

SPARTA
CODING
CLUB

스파르타코딩클럽은?



〈 자전거 타기 처럼! 〉

코딩을 접해본 적이 없는 사람
또는 기본기를 다시 다지고 싶은 사람을 대상으로

실습 위주로 코딩을 가르쳐서

빠른 시간 내 ‘혼자 만들 수 있는 단계’ 로 만드는 교육

스파르타코딩클럽은?



〈 자전거 타기 처럼! 〉

“고로 아무질문이나 막해도 됩니다. 사실 다 같이 모름.ㅇㅇ”

“백 번 설명하는 것보다 한번 해보는 게 답. 막 해보는 것을 두려워하지 마세요!”

“코딩의 본질은 삽질. 내가 전세계에서 첫번째로 하는 고민은 없다”

스파르타코딩클럽의 목표는,



집념과



구글만 있으면



크래프트맨

이 될 수 있는 사람들을 만들어 내는 것

그리고 그들간의 커뮤니티를 만드는 것

그래서 세상에 재밌는 일이 많이 생겨나게 하는 것

그래서 빠르게 다 배우는 커리큘럼을 채택

1~5주: 익히기

1주차: 프론트엔드(HTML, CSS, Javascript)

2주차: 프론트엔드+서버통신(jQuery, Ajax)

3주차: 파이썬(크롤링, mongoDB)

4주차: 미니프로젝트1, 2(flask)

5주차: 미니프로젝트3(1~4주차 종합)

6~8주: 삼질&만들기

수업: AWS에 내 프로젝트를 올리기
(다른 사람들이 볼 수 있게!)

코딩: 개인/팀 프로젝트 진행
- 튜터가 함께 기획 & 코드리뷰

그래서 빠르게 다 배우는 커리큘럼을 채택

1~5주: 익히기

1주차: **프론트엔드**(HTML, CSS, Javascript)

2주차: **프론트엔드+서버통신**(jQuery, Ajax)

3주차: **파이썬**(크롤링, mongoDB)

4주차: **미니프로젝트1, 2**(flask)

5주차: **미니프로젝트3**(1~4주차 종합)

6~8주: 삼질&만들기

수업: AWS에 내 프로젝트를 올리기
(다른 사람들이 볼 수 있게!)

