웹애플리케이션의 이해

동의과학대학 컴퓨터정보과
김진숙

학습내용

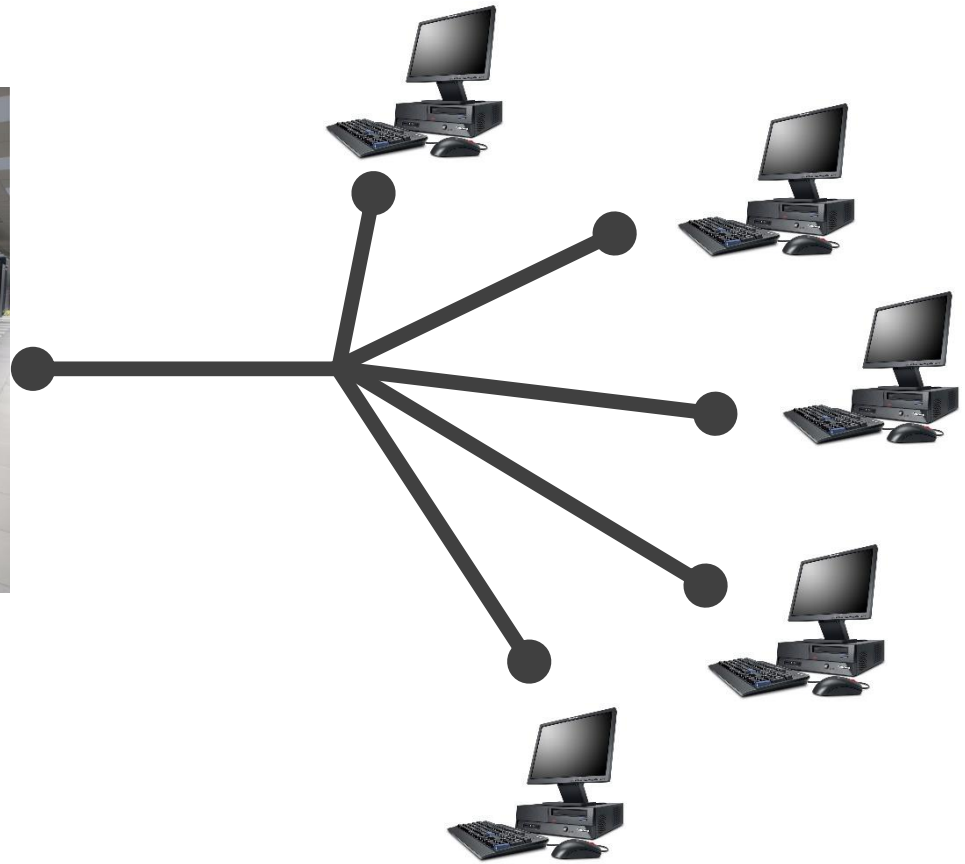
- 웹 애플리케이션이 나오기까지...과거와 현재의 기술 경향
  - Mainframe 시대
  - Client-Server Application 시대
  - Web Application 시대
  - N-Screen and Cloud 시대

[출처]
엄진영, 열혈강의 자바 웹 개발 워크북, 프리렉, 2016. pp. 839~854

Mainframe 시대(~1990년대)

- 대용량의 메모리와 고속의 처리속도를 지닌 멀티 유저용 대규모 컴퓨터에 터미널을 연결하여 사용

메인프레임
- 데이터 처리 로직
- 비즈니스 로직
- 화면 출력 로직



터미널
- 메인프레임과 연결 통신장비, 모니터,키보드로 구성
- 자체 연산 기능 없고 사용자 입출력만 수행

Mainframe 시대(~1990년대)

- Mainframe 특징
  - **중앙 집중형** : 모든 연산이 메인프레임에서 이루어짐
  - **보안 용이** : 메인프레임에 연결된 터미널을 통해서만 시스템 사용
  - 소프트웨어 **유지보수 용이**
    - SW가 메인프레임에 설치됨으로 기능 추가나 변경 시 메인프레임에만 배포하면 됨
  - 하드웨어 교체에 따른 **유지보수비용이 높은 구조**
    - 사용자, 사용량이 계속 증가 상황에서 하드웨어를 교체하는데 막대한 비용 발생

