧩3주차 - Python, MongoDB

▼ 👨‍🏫 수업 목표

1. 파이썬(Python) 기본 문법 익히기

   1) 앞으로 만들 API 에서 사용하는 프로그래밍 언어인 파이썬의 기본 문법을 배우기
2. 파이썬 활용 - 웹 페이지를 스크래핑(크롤링) 하기

   1) 정보 수집에 유용한 방법인 웹 스크래핑이 무엇인지 이해하기

   2) 파이썬(Python) 을 사용해 웹 스크래핑 하기
3. 데이터베이스 사용하기

   1) pymongo를 통해 mongoDB를 제어하기

▼ 🗺웹 기초 동작 원리

- 오늘은 우리가 앞으로 만들 API 에서 사용하는 프로그래밍 언어인 파이썬을 배워봤습니다.
- HTTP: 웹은 HTTP라는 규약(규칙)을 따릅니다. url에서 `http://` 가 바로 HTTP 라는 규약을 따른다는 표시에요.
- 클라이언트 : HTTP에서 요청을 하는 쪽
- 서버 : HTTP에서 요청을 받아 응답하는 쪽
- API 를 사용할 땐, 미리 정해둔 **약속** 을 따라야 작동합니다. 약속들은 API 페이지(문서)에 적혀있습니다.

우리가 앞으로 만들 API 에서 사용하는 프로그래밍 언어인 파이썬(Python)을 배웠습니다. 먼저 문법을 연습하고, 라이브러리를 활용하여 웹 스크래핑을 했죠.

그리고, 우리의 인생 첫 데이터베이스. mongoDB의 데이터를 CRUD 기능을 배웠습니다.
(API를 만드는 건 다음주에!)

▼ 전반부 주요 키워드

- [5분 꿀팁] 질문 잘하기 : 내가 해결하고 싶은 문제를 정확히 알려주기, 지금 내가 무엇을 알고 있는지 알려주기, 어디에 어떤 시간에 질문할지, 질문 해결방법에 대해 피드백하기
- API 를 화면 코드에 사용하기 : '나홀로 메모장 포스팅 가져오기 API' 를 사용해 화면에 포스트를 보여줬습니다. 서버에서 JSON 형식으로 아티클 정보(articles)를 응답(response) 데이터로 보내주죠!
- 파이썬: 파이썬은 매우 직관적인 언어이고, 할 수 있는 것도 많습니다. 필요한 것들은 구글링해서 찾아보면 됩니다.
- 프로그래밍에 도움되는 주요 스킬!

  1. **출력하기**
  값을 확인하거나, 에러를 찾을 때 자주 쓰입니다. 출력해서 버그찾기(Debugging By Printing) 스킬! Python 에서는 출력할 때, `print('출력할 값')` 사용합니다.

  2. 이름 짓기 (naming)
  담고 있는 데이터, 기능을 잘 나타낼 수 있는 이름으로 지어주세요. 보통 변수는 명사형으로, 함수는 동사형으로 많이 짓습니다.
  파이썬 에서는 이름짓기 규칙(naming convention)은 snake sytle 을 사용합니다. (`_`로

단어 연결하기. 예. `first_name`)
👉파이썬 공식 스타일 가이드(PEP 8 링크)

- 파이썬 패키지(Python Package) : 외부 라이브러리를 사용하기 위해서 패키지를 설치했습니다. requests, beauifulsoup4, pymongo 모두 외부 라이브러리입니다.
- 파이썬 가상환경(Python Virtual Enviorement) : 프로젝트별 공구함으로,
같은 시스템에서 실행되는 다른 파이썬 응용 프로그램들의 동작에 영향을 주지 않기 위해,
파이썬 배포 패키지들을 설치하거나 업그레이드하는 것을 가능하게 하는 **격리된 실행 환경** 입니다. (출처 : 파이썬 공식 용어집- 가상환경)
- 웹 스크래핑(web scraping): 웹 페이지에서 우리가 원하는 부분의 데이터를 수집해오는 것. 한국에서는 같은 작업을 *크롤링(crawling)* 이라는 용어로 혼용해서 씁니다.

