UVC
Ubiquitous Via Connect

웹 개발을 위한 기초 상식

비 전공자를 위한

온라인 사이트

html/css/javascript
- https://www.w3schools.com/

IT개발 일반
- https://wikidocs.net/book/2184

json formatter
- https://jsonformatter.org/

구글 신은 답을 줄 것이다.
- `stackoverflow`를 통해

Internet

Internet

Internet (International network)

- 컴퓨터끼리 전세계적으로 네트워크 망을 통해 연결 됨
- 만든 기관: 미국 국방부(ARPA)(1960~70)

Intranet (폐쇄 망/사설 망/내부 망)

- 컴퓨터끼리 내부적으로 네트워크 망을 통해 연결 됨
- 망 분리와 무슨 차이?? (H/W 망 분리, S/W 망 분리)
- 내부에서 외부로의 단절(이게 인트라넷?), 외부에서 내부로의 단절(이게 인트라넷?), 양방향 단절(이건 인트라넷)
- 인트라넷이라는 용어보다 그냥 " 내부 망 " 이라고 부름. (외부에서 못 들어가면 내부 망)
- VPN(Virtual Private Network)을 통해 접속 가능한 경우도 있음

Internet

Web/www(world wide web)/w3

- 인터넷을 통해 정보를 공유하는 서비스 중 하나(web !== internet, web < internet)
- 만든 사람: 유럽 입자 물리 연구소(CERN)의 팀 버너스-리(1989)
  - 어딘 가에 저장되어 있는 문서를 네트워크를 통해서 읽어보자.
  - 문서의 위치 및 읽어오는 방식을 규격화 하자(protocol) (각자 PC에 문서가 개인 취향에 따라 분류된 상황)
  - 하이퍼링크를 통해 다른 문서로 이동 가능하게 하자(web surfing) (각자 PC에 작성한 문서를 링크를 통해 공유 가능)
- 문서를 읽는 프로그램: web browser(~~넷스케이프~~, 크롬, 사파리, 엣지, 파이어폭스, 오페라, ~~IE~~)
- 문서에 접근하기 위해 호출하는 서버의 이름: domain
- 정확한 문서의 위치: URL
- 왜 대부분의 도메인 앞에는 www를 붙일까???
  - web서비스를 하는 서버의 이름을 몰라서.
  - 그냥 안 붙이면 안되니?? (안 붙이면 web서버로 연결하도록 담당자가 설정해 주어야 한다.)

Internet

URL/URI

- URI가 URL을 포함하는 규약이지만 중요치 않다.
- 그냥 주소, 웹 주소, URL로 통칭 한다.
- URL의 구조

URL의 (원래) 구조

scheme://<user>:<password>@<host>:<port>/<url-path>

URL의 일반적인 구조

scheme://<host>:<port>/<path>?<query>#<fragment>

(프로토콜)(도메인)(포트)(경로)(파라미터)(책갈피)

예)

https://kr.vuejs.org/v2/guide/instance.html#Vue-%EC%9D%B8%EC%8A%A4%ED%84%B4%EC%8A%A4-%EB%A7%8C%EB%93%A4%EA%B8%B0

(URL encoding???)

Domain의 구조

- IP주소를 가지고 서비스를 찾기엔 인간의 기억력이 너무 부족하다.
  - 전화번호: IP주소
  - 친구 이름: 도메인 주소 (전화번호를 숫자로 누르지 않아도 친구 이름만 고르면 통화 가능)
  - IP와 도메인을 매핑해 주는 서비스: DNS(Domain Name System)
- <서브도메인>.<도메인>.<최상위 도메인>
- 서브도메인: 서버 이름(www가 서버 이름이라고?, 서브 도메인이 없는 경우도 있던데?)
- 도메인: 서비스 이름(google, daum 같은 이름)
- 국가 코드 최상위 도메인 (링크)
- 일반 최상위 도메인 (링크)
- .go.kr 도메인은 국가에서 운영하는 서비스에만 붙일 수 있다. (믿을 수 있다.) (gov는 미국이 찜 했음.)
- (실습: whois 도메인 검색 방법 확인 https://domain.whois.co.kr/regist/dom.php)

Internet

UVC
Ubiquitous Via Connect

TCP/UDP

- TCP(Transmission Control Protocol): 연결형 프로토콜(like 전화)
  - TCP/IP가 한데 묶여서 사용됨
- UDP(User Datagram Protocol): 비 연결형 프로토콜(like 방송)