코딩: 개인/팀 프로젝트 진행
- 튜터가 함께 기획 & 코드리뷰

↑
`JavaScript'의 사용을 좀 더 쉽게 해주기
위해서 생긴 라이브러리

자 이제 시작해봅시다



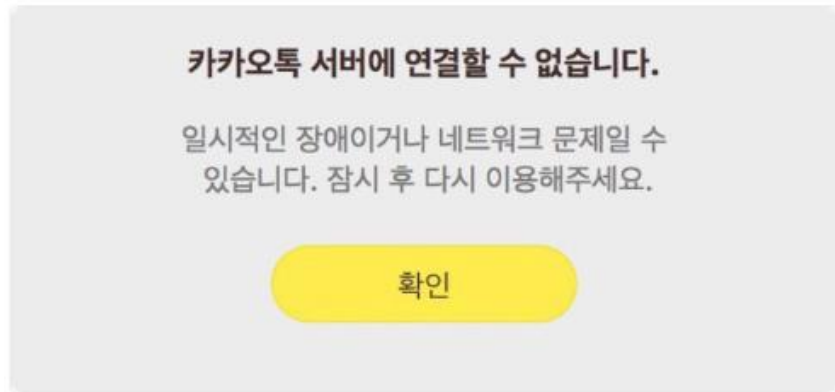
배우는 순서

서버와 클라이언트의 개념 잡기

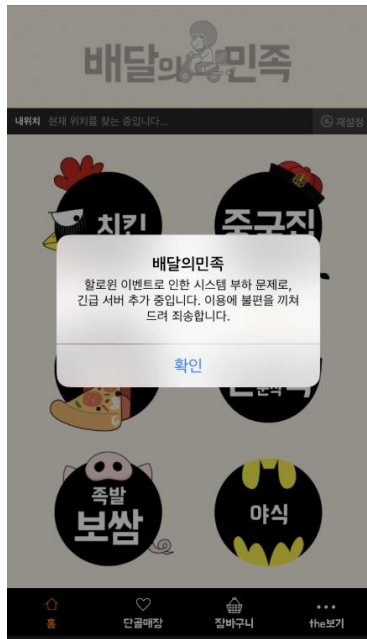
웹서비스의 동작 방식 파악하기

서버(API)를 만드는 방법에 관해

많이 듣던 얘기 – 서버/클라이언트의 개념을 잡아보자

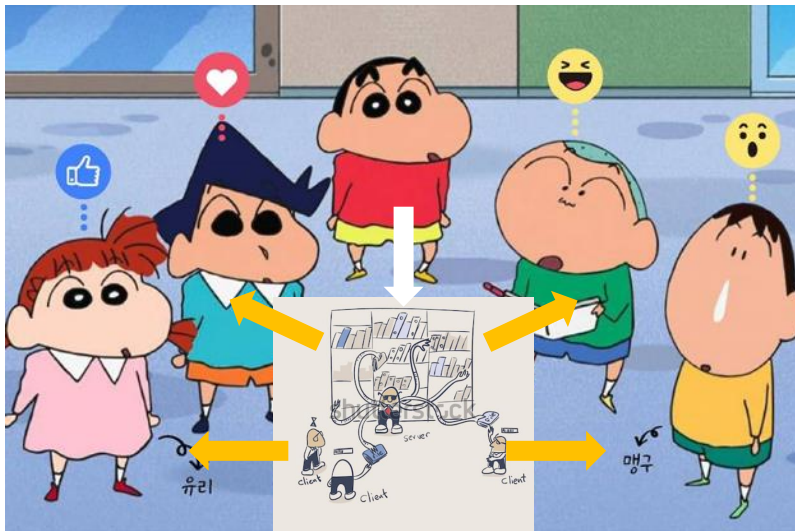


접속자 폭주 -> 어? 서버 터졌네



왜 터질까요?

그리고 터지면 왜 카톡이 안보내질까요?

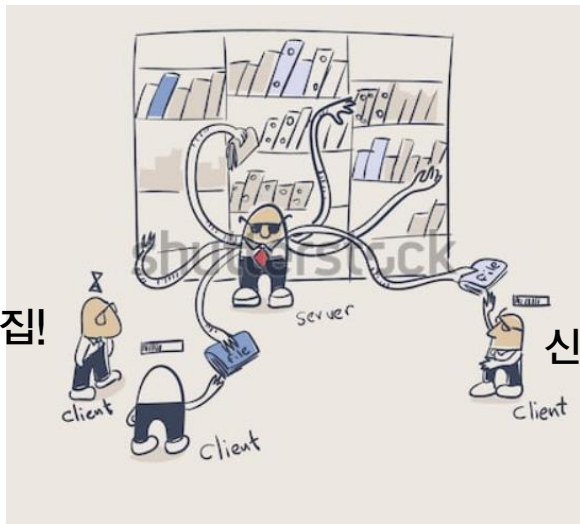


카카오톡 서버가 일을 한다는 뜻이겠지요

어떤 일을 하길래?

서버의 주된 역할은, 중앙에서 데이터를 주고 받는 것입니다.

역삼동 맛집!



신림동 맛집!

덕분에 모든 사용자들이 **일관된 정보**를 볼 수 있게 되지요

그럼 서버는 어떻게 생겼을까요?

상상과 달리 그냥 컴퓨터입니다



태어날 때부터 서버로 태어나지 않았습니다
그냥 서버 프로그램을 깔아서 그 일만 시키는거죠

그럼 서버는 어떻게 생겼을까요?

상상과 달리 그냥 컴퓨터입니다



클라이언트, 서버, DB, 라우터, 모니터링 등
이것들은 컴퓨터들의 직업이라 생각하면 됩니다
컴퓨터가 맡을 수 있는 역할

네이버를 바꿔보자!

네이버를 시작페이지로 > | 쉼터네이버 | 해피빈



NAVER

100



메일

카페

블로그

지식iN

쇼핑

Pay

TV

사전

뉴스

증권

부동산

지도

영화

뮤직

책

웹툰

더보기 >

4월 30일



연합뉴스 > 서울 주요 대학 모두 내년엔 정시 확대...연고대 40%...

스파르타!!!

연예 스포츠 경제



뉴스스탠드 > 구독한 언론사 · 전체언론사



연합뉴스TV	아시아경제	전자신문	머니투데이	Korea JoongAng Daily	이데일리
스포츠조선	일간스포츠	한겨레	프레시안	SBS	노컷뉴스
사사iN	중앙일보	한국경제TV	ScienceTimes	일간중앙	TBS

네이버를 더 안전하고 편리하게 이용하세요

NAVER 로그인

아이디 · 비밀번호찾기

회원가입

미세



보통

초미세



보통

대치등



< >



웹의 동작 원리를 추론해보기

1) 왜 내 브라우저에서 마음대로 바꿀 수 있는 걸까요?

2) 왜 F5를 누르면 다시 바뀌는 걸까요?

3) 브라우저의 역할은 무엇일까요?



1. 서버에 요청함.

2. 서버의 응답을 화면에 뿌려줌(= 그려줌)

서버가 클라이언트에게 전달하는 데이터

웹의 동작 원리를 추론해보기

- 1) 왜 내 브라우저에서 마음대로 바꿀 수 있는 걸까요?
→ 이미 받아온 걸 바꾸니까요!
- 2) 왜 F5를 누르면 다시 바뀌는 걸까요?
→ 새로 받아오니까요!
- 3) 브라우저의 역할은 무엇일까요?
→ 받아온 걸 그립니다!