▼ 🧩후반부 주요 키워드

- **CRUD :** 기본적인 데이터 처리 기능인 **Create**, **Read**, **Update**, **Delete** 의 두문자를 따서 **CRUD** 라고 합니다.
- MongoDB : No SQL 의 한 종류인 MongoDB를 pymongo 패키지를 이용해 python으로 조작해보았습니다. mongoDB 에 데이터를 CRUD 하는 것을 배웠습니다.
- Git : 작업 기록을 남기고 이력을 **추적**해서 코드를 손쉽게 관리하도록 도와줍니다.

🧩4주차 - Flaks , API 만들고 사용하기

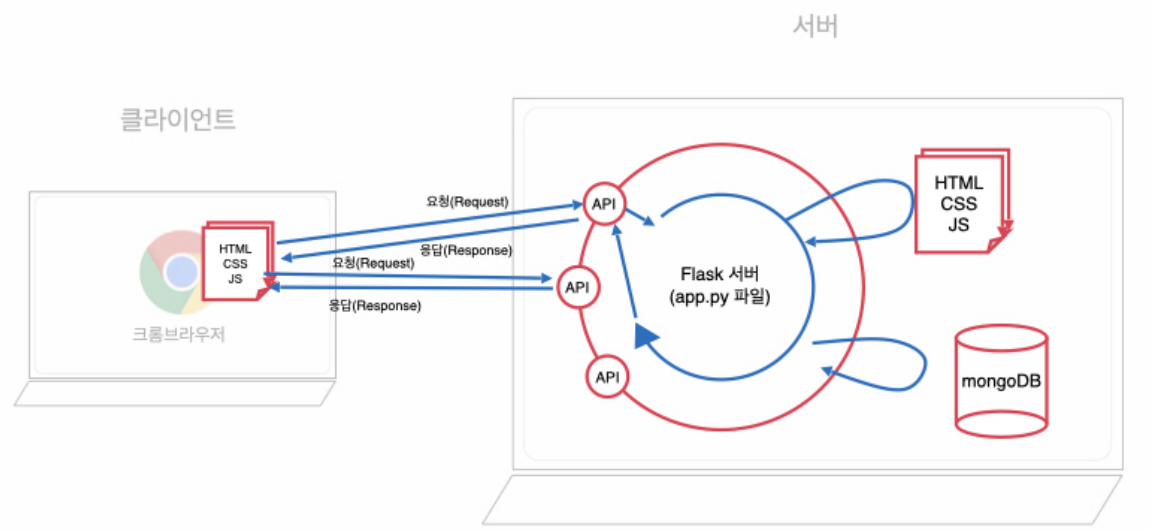
▼ 👨‍🏫 **수업 목표**

1. API 만들기 - API 설계하고, Flask 프레임워크를 사용해 만들기
2. API 사용하기 - 클라이언트 코드에서 API를 사용해 데이터를 보여주기
3. 프로젝트 두 개 완성시키기 - '모두의 책 리뷰', '나홀로 메모장'