Internet

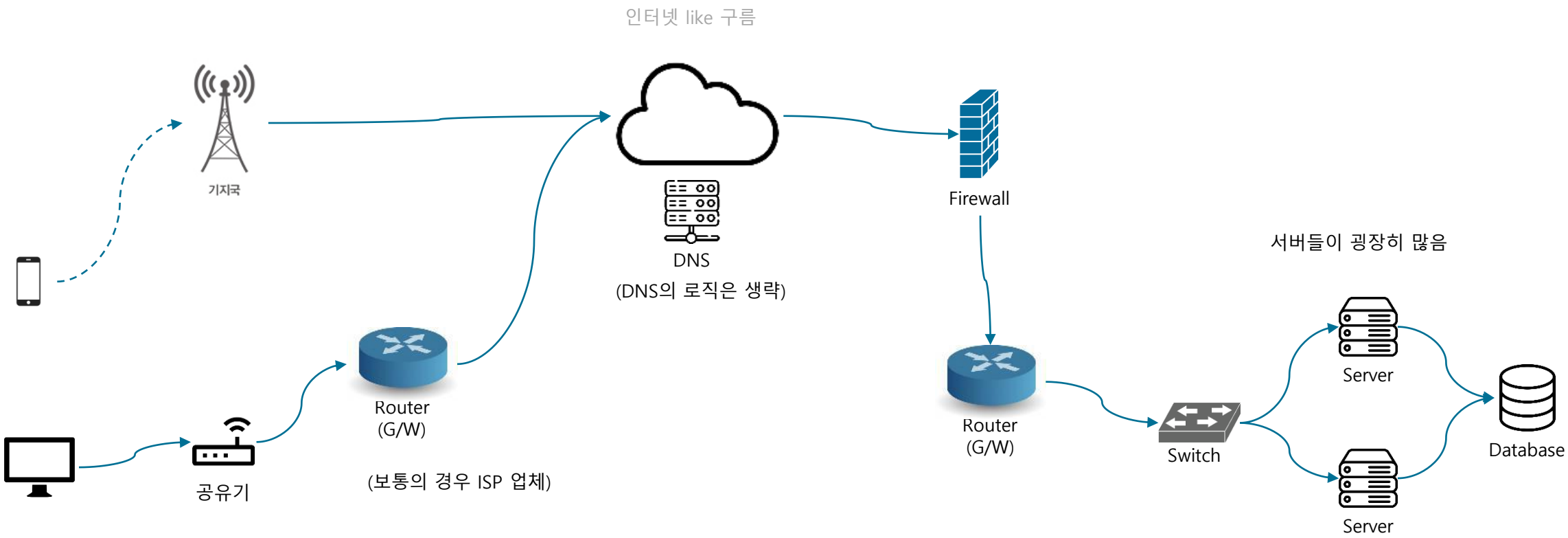
IP 주소

- 보통 IP주소(Internet Protocol Address)는 그냥 IP라고 부름
- IP할당
  - 고정된 IP주소 할당(PC나 서버에 IP주소를 적어줘야 함)
  - 동적 IP 할당(DHCP)(DHCP서버가 IP를 알아서 배포 함)
  - IP확인 방법
    - 윈도우: > ipconfig (실습)
    - 리눅스: $ ifconfig
  - IPv4: 일반적으로 알고 있는 IP주소 체계 (0.0.0.0 ~ 255.255.255.255)(=32비트=2^{32})(2^8 = 256 x 4개)
    - 외울 수 있음(자주 쓰는 서버나 본인의 개발용 PC)
    - 보통 서버의 명칭으로 사용함(마지막 끝자리 번호를 부름)
  - IPv6: IPv4의 고갈(예정)으로 인해 확장을 위한 만든 주소 체계
    - 이미 당신의 PC에도 할당 되어 있음
    - 외울 수 없음
- 내부 아이피: 192.168.x.x, 10.0.x.x
- 내 아이피: 127.0.0.1, localhost

Port

- IP가 전화번호라면 Port는 내선번호다. (0 ~ 65535)
- 네트워크를 통해 호출하는 서비스 중 Port가 할당되지 않은 서비스는 없다. (반드시 포트를 호출 해야함)
  - 하지만 난 포트번호를 써 본적도 없는데? (해당 프로그램이 자동으로 세팅해 줬을 뿐이다.)
  - 잘 알려진 포트 http: 80, https: 443, ssh: 22, ftp: 20,21

