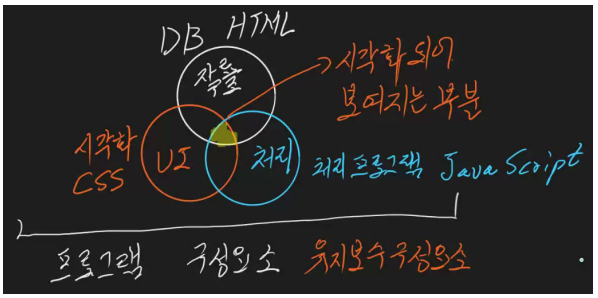
[10/24] Stateless, Stateful

- 프로젝트 PPT 예시 확인
  - Usecase Diagram, 예외처리(시험처리,시나리오에 대한 검증), WBS(Work Board Sheet), TDD(Test Driven Development)

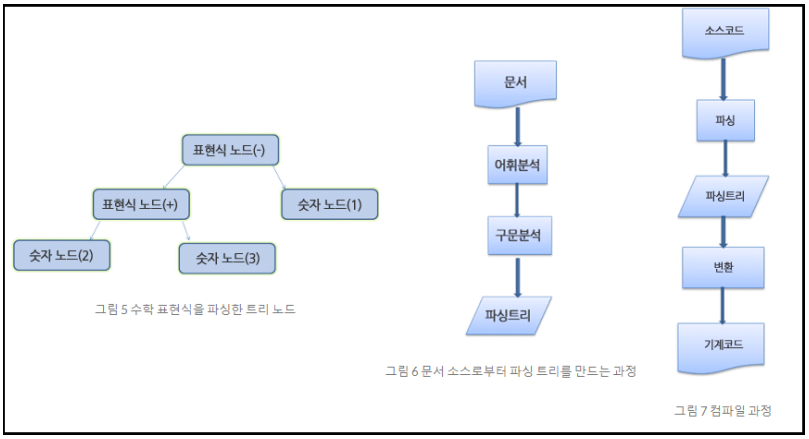
- 프로그램 구성 요소(유지보수 구성 요소)



  - HTML (Hypertext Markup Language) - Internet - HTTP(HyperText Transfer Protocol)
  - 내용, 자료 구조, 태그 - 사용자 환경(UI) - 전송, 처리 규칙

- internet
  - Web 1.0 : 초기의 인터넷
  - Web 1.1 : 현재
  - Web 2.0 : IPv6 시대
  - Web 3.0 : metaverse, 데이터의 소유주가 개인에게로

- Web Browser(Client) <-> Web Server(Server)
  - Stateless[1] : 초기 웹, 정적 연결, 수신만함(단향), 1.0 Age …ex) 웹 검색
    - Web Browser : http.request.method == get '요청값,리소스'
      - get : Web Server(DB)에 있는 내용을 Read
    - Web Server : http.response.
    - 구문 분석(parsing) : HTML+CSS+image
      - 파싱은 브라우저가 코드를 이해하고 사용할 수 있는 트리 구조로 변환하기 위해 토큰 형태로 분해

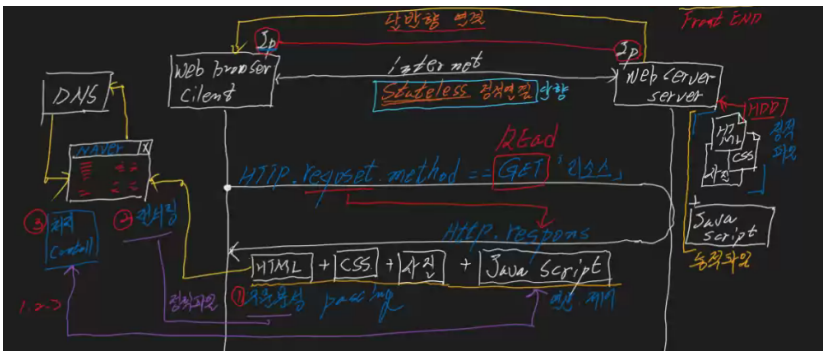


    - Rendering : 요청한 콘텐츠 표시, HTML 요청시 HTML과 CSS를 파싱하여 화면에 표시함.[2]

---

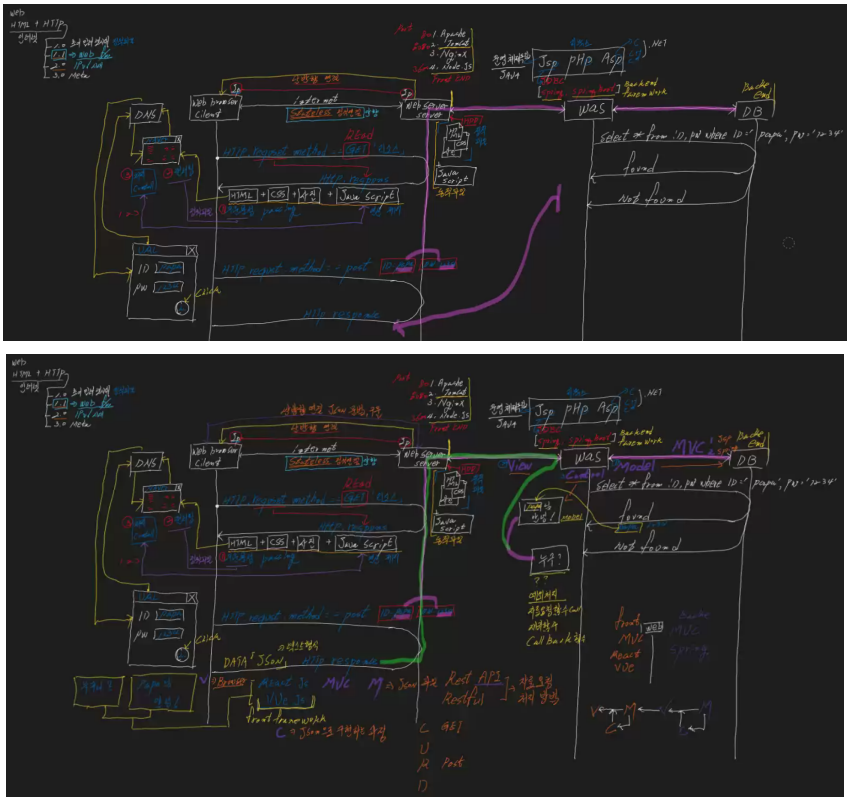
[1] Difference between stateless and stateful protocols / Stateful/Stateless