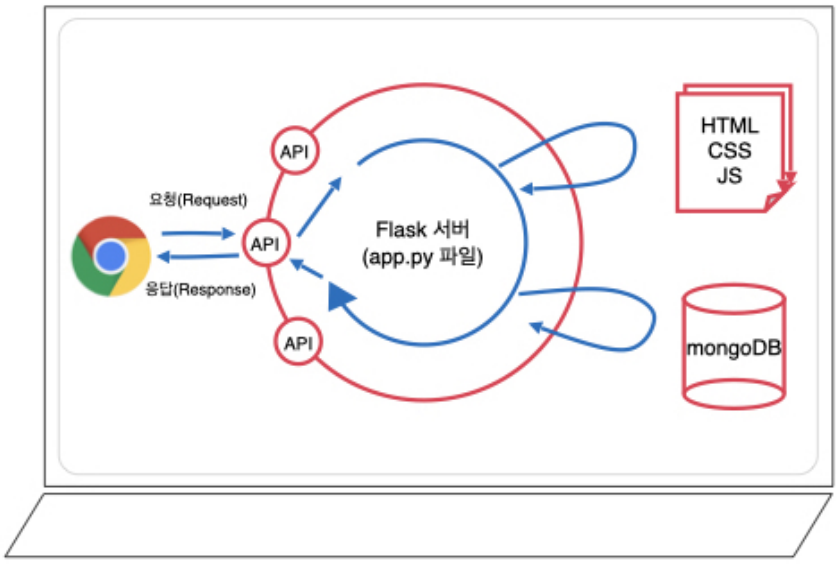
▼ 🗺️웹 기초 동작 원리

서버(백엔드) API를 만들어보았습니다.
그리고 HTML 화면과 mongoDB까지 연동해서 두 가지 프로젝트를 만들었죠.

1. 모두의 책 리뷰 (링크)
2. 나홀로 메모장 (링크)



헷갈리지 말기!
우리는 컴퓨터 한 대를 서버로도 쓰고, 클라이언트로도 사용했습니다.
이것을 바로 "로컬 개발환경"이라고 했죠!



▼ 전반부 주요 키워드

- API(Application Programming Interface): 응용 프로그램(application, 애플리케이션)에서 기능을 사용하거나 데이터를 주고 받기 위한 기능
  - 우리가 사용하는 API는 아래 적힌 모든 것을 미리 약속해두고 그대로 동작합니다.
    1. 요청 정보 : 요청 URL, 요청 방식 (GET / POST /...)
    2. 서버가 제공할 기능 : 데이터 조회(Read), 데이터 생성(Create) 등

3. 응답 데이터 : 응답 데이터 형식 (어떤 key 로 어떤 데이터를 줄지, 예. response['**img**'])

- Flask : 웹을 만들고 서버를 구동시키기 편하게 하는 프레임워크(Framework)(Flask 공식 문서 / 비공식 한글 번역문서)
  - `@app.route('/')` 부분을 수정해서 URL을 부여할 수 있습니다
  - **templates** 폴더 : HTML 파일을 담아둡니다. 실행할 때에는 이 폴더에서 화면을 불러오죠.
  - **static 폴더 :** 이미지나 css파일과 같은 정적 파일을 담아두는 역할을 하지요!
- 클라이언트 요청 방식 - GET, POST
  - GET → 통상적으로! 데이터 조회(Read)를 요청할 때
    예) 영화 목록 조회
    → **데이터 전달** : URL 뒤에 물음표를 붙여 key=value로 전달
    → 예: google.com?q=북극곰
  - POST → 통상적으로! 데이터 생성(Create), 변경(Update), 삭제(Delete) 요청할 때
    예) 회원가입, 회원탈퇴, 비밀번호 수정
    → **데이터 전달 :** 바로 보이지 않는 HTML body에 key:value 형태로 전달

▼ 🧩후반부 주요 키워드

- meta 태그: `<head>~</head>` 부분에 들어가는, 눈으로 보이는 것(body) 외에 사이트의 속성을 설명해주는 태그들입니다.

  예) 구글 검색 시 표시 될 설명문, 사이트 제목, 카톡 공유 시 표시 될 이미지 등

🧩5주차 - Flaks , API 만들고 사용하기 복습

▼ 👨‍🏫 **수업 목표**

1. API 만들고 사용하기
2. 프로젝트 완성시키기 - '마이 페이보릿 무비스타'(링크)
3. 내 프로젝트 기획안 완성하기

- 5주차는 4주차 복습!