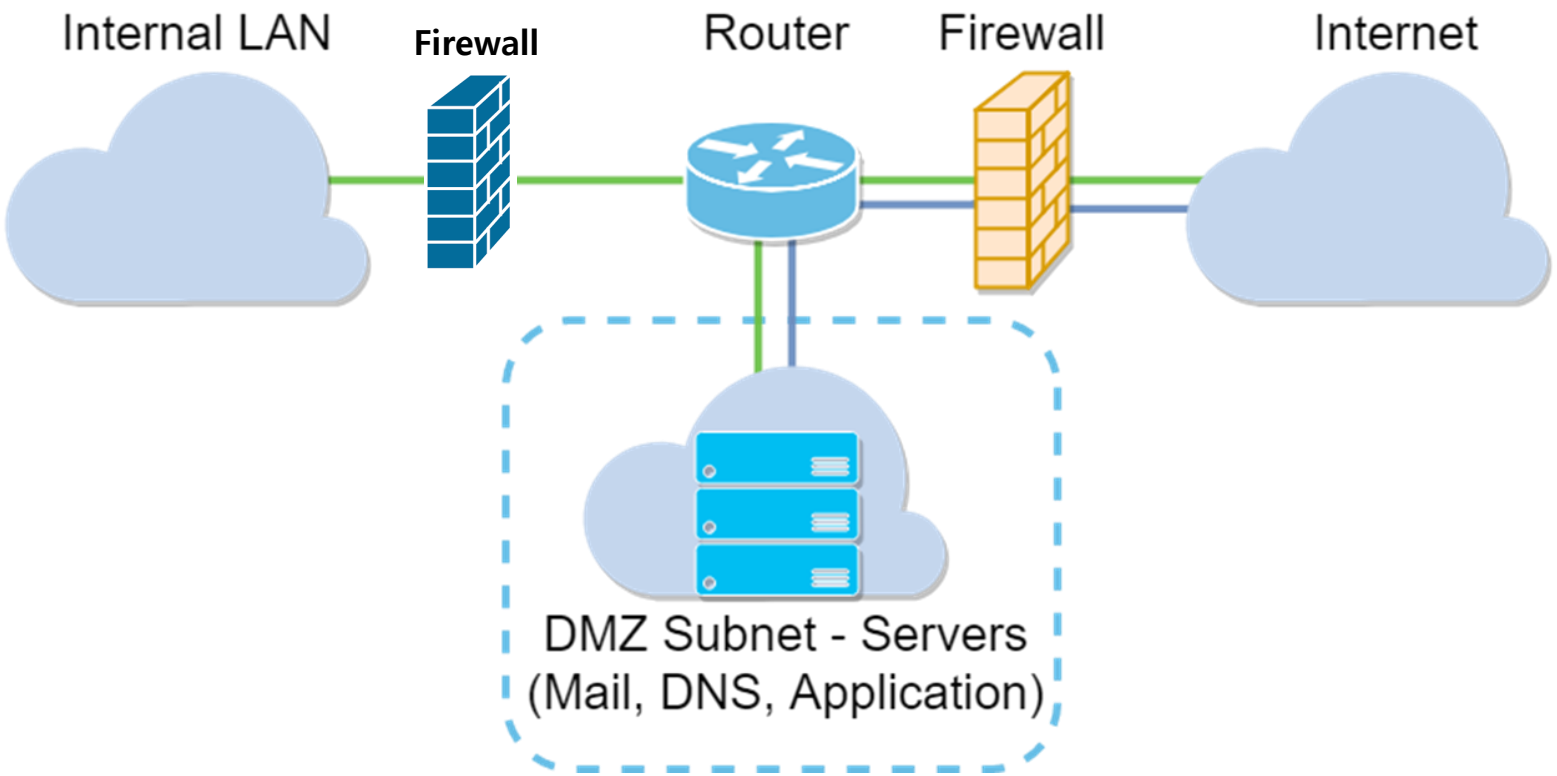
인터넷 네트워크 구성 개요도



(인터넷이 안돼요! 라고 할 때 어디가 문제일까?)

DMZ

- 외부 인터넷을 통해 서비스를 하지만 내부 네트워크로의 침입은 막아야 하는 경우
- 예) 시스코 라우터의 DMZ 구성 옵션
- https://www.cisco.com/c/ko_kr/support/docs/smb/routers/cisco-rv-series-small-business-routers/smb5875-dmz-options-for-rv160-rv260-routers.html