핵심: 웹은 서버에서 **그릴** 것도 주고, **데이터**도 준다

웹의 동작 원리 (설명서 주기)

주소 창에 URL을 딱 치면,



쳤어. 나 뭐해?



서버

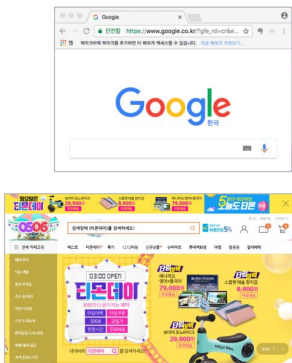


사람이 잘 볼 수 있게
요대로 그려주면 돼
(이 위치에 메뉴바 그려주고..)

그리고 요렇게 움직여줘
(1초마다 이미지 넘기고..)

웹의 동작 원리 (설명서 주기)

주소 창에 URL을 딱 치면,



쳤어. 나 뭐해?

(이 위치에 메뉴바 그려주고..)

→ HTML: 뼈대

→ CSS: 예쁘게

(1초마다 이미지 넘기고..)

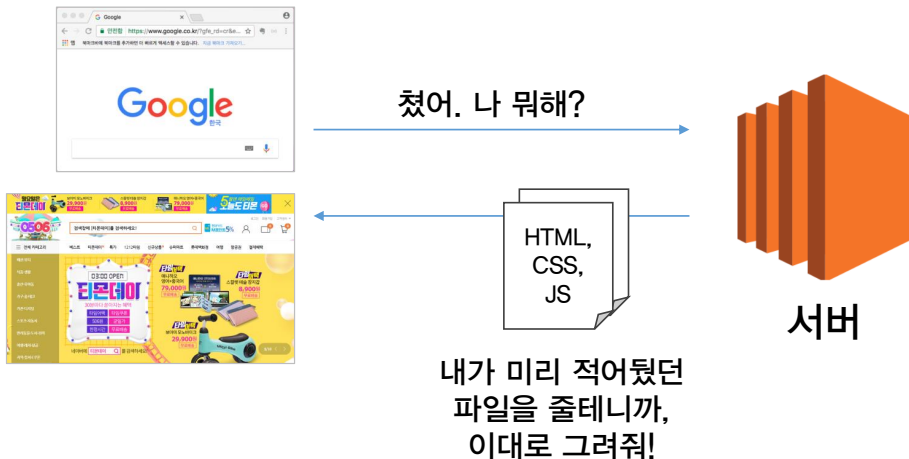
→ Javascript

브라우저가 유일하게 알아듣는 언어
(요렇게 그려줘 / 요렇게 움직여줘)



웹의 동작 원리 (설명서 주기)

주소 창에 URL을 딱 치면,



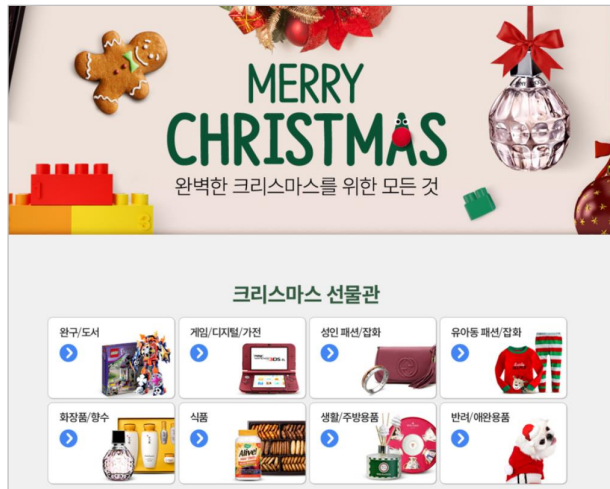
앱도 같을까?

앱과 웹의 차이점?

앱은 업데이트 해야하는데, 웹은 그런 말 없었다! 왜?

“난 업데이트 안해서 카톡에 눈이 안와”
→ 이런 사람 있다.

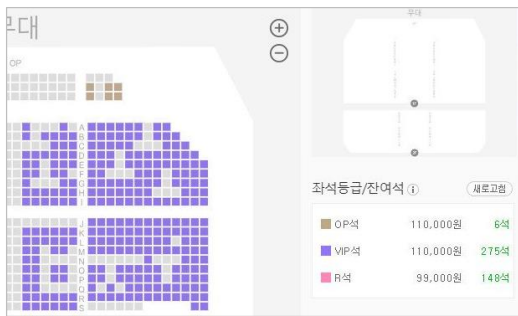
“난 업데이트 안해서 쿠팡 크리스마스
이벤트 안보여”
→ 이런 사람 없다.