🧩6주차 - 클라우드 구입해 사용해보기

▼ 👨‍🏫 **수업 목표**

1. 내 서비스 공개하기! 배포 첫 걸음

1) 서버(EC2) 구매 후 접속하기

2) 구입한 서버(EC2)에서 Flask 프로젝트 실행하기

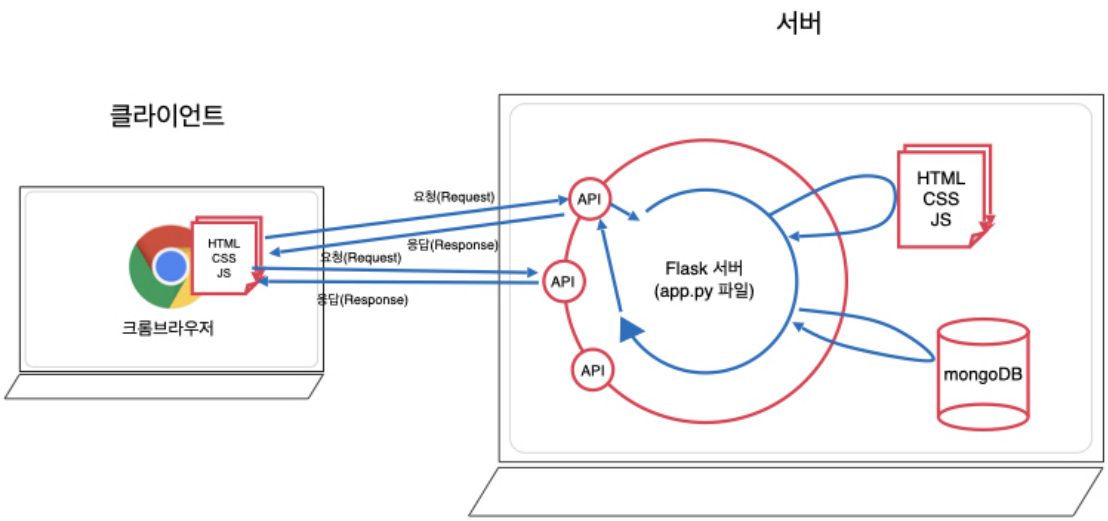
2. 내 프로젝트 발전시키기

▼ 웹 기초 동작 원리 - 총 복습

▼ 총 복습 - HTTP 요청(GET/ POST), HTTP 응답, API, Flask

- HTTP: 웹은 HTTP라는 규약(규칙)을 따릅니다. url에서 `http://` 가 바로 HTTP 라는 규약을 따른다는 표시에요.
- 클라이언트 : HTTP에서 요청을 하는 쪽
- 서버 : HTTP에서 요청을 받아 응답하는 쪽입니다. 응답하는 데이터로는 화면 코드(HTML, CSS, JS) 와 JSON같은 데이터 파일 등이 있습니다.
- 클라이언트 요청 방식 - GET, POST
  - GET → 통상적으로! 데이터 조회(Read)를 요청할 때
    예) 영화 목록 조회
    → **데이터 전달** : URL 뒤에 물음표를 붙여 key=value로 전달
    → 예: google.com?q=북극곰
  - POST → 통상적으로! 데이터 생성(Create), 변경(Update), 삭제(Delete) 요청 할 때
    예) 회원가입, 회원탈퇴, 비밀번호 수정
    → **데이터 전달 :** 바로 보이지 않는 HTML body에 key:value 형태로 전달
- API(Application Programming Interface): 응용 프로그램(application, 애플리케이션)에서 기능을 사용하거나 데이터를 주고 받기 위한 기능
  - API 를 사용할 땐, 미리 정해둔 **약속** 을 따라야 작동합니다. 약속들은 API 페이지(문서)에 적혀있습니다.
  - 우리가 사용하는 API는 아래 적힌 모든 것을 미리 약속해두고 그대로 동작합니다.
    1. 요청 정보 : 요청 URL, 요청 방식 (GET / POST /...)
    2. 서버가 제공할 기능 : 데이터 조회(Read), 데이터 생성(Create) 등
    3. 응답 데이터 : 응답 데이터 형식 (어떤 key 로 어떤 데이터를 줄지, 예. response['**img**'])
- Flask : 웹을 만들고 서버를 구동시키기 편하게 하는 프레임워크(Framework)(Flask 공식 문서 / 비공식 한글 번역문서)
  - `@app.route('/')` 부분을 수정해서 URL을 부여할 수 있습니다
  - **templates** 폴더 : HTML 파일을 담아둡니다. 실행할 때에는 이 폴더에서 화면을 불러오죠.
  - **static 폴더 :** 이미지나 css파일과 같은 정적 파일을 담아두는 역할을 하지요!