Web server / WAS

Web server / WAS

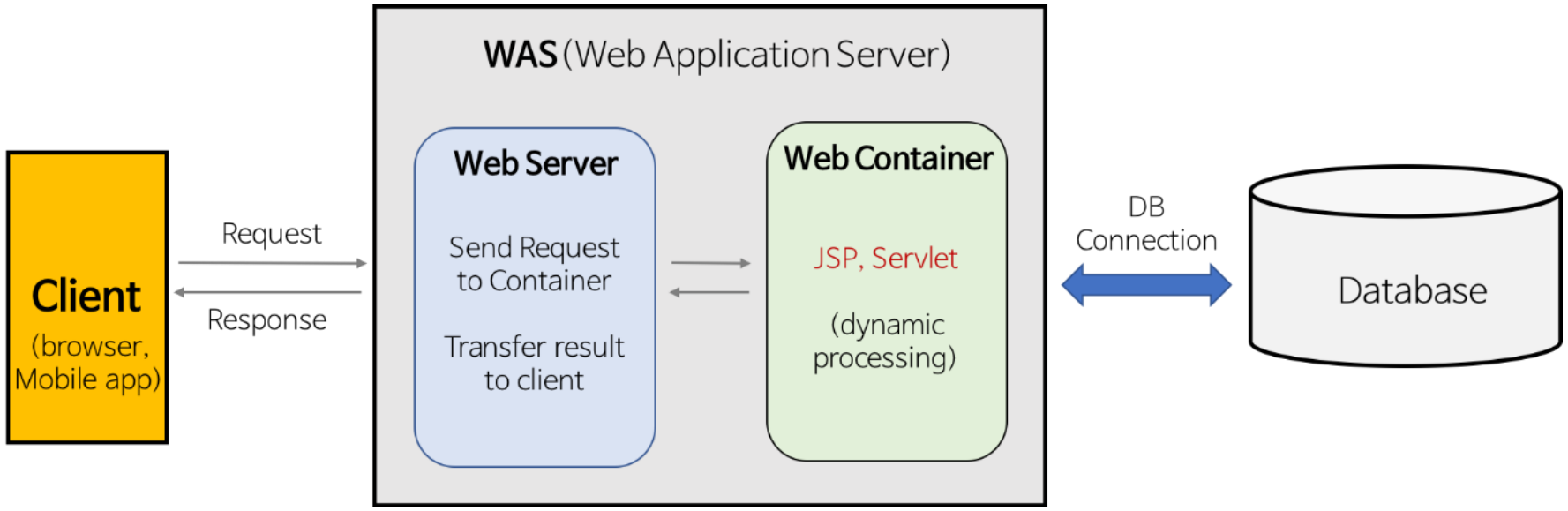
Web server

- 웹을 구동 시키는 서버
- 주로 정적 파일을 서비스 함. (html, css, image 등)
- 주요 제품
  - IIS(Internet Information Server)
    - MS에서 만든 웹서버. Windows Server구매 시 포함 됨. asp/aspx를 서비스하기 위해 사용 함(그 이외에는 잘 안 씀)
  - Apache
    - 아파치 소프트웨어 재단에서 관리하는 오픈소스
    - 최강의 성능
  - Nginx
    - 러시아 개발자가 만든 웹 서버
    - 프록시 역할에 최적(동시 접속 처리가 좋다는 뜻) – 클라우드에서 쓰기 좋다는 말.
    - 사용법은 너무나 간단함. (다만 프록시 개념을 제대로 이해해야 제대로 세팅 가능)

Web server / WAS

WAS (Web Application Server)

- 주로 동적 스크립트인 JSP/Servlet를 처리하기 위한 미들웨어
  - ASP, PHP는 WAS가 없다. (웹서버(IIS, Apache)가 처리 함)
- 예전엔 많은 제품들이 난립 하였으나 Tomcat으로 천하통일 된 이후 다른 제품들은 알 필요가 없어 졌다.
  - Resin, Web Logic, Web Spere, Jeus
- Apache + Tomcat를 연동하기 위해 많은 고난이 있었다. (WAS 단독으로도 서비스가 가능하지만 성능은 보장 못함)
  - OS + Web Server + WAS의 연동 조합은 각각의 경우의 수와 같다.

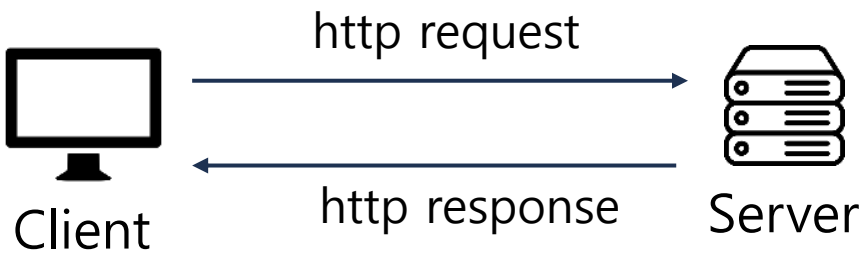


(WAS(혹은 Tomcat)를 쓴다는 말은 java servlet이나 jsp로 되어있다는 뜻.)