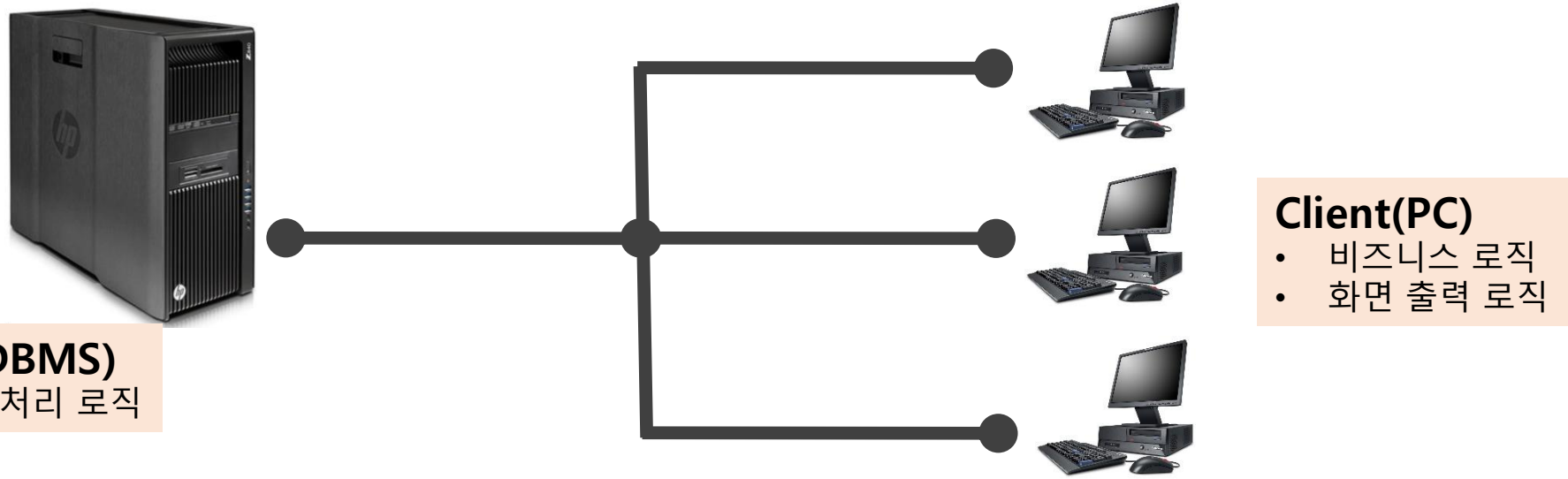
- 개발 도구(언어)
  - Cobol
    - 미 국방성에서 사무처리 언어 통일을 위해 1959년 개발됨
    - 국내에서도 1990년대 중반까지도 은행 시스템의 주요 개발언어였음
  - Fortran
  - PL/1
  - C

Mainframe 시대(~1990년대)

- 개발 프로세스
  - 절차적 프로그래밍이나 구조적 개발 방법론
  - Top-Down 방식
  - Function 중심 개발 방식이 사용됨

Client/Server 시대(1990 중반~2000년대)

- 배경
  - 하드웨어 사양의 향상으로 예전 메인프레임급 수준의 **서버용 컴퓨터**를 저렴하게 구축할 수 있게 됨
  - **LAN**(Local Area Network)과 **WAN**(Wide Area Network)으로 컴퓨터들이 연결되기 시작
  - 국내에서는 1990년대 중반 인터넷이 보급되기 시작
  - 메인프레임이 하던 일 중 **비즈니스 로직**과 **화면출력**부분을 개인용 컴퓨터(PC)가 담당
  - **데이터 처리 로직**은 워크스테이션 수준의 소형 서버가 담당



Server(DBMS)
- 데이터 처리 로직

Client(PC)
- 비즈니스 로직
- 화면 출력 로직

Client/Server 시대(1990 중반~2000년대)

- Client/Server 시대의 특징
  - 윈도우 프로그래밍(텍스트 기반에서 윈도우로)
    - 1985년 11월 Windows 1.0이 정식 릴리즈됨
    - 1990년에 발표한 Windows 3.0이 대대적인 성공을 거두어 PC사용자 환경이 **GUI**로 변경
  - 인터넷 보급(1995년부터 World Wide Web서비스 시작)
    - 국내에서는 천리안 서비스가 1994년 **인터넷에 연결**되었고 1995년부터 WWW 서비스 시작
    - 우리나라 인터넷의 아버지 : 전길남 (https://www.youtube.com/watch?v=ekw2ljVPicA&ab_channel=%EB%94%94%EC%94%A8%EB%A9%98%ED%84%B0%EB%A6%AC)
  - 서버 구축 및 운영비 절감
    - 서버의 기능 일부를 PC가 가져가면서 서버의 부하 감소
    - 서버 구축비나 유지보수 비용 감소
  - 회사 홈페이지 구축 유행
    - 인터넷의 보급으로 게시판, 방명록 등과 같은 기능을 중심으로 웹 환경의 CGI 프로그래밍이 구축되기 시작