서버와 클라이언트가 하나의 컴퓨터에 있던 로컬 개발 환경에서 한 걸음 더 나아가 클라우드에서 구입한 서버에서 Flask 프로젝트를 실행시켜 보았습니다.
실제 서비스 환경처럼 말이죠!



▼ 추가 개념

▼ 클라우드란?

- 그동안 우리는 클라이언트와 서버가 같은 컴퓨터에 있기 때문에 별 다른 설정없이 접근할 수 있었어요.
- 만약 다른 컴퓨터(클라이언트)에서도 내가 만든 프로그램이 실행되고 있는 서버에 접근할 수 있게 하려면 어떻게 해야할까요?
  - 모두가 접근할 수 있는 공개 주소인 공개 IP 주소(Public IP Address)로 나의 웹 서비스에 접근할 수 있도록 해야해요.
  - 그리고 서버가 요청에 언제나 응답할 수 있게 서버가 항상 켜져있고, 웹 서비스가 항상 실행되어 있어야하죠.
- 여러가지 설정을 해서 내 컴퓨터를 항상 켜두어도 되지만, 우리는 웹 서버를 실행하기 편하도록 클라우드에서 서버 사용권을 구입할 거에요.
- ❓클라우드 많이 들어보긴 했는데 무슨 뜻인가요?
  - ❗ 클라우드(인터넷)을 통해 컴퓨터의 리소스를 사용할 수 있는 것입니다. 여기서 컴퓨터의 리소스는 컴퓨터를 이루고 있는 메모리, 저장장치(하드디스크, SSD), CPU 등을 이야기합니다.
  - 네이버 클라우드, 구글 드라이브에 파일을 저장하고 읽어올 수 있죠? 바로 원격으로 쓸 수 있는 저장장치를 산 것과 같죠.

- 우리는 AWS(Amazon Web Service) 라는 곳의 EC2 라는 서비스를 사용할 거에요. 원격으로 어딘가에 내가 사용할 수 있는 컴퓨터를 샀다고 생각하면 됩니다.

▼ **[열 걸음 더 💃]** 클라우드 컴퓨팅의 특징 - 스킵하셔도 괜찮아요!

- 미국 국립표준연구소(NIST) 가 정의한 클라우드 컴퓨팅은 아래와 같은 특징을 가지고 있어요. (원문. The NIST Definition of Cloud Computing)

5 가지 주요 특징 Essential Characteristics

- On-demand self-service: consumer가 컴퓨팅 자원을 요구하는 즉시 자동으로 제공 (인간의 개입이 불필요).
- Broad newtwork Access: 어디 있던 인터넷을 통해 리소스에 액세스
- Resource pooling: provider는 리소스 풀을 확보해서 multi-tenant model로 제공. 규모의 경제. 고객은 리소스 위치에 대해 신경쓸 필요없음.
- Rapid elasticity: 탄력적으로 리소스를 줄이거나 늘릴 수 있음(scale up and down). Capabilities can be elastically provisioned and released
- Measured service: 리소스 사용량이 측정되어서 쓴 만큼만 지불함. 투명성, 리소스 모니터링, 제어 및 보고 가능