HTTP

HTTP(Hyper Text Transfer Protocol)

- 인터넷에서 데이터(html, xml, text, json등)를 주고받을 수 있는 프로토콜 (보통 웹브라우저가 알아서 사용해 줌)
- request, response로 이루어 짐 (요청 -> 응답 -> 끊김)
  - 비연결성(connectionless), 무상태(stateless)
  - 매번 연결 시 마다 서버의 입장에서는 "너 누구니? " 가 됨
    - 이걸 해결하기 위해 session, cookie, storage같은 "나 누군데요 " 기술이 등장
  - 페이지가 로딩 된 후 인터넷을 끊어도 페이지가 유지됨. (ssh, telnet같은 서비스는 연결 끊기면 바로 서비스도 끊김)
- (실습: http://www.uvc.co.kr 접속해서 주소 확인, curl을 통한 http 통신 확인)



번외) 챗봇이 멍청해 보이는 이유

- 비연결성으로 인해 자연스러운 대화는 어렵다 (문맥 유지가 안됨)
  - 사람) 부산가는 KTX 몇 시에 있나요?
  - 챗봇) 2시에 있습니다.
  - 사람) 그럼 티켓 하나 주세요. (부산가는 KTX 티켓 이라는 정보를 개발자가 넣어줘야 함. IBM Watson의 실패)
  - 챗봇) 뭘요?

HTTPS (Http over Secure Socket Layer)

- SSL(Secure Socket Layer)을 Http에 얹은 프로토콜
- 즉, 도청(?)이 가능한 http통신을 암호화 하여 전송하는 기술
- 도청을 막을 순 없지만 암호화 하여 못 알아 보게 하겠다.
  - 양자컴퓨터 나오면 암호체계 다 뚫린다 던데? (8비트(2^8=256) VS 8큐비트(2^8=256))
  - 그건 수학적으로 암호를 푸는 게 아니라 무한대에 가까운 입력 값을 주어서 결과값과 비교하여 맞춰보겠다는 것임
  - 지금도 일반 컴퓨터로 해 볼 수 있으나 시간이 걸린다는 게 함정(몇 만년). 양자 컴퓨터는 수백시간쯤??
- CA(Certificate Authority)라는 인증기관에서 공식적으로 인증 한다.
  - 웹 브라우저는 CA업체를 모두 알고 있다. (자체적으로 만든 SSL은 웹 브라우저가 경고함)
  - 그럼 CA는 누가 인증하나?
- 공개키와 개인키가 있는데 공개키는 공개하는 것이고, 개인키는 노출되면 절대 안된다. (서버에 잘 보관)
  - 공개키로 암호화 → 개인키로만 풀 수 있음 (무궁화 → 암호화 → ABC)
  - 개인키로 암호화 → 공개키로만 풀 수 있음 (이 기능을 이용해서 해당 사이트가 안전한 것인지 확인 해 줌)
  - 공인 인증서와 같은 원리
- (실습: https://www.uvc.co.kr 접속해서 주소 확인, Root인증서 확인)

HTTP Method

- 공식문서: https://www.w3.org/Protocols/rfc2616/rfc2616-sec9.html
- "동사 " 의 역할을 한다. (REST API에서 중요한 개념)
- GET: 리소스를 보여 주세요. (멱등성 O)
- HEAD
- POST: 리소스를 입력해 주세요. (멱등성 X)
- PUT: 리소스를 수정해 주세요. (전부 교체, 멱등성 O)
- DELETE: 리소스를 삭제해 주세요. (멱등성 O)
- CONNECT
- OPTIONS
- TRACE
- PATCH: 리소스를 수정해 주세요. (부분 교체, 멱등성 X) (실업무에서 사용하지 않음, 구현 복잡하고 브라우저에서 지원 안 할 수 있음)
- (실습: Postman에서 GET, POST, PUT, DELETE 확인)

HTTP Status code

- 응답 결과를 코드로 정리한 것
- 많은 상태코드가 있지만 주로 다음의 코드를 많이 사용 한다.
- 2xx: 성공(요청이 성공한 경우)
  - 200: OK
- 3xx: 리다이렉션 (리다이렉트 시키는 경우)
  - 301: Moved Permanently (영구 이동 (요청한 페이지가 영구히 이동된 상태 → 새 위치로 자동 이동 됨))
- 4xx: 요청 오류 (클라이언트가 잘못 요청한 경우)
  - 400: Bad Request (잘못된 요청: 클라이언트의 요청이 무언가 잘못 되어있다.)
  - 401: Unauthorized (권한 없음: 인증이 안된 상태, 즉 로그인이 필요하다)
  - 403: Forbidden (금지됨: 해당 자원에 접근할 권한이 없다.)
  - 404: Not Found (찾을 수 없음: 요청한 자원이 없다. 우리가 자주 보는 Page Not Found)
- 5xx: 서버 오류 (서버에서 에러가 난 경우)
  - 500: Internal Server Error (내부 서버 오류: 서버에서 에러가 발생하였음. 코드가 잘못 된 경우)

개발자 관점

[4xx에러] 니가 잘못한 거. (잘 좀 해라)
[5xx에러] 내가 잘못한 거. (죄송합니다.)