웹의 동작 원리 (데이터만 주기)

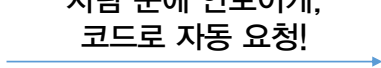
'Ajax' 통신

주소 창에 URL을 딱 치면, 칠 필요 없이 뒤에서 요청 주고받게



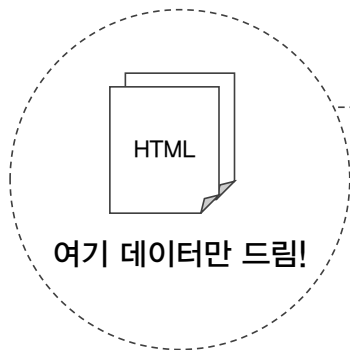
브라우저에 엔터 치면 리프레시 되잖아..
데이터만 줄 수는 없어?
그러면 그 부분만 갈아 끼우게

사람 눈에 안보이게,
코드로 자동 요청!



서버

여기 텍스트만 드림!
(텍스트=데이터)



여기 데이터만 드림!

대략 이렇게
생겼습니다

```
← → ↻ ⓘ 주의 요함 | openapi.seoul.go.kr:8088/6d4d7
{
  - RealtimeCityAir: {
    list_total_count: 25,
    - RESULT: {
      CODE: "INFO-000",
      MESSAGE: "정상 처리되었습니다"
    },
    - row: [
      - {
        MSRDT: "201912051900",
        MSRRGN_NM: "도심권",
        MSRSTE_NM: "중구",
        PM10: 16,
        PM25: 6,
        O3: 0.022,
        NO2: 0.014,
        CO: 0.4,
        SO2: 0.003,
        IDEX_NM: "좋음",
        IDEX_MVL: 37,
        ARPLT_MAIN: "03"
      },
      - {
        MSRDT: "201912051900",
        MSRRGN_NM: "도심권"
```

이런 형식으로 데이터 주는 것을 'JSON' 형식이라고 합니다(곧 배워요!)

요청은 그냥 아무나/아무렇게나 하면 되는건가요?

서버와 클라이언트는 어떻게? 통신하는 걸까?



예를 들어, 은행에는 수많은 데이터가
있으니까,

은행을 서버라고 해봅시다. 고객은
클라이언트겠죠

저는, 잔고를 확인하고 싶어요.

어떻게 해야하죠?

서버가 은행이라고 하면

서버 = 은행 / 클라이언트 = 고객

Q. 은행에 가서, 내 통장 잔고를 확인하는 방법은?

- 1) 아무나에게 가서 물어본다
- 2) 대출 창구에 간다
- 3) 크게 소리친다
- 4) 번호표를 뽑고 입출금 창구에 가서 민증을 보여준다

서버가 은행이라고 하면

서버 = 은행 / 클라이언트 = 고객

Q. 은행에 가서, 내 통장 잔고를 확인하는 방법은?

- 1) 아무나에게 가서 물어본다
- 2) 대출 창구에 간다
- 3) 크게 소리친다
- 4) 번호표를 뽑고 입출금 창구에 가서 민증을 보여준다

서버가 은행이라고 하면

창구는 고객이 은행이 마련해놓은 서비스를 요청할 수 있는 방법



은행이 수많은 데이터를 가지고 있지만,
정해진 업무를 하는 창구에서 정해진 방식으로 소통해야하죠

서버도 마찬가지로?

수 많은 데이터와 기능을 할 수 있겠지만, 미리 짜놔야 고객이 실행할 수 있는 것입니다.

예를 들어!

- 1) 내가 <http://api.naver.com/myinfo> 라고 하면서
아이디 비번을 주면 → 내 정보를 보여주거나
- 2) 내가 <http://spartacoding.kr/api/classes> 라고 하면
클래스 목록을 보여주는 것들이죠.

클라이언트가 정해진 규칙대로 요청을 보내면, 정해진 기능이 동작하는 것입니다

서버가 은행이라고 하면

즉, 창구는 고객과 은행의 데이터 뽑기 약속입니다

창구 = API = 약속 (규칙 · 제약)

서버가 은행이라고 하면

즉, 창구는 고객과 은행의 데이터 뽑기 약속입니다

창구 = API = 약속

<클라이언트의 요청
방식으로 '서버 프로그래머'
가 설정한다.>

서버 프로그래머:

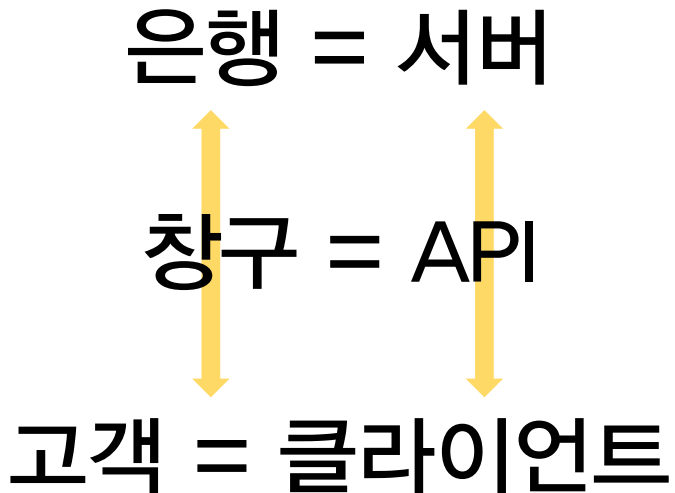
- 1) 어떤 데이터를 받아서
- 2) 어떻게 처리해서
- 3) 어떤 데이터를 줄지

(ex: `router.get('/', ~)`)

동작을 미리 짜두는 사람

서버가 은행이라고 하면

중요한 거니까 한번 더 정리



그럼 마지막으로, 서버는 어떻게 만들어요?