3가지 서비스 모델 Service Models

- Software as a Service (SaaS) : 소프트웨어처럼 바로 사용할 수 있음. 예를 들면, MS 오피스 365, 구글 클라우드, 네이버 클라우드
- Platform as a Service (PaaS) : 응용 프로그램(Application)을 작성하고 실행할 수 있는 환경을 제공하는 것. 예를 들면, Google App Engine, Heroku
- Infrastructure as a Service (IaaS) : 사용할 수 있는 인프라를 제공하는 것. 예를 들면, 우리가 사용할 AWS, GCP

4가지 배포 모델 Deployment Models

- Private cloud : 비공개 클라우드. 단일 조직에서만 독점적으로 사용하는 것.
- Community cloud : 특정 용도로만 제한된 조직에서만 사용하는 것.
- Public cloud : 공개 클라우드. 일반인이 공개적으로 사용할 수 있는 것. 대부분의 클라우드 서비스는 여기에 속하겠죠?
- Hybrid cloud : 두 종류 이상의 클라우드로 구성된 것.

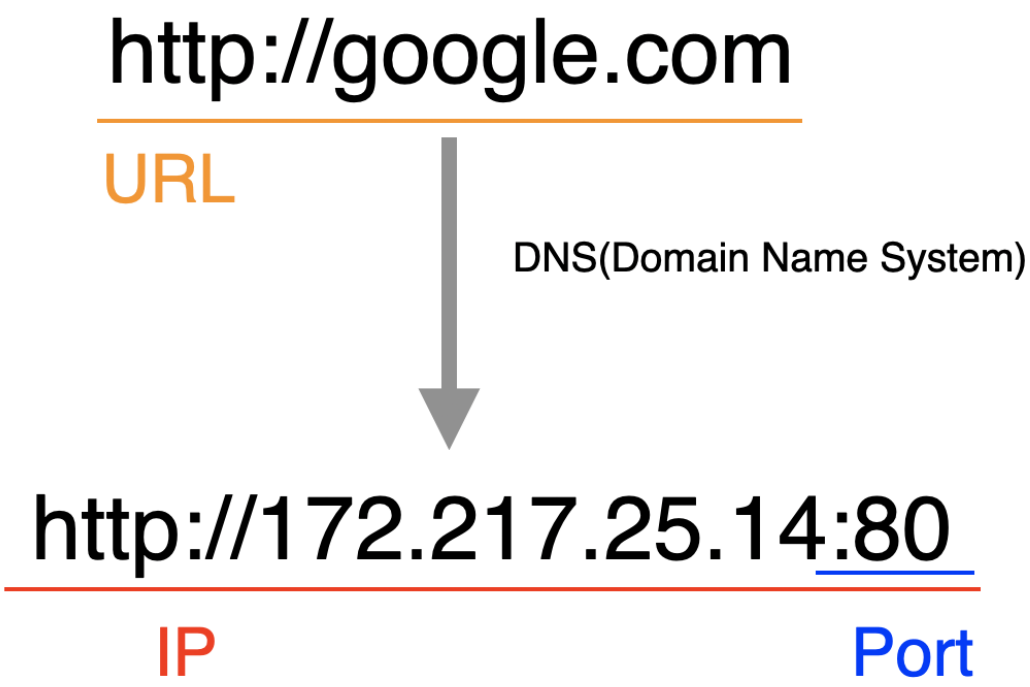
▼ "웹서비스를 런칭하기 위한 작업" 에 필요한 개념 소개

- 배포
  - 이제 내가 만든 프로젝트를 배포해봅니다. 배포는 누구나 내 서비스를 사용할 수 있게 하기 위해서 작업들이에요. 웹 서비스를 런칭하는 거죠!