프론트에서는 백엔드의 응답코드와 프론트의 응답 코드를 분리해서 생각해야 한다.

HTML

HTML (Hypertext Markup Language)

- 마크업이란?
  - 원고의 교정부호나 주석 등으로 원문을 어떻게 표현하면 좋을지에 대해 마킹 한 내용
- error(에러)나 exception(예외)가 없다. (프로그래밍 언어가 아님)
- 메모장으로 코딩이 가능하다.
- 웹 브라우저가 해석해서 보여준다.
  - 소스보기를 통해 원문을 볼 수 있다. (사실 원문 이라기보다는 서버에서 만들어서 내려준 "결과 문"이다.)
  - 웹 브라우저의 개발자 도구를 통해 결과를 조작할 수 있다. (캡쳐 화면을 믿지 마라)
  - (실습: 소스보기, 개발자 도구 보기, html 조작)

jQuery

- 사전적 의미: HTML의 클라이언트 사이드 조작을 단순화 하도록 설계된 크로스 플랫폼의 자바스크립트 라이브러리 이다.
- 뜻 풀이
  - 클라이언트 사이드 조작: 웹 브라우저에서 입력이나 출력 값을 조작 할 수 있다.
  - 단순화 하도록: 일반 자바스크립트로 하면 매우 복잡해 진다.
  - 클로스 플랫폼: 브라우저의 종류를 (가능한)가리지 않는다.
  - 자바스크립트: 자바스크립트로 만들어 졌다.
  - 라이브러리: 다운받아서 쓸 수 있다.
- 한 때 엄청 잘 나갔음 (거의 천하 통일)
- 지금은 이를 대처할 front-end기술이 많아져서 줄어드는 추세
  - 클라이언트 화면에서 제어할 로직이 많아지면 거의 난독화 수준으로 코드가 복잡해짐
  - 이게 왜 작동하고, 저건 왜 작동 안 하는지 디버깅이 매우 어려움
  - 개발자가 일일이 다 제어해야 함. (vue.js나 react에서는 제어는 자동이고 개발자는 로직만 짜면 됨)
- (실습: jquery 예제 코드 보기)

CSS (Cascading Style Sheets)

- 사전적 의미: 종속형 스타일 시트
- 실질적 의미: 화면을 예쁘게 꾸미는 방법에 대한 언어
- 퍼블리셔가 반드시 숙지해야 하는 언어 ~~(개발자는 잘 못하는 영역)~~
- CSS를 만들어주는 전처리/후처리 언어가 있다. (Less, Sass, SCSS 등)

```
/**CSS Syntax**/

nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

```
/**Sass Syntax**/

nav
  ul
    margin: 0
    padding: 0
    list-style: none

  li
    display: inline-block

  a
    display: block
    padding: 6px 12px
    text-decoration: none
```

```
/**SCSS Syntax**/

nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }

  li { display: inline-block; }

  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

XML (eXtensible Markup Language)

- 사전적 의미: 다른 특수한 목적을 갖는 마크업 언어를 만드는데 사용하도록 권장하는 다목적 마크업 언어이다.
- 실질적 의미
  - html이 정보를 표시하기에 좋은 것 같은데 html처럼 태그로 정보 표시를 한번 해보자.
  - 자료 구조로써 써보자.
  - 자료 구조로 써 보기엔 html 태그가 너무 한정적이니 태그를 한정 짓지 말고 사용 하자.
- 서버끼리 데이터 주고 받을 때 사용 함
  - 요즘은 json을 더 많이 사용 함.
- 서버 환경 설정할 때 많이 사용 함.
- (실습: xml예제 코드 보기, 만들어 보기)

MARKDOWN

- 말장난? 마크업을 다운 시킨 언어
- html에서 주로 사용하는 것을 보다 단순화 해서 표현해줄 수 있는 언어
- 웹 브라우저에서 해석하지 않음 (git에서 README.md 파일이 마크다운으로 작성됨)

값	의미	기본값
`static`	유형(기준) 없음 / 배치 불가능	`static`
`relative`	요소 **자신**을 기준으로 배치	
`absolute`	위치 상 ***부모***(조상)요소를 기준으로 배치	
`fixed`	**브라우저 창**을 기준으로 배치	

markdown

```
| 값 | 의미 | 기본값 |
|---|:---:|---:|
| `static` | 유형(기준) 없음 / 배치 불가능 | `static` |
| `relative` | 요소 **자신**을 기준으로 배치 | |
| `absolute` | 위치 상 ***부모***(조상)요소를 기준으로 배치 | |
| `fixed` | **브라우저 창**을 기준으로 배치 | |
```