그리고 DB라는 친구도 있던데.. 개는 뭐죠?



참 좋은 질문입니다

지금까지 우리는 ‘서버’ 라는 친구로

클라이언트가 아닌 모든 아이들을 지칭했었죠



사실 서버 안에는
엄청나게 많은 역할들이 존재합니다
(은행도 창구 상담원, 금고 관리원 등 많은 역할이 있듯이!)

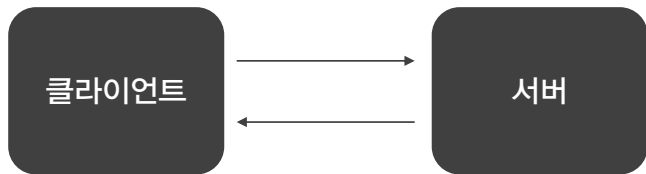
지금까지 우리는 '서버'라는 친구로

클라이언트가 아닌 모든 아이들을 지칭했었죠

서버와 클라이언트 구성. 간단부터 심오 버전까지

쉽게 이해하자!

사용자가 별로 없을 때(=작은 은행일 때)



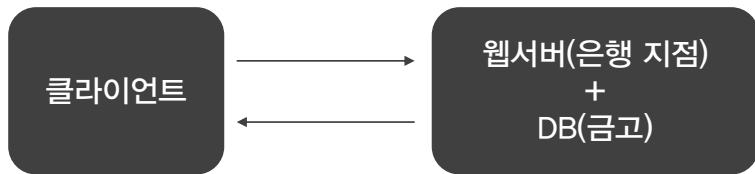
지금까지 우리는 '서버'라는 친구로

클라이언트가 아닌 모든 아이들을 지칭했었죠

서버와 클라이언트 구성. 간단부터 심오 버전까지

쉽게 이해하자!

사용자가 별로 없을 때(=작은 은행일 때)

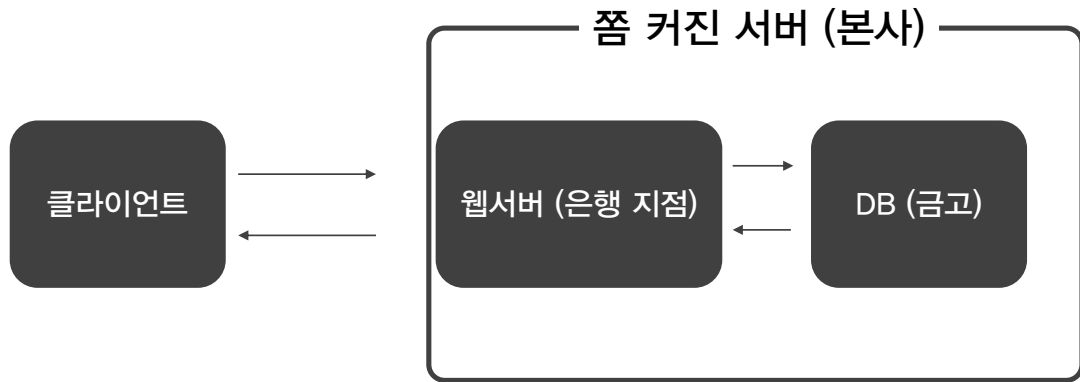


지금까지 우리는 '서버'라는 친구로

클라이언트가 아닌 모든 아이들을 지칭했었죠

서버와 클라이언트 구성. 간단부터 심오 버전까지

좀 잘된다. 유저가 많아져서 데이터 저장소를 따로 뺌 (DB)