Client/Server 시대(1990 중반~2000년대)

- 개발 도구
  - 윈도우 프로그래밍을 위한 4세대 개발 도구의 사용
    - Server
      - CS 환경에 맞춰 오라클, 인포믹스, 사이베이스 DBMS
    - Client
      - 파워빌더, 델파이, 비주얼베이직 등 윈도우 애플리케이션
  - PHP, ASP, Perl 등의 CGI 스크립트
    - CGI(Common Gateway Interface) 프로그래밍 초기에는 C 언어가 사용되었으나 불편함으로 인해 인터프리터 언어(PHP, ASP, Perl)가 주류로 떠오름
- 개발 프로세스
  - 경영 정보 개념 도입
  - CASE(Computer Aided of Software Engineering) 도구를 활용한 개발 자동화 시작

Client/Server 시대(1990 중반~2000년대)

- Client/Server 단점
  - 소프트웨어 갱신 및 재배포가 불편
    - PC에 애플리케이션을 설치하는 방식이어서 기능 추가나 변경될 때마다 기존의 프로그램을 제거하고 다시 설치해야 하는 불편함 있음
  - 데이터베이스 접근 보안에 취약
    - 클라이언트에서 DBMS에 직접 접근하기 때문에 접근 아이디, 비번에 노출될 경우 최악의 상황이 발생할 수 있음
    - 네트워크 상에 데이터 전송을 위한 구현이 어려움(소켓, RPC 등)