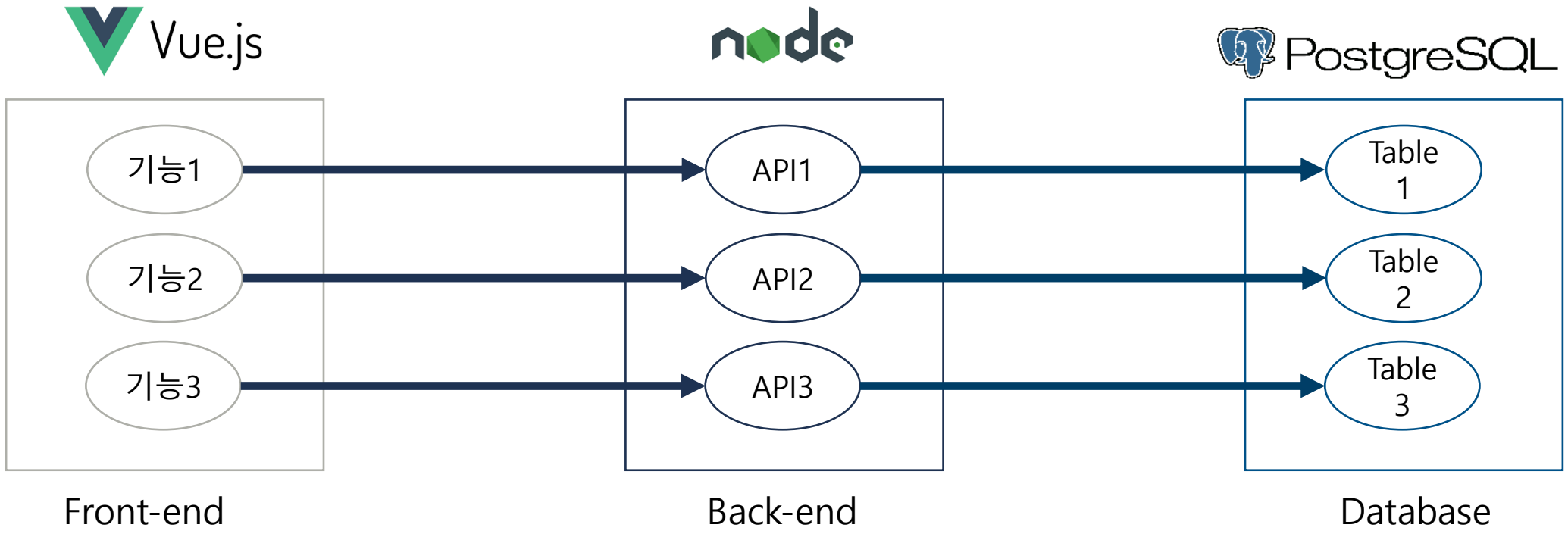
html

```
<table>
   <thead>
      <tr>
         <th>값</th>
         <th style="text-align:center">의미</th>
         <th style="text-align:right">기본값</th>
      </tr>
   </thead>
   <tbody>
      <tr>
         <td><code>static</code></td>
         <td style="text-align:center">유형(기준) 없음 / 배치 불가능</td>
         <td style="text-align:right"><code>static</code></td>
      </tr>
      <tr>
         <td><code>relative</code></td>
         <td style="text-align:center">요소 <strong>자신</strong>을 기준으로 배치</td>
         <td style="text-align:right"></td>
      </tr>
      <tr>
         <td><code>absolute</code></td>
         <td style="text-align:center">위치 상 <strong><em>부모</em>(조상)요소</strong>를 기준으로 배치</td>
         <td style="text-align:right"></td>
      </tr>
      <tr>
         <td><code>fixed</code></td>
         <td style="text-align:center"><strong>브라우저 창</strong>을 기준으로 배치</td>
         <td style="text-align:right"></td>
      </tr>
   </tbody>
</table>
```