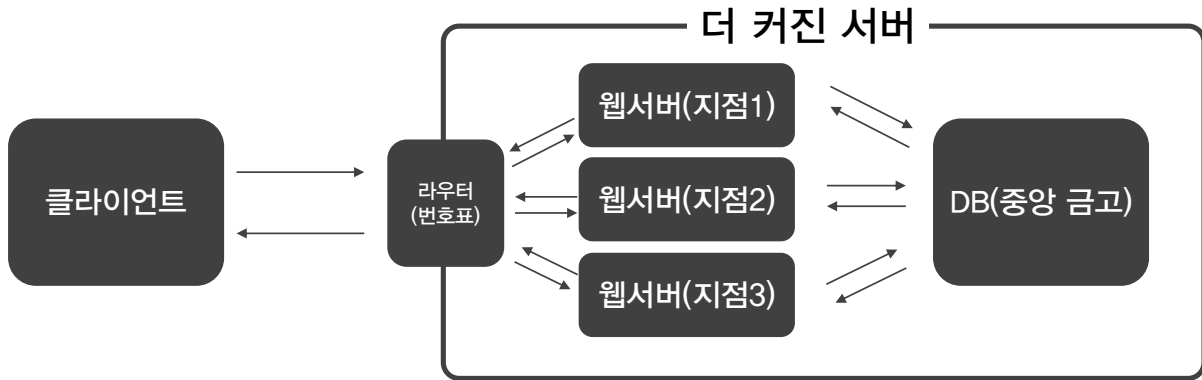


지금까지 우리는 '서버'라는 친구로

클라이언트가 아닌 모든 아이들을 지칭했었죠

서버와 클라이언트 구성. 간단부터 심오 버전까지

이번주 트래픽 터질지도 모름

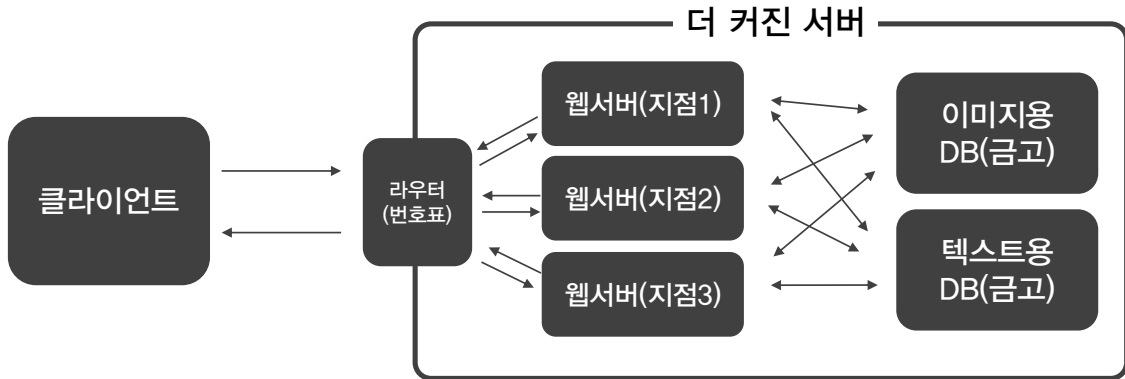


지금까지 우리는 '서버'라는 친구로

클라이언트가 아닌 모든 아이들을 지칭했었죠

서버와 클라이언트 구성. 간단부터 심오 버전까지

이미지는 따로 저장하자! 파일 사이즈가 너무 크네

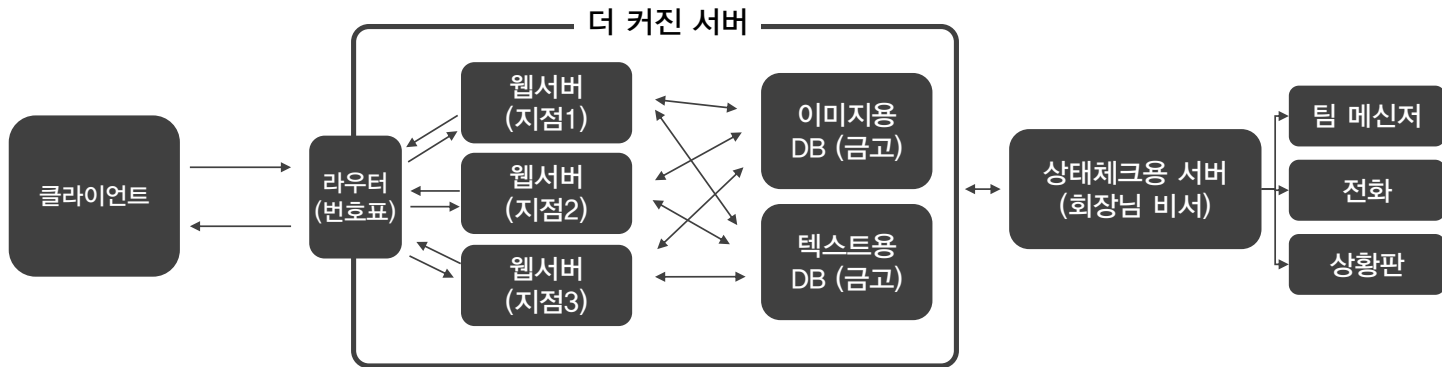


지금까지 우리는 '서버'라는 친구로

클라이언트가 아닌 모든 아이들을 지칭했었죠

서버와 클라이언트 구성. 간단부터 심오 버전까지

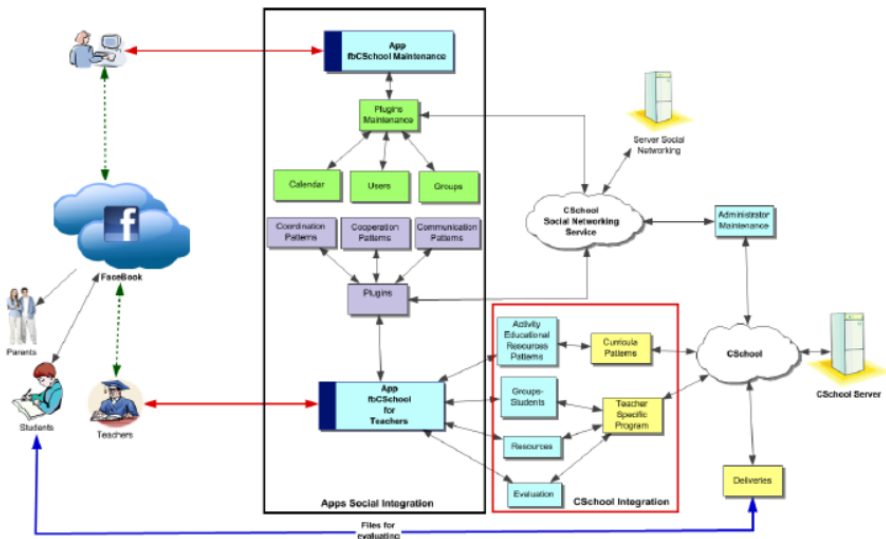
서버상태도 체크하고 싶어. 갑자기 죽으면 어떡해



지금까지 우리는 '서버'라는 친구로

클라이언트가 아닌 모든 아이들을 지칭했었죠

이런 게 모아지고, 모아지면 아래와 같은 '시스템 아키텍처' 가 되는 것