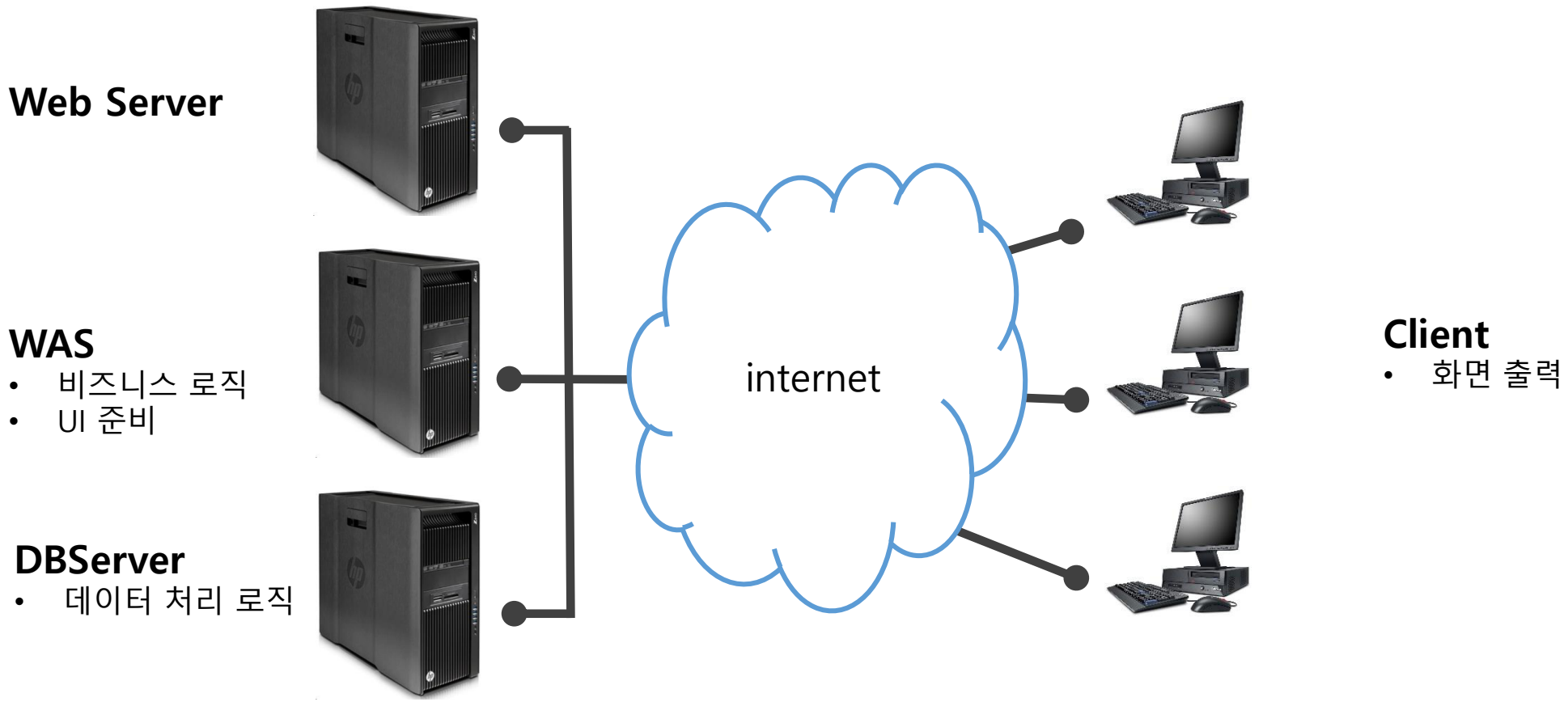
→ 웹이 이러한 문제를 해결할 수 있는 환경을 가지고 있음

Web 시대(2000~2010년대)

- 배경
  - 고사양의 하드웨어 개발
  - 초고속 인터넷 보급
    - 1998년부터 초고속 인터넷으로 인터넷 활용이 더욱 활성화 됨
    - 2000년대는 단순 홈페이지 수준을 넘어서 업무용 애플리케이션이 실행되는 플랫폼으로 역할 함
    - 인터넷이 글로벌 비즈니스 경향과 맞물려 산업전반에 걸쳐 업무용으로 사용되기 시작

Web 시대(2000~2010년대)

- Web 구조
  - Client/Server 프로그램을 **웹의 환경 구조** 내에서 사용



Web 시대(2000~2010년대)

- Web 시대의 특징
  - 글로벌 비즈니스의 가속화로 **시스템의 잦은 변경**
    - 세계화에 따른 글로벌 비즈니스의 무한경쟁으로 제품, 서비스 출시 주기가 짧아져 업무지원 시스템이 자주 변경
  - 웹애플리케이션 서버(WAS)의 도입
    - 업무 변화가 많은 경우 애플리케이션이 계속 재설치 해야 하는 C/S 방식은 적합하지 않음
    - PC가 처리하던 업무를 다시 서버로 이전
      - 여러 개의 소형 서버로 역할을 분산 배치
      - 데이터 처리는 서버, 비즈니스 로직은 전용 애플리케이션 서버에서 처리

Web 시대(2000~2010년대)

- 웹 기술의 활용
  - **웹 브라우저**
    - PC(클라이언트)는 웹 브라우저를 통해 웹 서버에게 작업 요청
    - 운영체제에 독립적인 UI 구현(HTML, CSS, Javascript)
  - **웹서버**
    - PC(클라이언트)와의 연결처리를 웹서버가 처리
    - 애플리케이션 서버에 작업을 위임
  - **웹 애플리케이션**
    - WAS에서 구동되는 애플리케이션을 웹애플리케이션이라고 함
    - 애플리케이션 서버는 요청을 처리할 애플리케이션을 찾아 실행하고 그 결과를 웹서버에 전달
  - **애플리케이션의 배포가 용이**
    - 모든 작업은 서버에서 실행되기 때문에 기능이 추가되거나 변경될 때 서버에만 배포하면 되기 때문에 애플리케이션 유지보수가 용이

Web 시대(2000~2010년대)