[2] 브라우저는 어떻게 동작하는가? - NaverD2

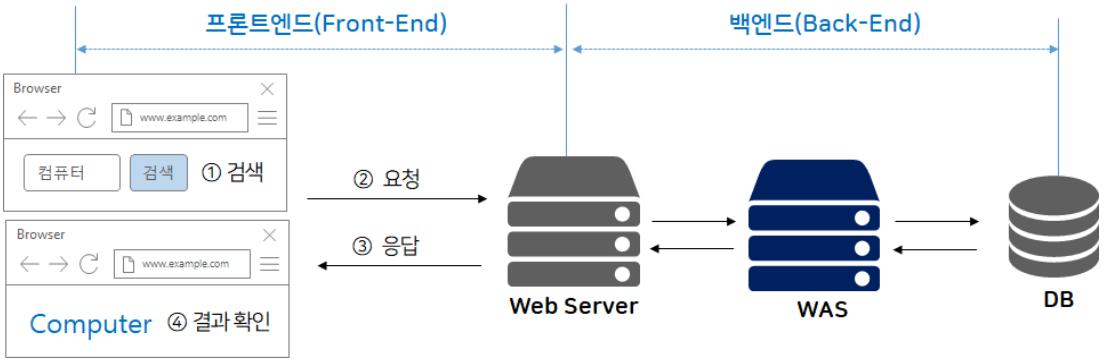


Web 1.0 , Stateless

- Stateful : 동적 연결, 양방향으로 문법과 구문 필요 …ex) 상품 주문
  - Event : 시스템에서 일어나는 사건, 발생
  - Web Browser : http.request.method == post ID=abc, PW=1234
  - Web Server : http.response
    - WAS : select * from ID where ID='abc', PW=1234
    - DB : found or not found
      - found : WAS가 Web Server로 전송
      - not found : 예외처리, 재귀함수, callback
      - C(Post, insert), R(get, select), U(post, update), D(delete)
  - DB를 **M**odeling 하여 **V**iew로 송출되도록 **C**ontrol(MVC) : Spring
    - MVC Model : Model(DB), View(Web server), Control(Spring)
  - Web Server에서 전송된 Data를 JSon 이 텍스트 형식 :
    React or Vue(Front Framework)가 재조립 및 렌더링
    - MVC Model : Model(Json file, Rest API, Restful), View(Browser), Control(Json)
      - 자료 요청 처리 방법 : Rest API, Restful



Web 1.1 , Stateful



	Stateless	Stateful
정의	클라이언트가 서버로 요청을 보내고, 주어진 상태에 따라 서버의 응답을 다시 보내주는 네트워크 프로토콜	클라이언트가 서버에 요청을 보냈을 때, 어떤 종류의 응답을 기대하고 만약 응답이 없을 경우에는 다시 요청을 보내는 네트워크 프로토콜
예	HTTP	FTP(파일 전송 프로토콜), TCP
서버 제한	서버 정보나 세션의 세부 정보를 자체적으로 유지하고자 할때 서버가 필요하지 않다.	현재의 상태와 세션 정보를 유지하기 위해서 서버가 필요하다.
의존성	서버와 클라이언트가 느슨한 결합으로 이루어져 있기 때문에, 서로 독립적으로 작동할 수 있다.	상태 저장이 필요하므로 서버와 클라이언트가 밀접하게 연결될 수 밖에 없다.
디자인	서버 설계가 구현하기에 간단하다.	설계 디자인이 복잡하고, 구현하기도 상대적으로 더 어렵다.
충돌 관리	서버 장애가 발생했더라도, 출돌 후 다시 쉽게 시작이 가능하다.	서버가 세션 및 여러 세부 정보들을 유지해야 하기 때문에, 충돌 관리가 더 어렵다.
트랜잭션	서버에서 트랜잭션을 빠르게 처리할 수 있다.	서버가 비교적 느리게 작동한다.

- Spring, Spring Boots : 가상머신(ex; JVM) 위에서 구동되는 App 들은 OS와 상관없는 높은 호환성을 보유하도록 유도, 해당 app 들의 묶음을 위해 등장
    - OS와 상관없는 : JSP는 java로, PHP는 laravel, ASP는 C,C++ 에서만 구동되었다.
    - Docker[3] : JSP & Spring

- Web Server
    - Apache : 정적
    - Tomcat : 동적
        - PHP, ASP, JSP

[3] 도커와 쿠버네티스 간단 비교