복잡해 보이지만, 이해할 필요는 없음 (우리회사꺼 아니면 개발자도 쉽게 이해 불가)



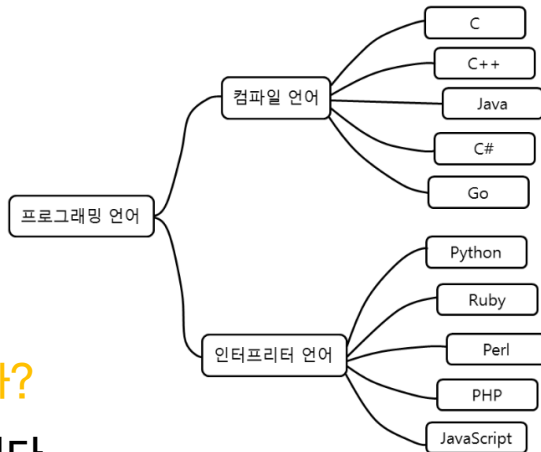
그럼 이걸 어떻게 설치하고 작동시키느냐!

그것을 프로그래밍언어로 하는 것입니다.

10111110111010101011...

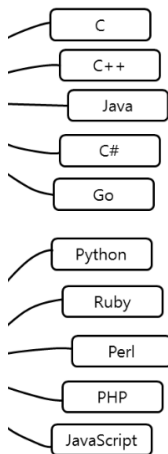
Q. 언어라 함은? 왜 언어가 필요한가?

A. 컴퓨터와 소통하기 위해 필요합니다



그럼 이걸 어떻게 설치하고 작동시키느냐!

그것을 프로그래밍언어로 하는 것입니다.



“컴퓨터와 소통하기 위해 필요합니다” 의 뜻

-> 즉, 꼭 서버를 만들기 위해 필요한 것이 아닙니다!!

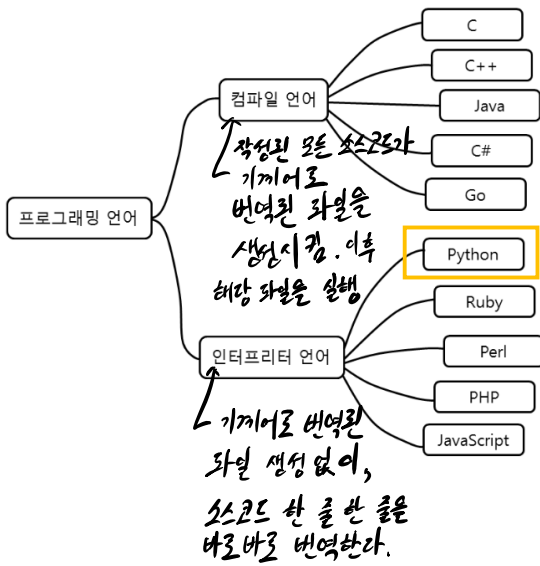
-> 컴퓨터에게 모든 동작을 명령내리기 위한 것

-> 서버를 만들 수도 있고,
DB와 소통할 수도 있고,
파일 100개의 이름을 바꿀 수도 있고,
1시간 뒤에 꺼지게 할 수도 있고

그럼 이걸 어떻게 설치하고 작동시키느냐!