- 웹 서비스를 런칭하기 위해 클라이언트의 요청에 항상 응답해줄 수 있는 서버에 프로젝트를 실행시켜줄 거에요.
  언제나 요청에 응답하려면,
  1) 컴퓨터가 **항상** 켜져있고 프로그램이 실행되어 있어야하고,
  2) 모두가 접근할 수 있는 공개 주소인 공개 IP 주소(Public IP Address)로 나의 웹 서비스에 **접근할 수 있도록** 해야해요.
- 서버는 그냥 컴퓨터라는거 기억나시죠? 외부 접속이 가능하게 설정한 다음에 내 컴퓨터를 서버로 사용할 수도 있어요.
- 우리는 AWS 라는 클라우드 서비스에서 편하게 서버를 관리하기 위해서 항상 켜 놓을 수 있는 컴퓨터인 EC2 사용권을 구입해 서버로 사용할 겁니다.

▼ **[열 걸음 더 🏃‍♂️]** IP 주소와 포트

- 사실 우리가 접속하는 컴퓨터는 숫자로 되어있는 주소(IP 주소)가 붙어있어요. 우리가 아는 URL 은 우리가 알아보기 쉽게 하는 등의 이유로 IP 주소를 알파벳으로 바꾼 거에요. 이렇게 변환해주는 시스템을 DNS 라고 합니다.



- IP 주소(줄여서 IP라고 부릅니다)
  - 컴퓨터가 통신할 수 있도록 컴퓨터마다 가지는 고유한 주소라고 생각하면 됩니다. 정확히는 네트워크가 가능한 모든 기기가 통신할 수 있도록 가지고 있는 특수한 번호입니다. 서버는 하나의 주소를 가지고 있습니다.
- 포트(port) : 하나의 IP에 여러 포트가 있습니다. 하나의 포트에 하나의 프로그램을 실행시킬 수 있습니다.

출처: 연합뉴스 https://www.yna.co.kr/view/AKR20141231090000004

- 하나의 주차장에 A1, A2, A3, ... 처럼 여러 차들이 각 섹션에 주차할 수 있죠? 주차장 주소를 IP, 포트를 각 섹션, 서버에 실행되고 있는 프로그램을 자동차 로 비유할 수 있습니다.
- 하나의 서버(하나의 IP 를 가짐) 에 여러 포트에 각각 프로그램이 실행될 수 있습니다. 하나의 주차 자리에 하나의 자동차만 주차할 수 있는 것처럼, 하나의 포트에는 하나의 프로그램만 실행될 수 있습니다. 만약 하나의 포트에 여러가지 프로그램이 있다면, 어떤 프로그램에 요청을 했는지 알 수 없겠죠?
- 즉, 클라이언트는 서버의 주소(URL 또는 IP)를 통해 프로그램의 API에 요청(Request)하는 것입니다.

🧩7주차 - 클라우드 서버에 mongoDB 설치하기, 포트 포워딩

▼ 👨‍🏫 수업 목표

1. 내 서버에 mongoDB를 설치하기
2. 서버에서 나홀로 메모장 실행하기
3. 진짜 웹 서비스처럼 실행하기
   1) SSH 접속을 끊어도 서버가 계속 프로젝트를 실행할 수 있게 하기
   2) IP 주소 뒤에 붙는 포트 번호(5000)을 없애기

🧩8주차 - 도메인 연결하기, 웹사이트 정보 og 태그 넣기

▼ 👴🏫 **수업 목표**

1. 웹 사이트 정보 설정하기- og tag 사용
2. 내 서비스에 도메인 주소로 접속할 수 있게 만들기
3. 내 프로젝트를 발표하기!

▼ og tag 만들기

- Flask의 static 폴더 아래에 이미지 파일을 넣고, 각자 프로젝트 HTML의 <head>~ </head> 사이에 아래 내용을 작성하면 og 태그를 개인 프로젝트에 사용할 수 있습니다.

1. "내 사이트의 제목" 입력하기
2. "보고 있는 페이지의 내용 요약" 입력하기
3. 적당한 이미지를 만들거나/골라서 static폴더에 ogimage.png로 저장하기!

```
<meta property="og:title" content="내 사이트의 제목" />
<meta property="og:description" content="보고 있는 페이지의 내용 요약" />
<meta property="og:image" content="{{ url_for('static', filename='ogimage.png') }}" />
```