Front-end

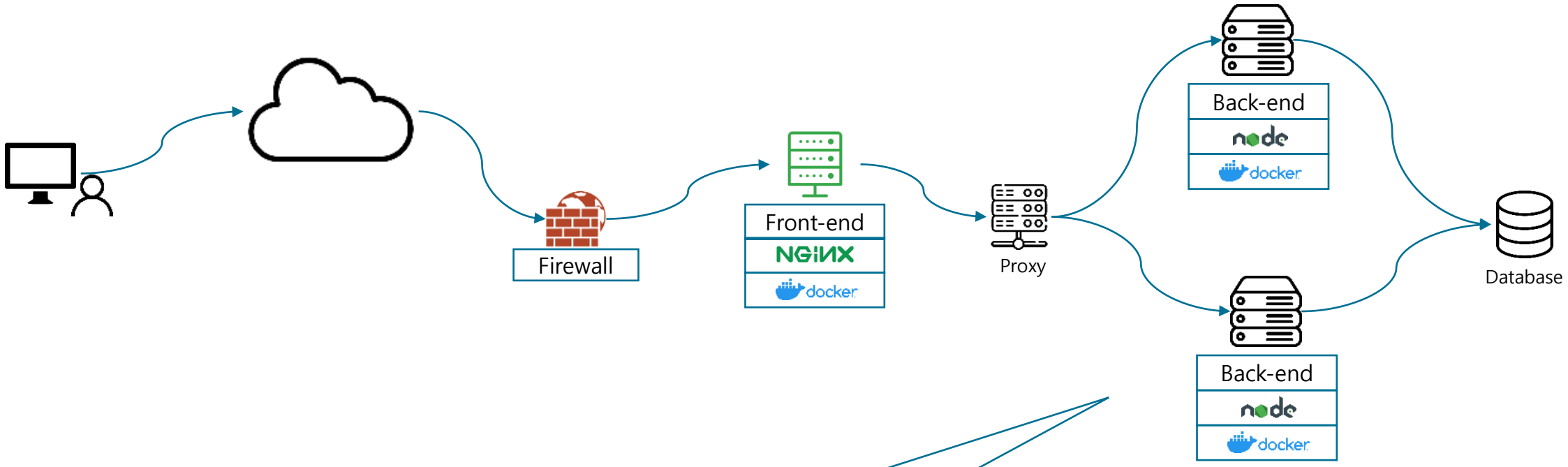
기본 아키텍처 (Front-end, Back-end, Database)

- 예전엔 Front-end와 Back-end가 하나의 스크립트(asp, jsp, servlet, php)안에 모두 포함되어 있었다.
- 왜 분리 했을까?



Front-end

시스템 구성도



비즈니스 로직의
분산 처리 가능
손쉬운 scale in/out
MSA 구조 가능

MSA(MicroService Architecture)란? 비즈니스 로직을 레고처럼!

Front-end

UVC
Ubiquitous Via Connect

SPA(Single Page Application)

- 매번 페이지를 Reloading하면 해당 페이지의 html을 계속 다운로드 해야 하는 서버/클라이언트 측의 부담이 생긴다.
- 미리 템플릿을 만들어 놓고 데이터만 바꾸면 어떨까?
- html구조에서 바뀌는 부분만 Rendering 하면 어떨까?
- 주로 vue.js, react, Angular로 만든다.
- 처음 접속시에는 약간 느리지만 이후 클릭하면 굉장히 빠르게 화면 전환이 이루어 진다.
- 브라우저를 reload 하게 되면 처음 접속한 것처럼 느리게 반응 한다.

Angular JS(Angular1), Angular(Angular2) - 버전 별 이름이 이상해 졌다.

- Angular JS는 학습 비용이 매우 크다. (어렵다는 뜻)
- Angular(Angular2)는 Angular JS와 사실상 아무 관계가 없다. (그냥 새로운 언어임)
  - Angular를 그냥 Angular2라고 부르고 Angular JS는 Angular1이라고 부르기도 한다.
  - Angular2는 쓸 만 하지만 Angular JS에 데인 개발자들은 이미 변심한 상태(React, Vue.js)
- (실습: Angular 소스코드 보기)

React

- 학습 비용이 매우 크다.
- 코딩 양이 매우 많다.
  - 체감 상 동일한 로직 처리하는데 vue.js의 세배 정도 코딩해야 하는 느낌.
- 일반적으로 알고 있는 자바스크립트 대로 동작 하지 않는다.
  - 변수 선언 및 사용법부터 차원이 다르다. (변수 선언이 그냥 안되고, 사용도 그냥 못한다.)
- 시중에 압도적으로 많이 배포되어 있다. (개발자가 많다)
- (실습: React 소스 보기)

Vue.js

- 학습 비용이 적다.
- Document가 잘 되어 있다.
- React보다 쉬운 것 같다.
- React보다 코딩 양이 적은 것 같다.
- (실습: vue.js 소스 보기)

Bootstrap

Bootstrap

Bootstrap

- 웹사이트를 예쁘게 만들어 주는 프레임 워크.
- 사용자의 입력 폼(input, select, radio, checkbox, textarea)이나 버튼 등을 예쁘게 만들어 준다.
- 화면 구조를 잡아 주기 쉽다.
    - div row를 만들 수 있다. (div는 가로 전체를 뜻함)

Bootstrap-vue

- bootstrap을 vue.js에서 사용할 수 있도록 했다.
- (실습: bootstrap-vue 사이트 방문)