그것을 프로그래밍언어로 하는 것입니다.

먹다 = Eat = 吃 = ... ? 못하는 건 없지만 특성이 다른 것



Q. 왜 파이썬이예요?

여러 선택지가 있는데, 가장 배우기 쉬워서
#직관적인_문법 #방대한_자료
#갖다_쓸_라이브러리_많음

하지만 1부터 다 만들기란 쉽지 않죠

그래서 **프레임워크**라는 것이 등장합니다. 3분 카레나 국간장, 케찹 같은 것이죠

↑ 이리 만들어져 있는 '틀'.



VS



프레임워크 없이

프레임워크로
(그래도 할 게 아주 많음)

자 이제 커리큘럼을 다시 볼까요?

1~5주: 익히기

1주차: 프론트엔드(HTML, CSS, Javascript)

2주차: 프론트엔드+서버통신(jQuery, Ajax)

3주차: 파이썬(크롤링, mongoDB)

4주차: 미니프로젝트1, 2(flask)

5주차: 미니프로젝트3(1~4주차 종합)

무작정따라하기

웹의 동작 원리 (움직그림 설명서 주기)

주소 창에 URL을 딱 치면,



서버가 줄 것을
만들어두는 단계

자 이제 커리큘럼을 다시 볼까요?

1~5주: 익히기

1주차: 프론트엔드(HTML, CSS, Javascript)

2주차: 프론트엔드+서버통신(jQuery, Ajax)

3주차: 파이썬(크롤링, mongoDB)

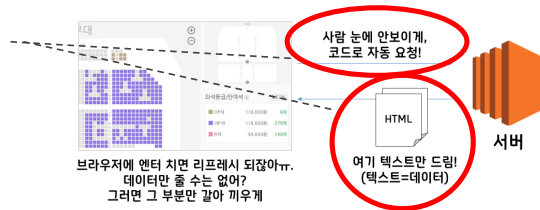
4주차: 미니프로젝트1, 2(flask)

5주차: 미니프로젝트3(1~4주차 종합)

무작정따라하기

웹의 동작 원리 (데이터만 주기)

주소 창에 URL을 딱 치면, 칠 필요 없이 뒤에서 요청 주고받게



서버에 데이터를
요청하고 받는 단계

자 이제 커리큘럼을 다시 볼까요?

1~5주: 익히기

1주차: 프론트엔드(HTML, CSS, Javascript)

2주차: 프론트엔드+서버통신(jQuery, Ajax)

3주차: 파이썬(크롤링, mongoDB)

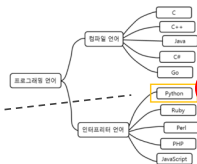
4주차: 미니프로젝트1, 2(flask)

5주차: 미니프로젝트3(1~4주차 종합)

그럼 이걸 어떻게 설치하고 작동시키느냐!

그것을 프로그래밍언어로 하는 것입니다.

먹다 = Eat = 吃 = ... ? 못하는 건 없지만 특성이 다른 것



Q. 왜 파이썬이예요?

여러 선택지가 있는데, 가장 배우기 쉬워서
#작업적인_방법 #병대한_자료
#값다_올_라이브러리_많음

파이썬을 가지고 놀아보자!
(영화 제목 자동으로
클어오기 등)

자 이제 커리큘럼을 다시 볼까요?

1~5주: 익히기

1주차: 프론트엔드(HTML, CSS, Javascript)

2주차: 프론트엔드+서버통신(jQuery, Ajax)

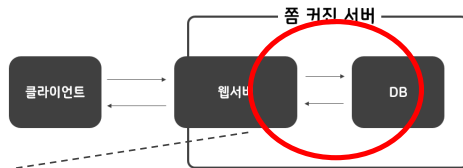
3주차: 파이썬(크롤링, mongoDB)

4주차: 미니프로젝트1, 2(flask)

5주차: 미니프로젝트3(1~4주차 종합)

서버와 클라이언트 구성. 간단부터 심오 버전까지

좀 잘된다. 유저가 많아져서 데이터 저장소를 따로 땀 (DB)



DB를 설치하고
파이썬으로
제어해보자

자 이제 커리큘럼을 다시 볼까요?

1~5주: 익히기

1주차: 프론트엔드(HTML, CSS, Javascript)

2주차: 프론트엔드+서버통신(jQuery, Ajax)

3주차: 파이썬(크롤링, mongoDB)

4주차: 미니프로젝트1, 2(flask)

5주차: 미니프로젝트3(1~4주차 종합)



프레임워크를
이용해서
API를 만들어보자
(API = 창구!)

수고하셨습니다!
이제 즐거운 마음으로 1주차 수업을 기다려보시죠