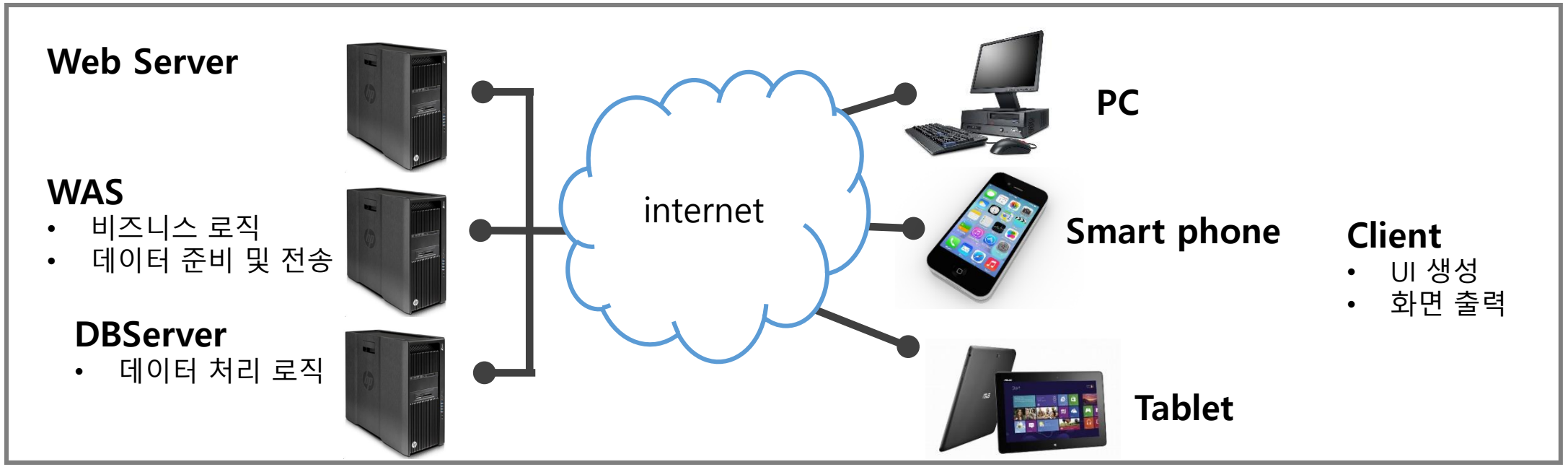
- Web 시대의 장점
  - 네트워크 프로그래밍으로부터 탈출
    - 서버와 PC간의 데이터 전송을 **웹서버와 웹브라우저가 담당**하여 개발자는 더 이상 네트워크 프로그래밍을 할 필요가 없어짐
    - 웹서버가 클라이언트와의 연결을 처리함으로 여러 클라이언트의 동시 연결 관리를 위한 멀티프로세싱, 멀티 스레딩 처리를 할 필요가 없어짐
  - 운영체제에 독립적인 UI 생성
    - 표준 웹 기술을 사용하여 UI를 만들면 어떠한 OS에서도 실행 가능
  - 애플리케이션 배포 용이
    - 모든 작업은 서버에서 실행되기 때문에 기능이 추가되거나 변경될 때 서버에만 배포하면 됨
    - PC에 재설치 하는 일이 없음

Web 시대(2000~2010년대)

- Web 시대의 단점
  - 네트워크 오버헤드 증가
    - **중앙 처리 방식**이므로 작업 요청 시 매번 서버에서 화면 데이터를 받아 옴
    - 사용자가 많을 수록 네트워크 부하가 많이 걸림
- 개발 도구
  - 서버 측 개발 : PHP, Python, ASP, Servlet/JSP
  - 클라이언트 측 개발 : HTML, CSS, Javascript
- 개발 프로세스
  - 객체 중심 개발(OOP)에서 컴포넌트 중심 개발(CBD)로 진화
  - Agile 방법론 대두
    - 소프트웨어 개발 방법에 있어서 아무런 계획이 없는 개발 방법과 계획이 지나치게 많은 개발 방법들 사이에서 타협점을 찾고자 하는 방법론
    - 문서를 통한 개발 방법이 아니라, code-oriented, 실질적인 코딩을 통한 방법론
    - 일정한 주기를 가지고 끊임없이 프로토 타입을 만들어내며 그때 그때 필요한 요구를 더하고 수정하여 하나의 커다란 소프트웨어를 개발해 나가는 적응적 스타일의 방법론
    - 개발 방법론 : 익스트림 프로그래밍, 스크럼 등

N-Screen과 cloud 시대(2010년 ~)

- 배경
  - 2000년대 말에 등장한 스마트폰을 시작으로 스마트 패드, 스마트 워치, 스마트 TV 등 다양한 종류의 디바이스 등장
  - 서비스로서의 소프트웨어(SaaS : Software as a Service), 서비스로서의 플랫폼(Platform as a Service)와 같이 SW나 HW를 구축하지 않고 임대하여 사용하는 방식이 인기를 끔



N-Screen과 Cloud 시대(2010년 ~)

- 특징
  - 리치 인터넷 애플리케이션
    - 사용자 행위에 역동적이고 기민한 반응을 제공하는 웹 애플리케이션을 RIA(Rich Internet Application)라고 부름
    - HTML5에 기반한 앱을 지칭하기도 함
  - 다양한 디바이스와 해상도에 대응할 수 있는 애플리케이션 요구
    - 기기에 상관없이 일관성 있는 사용자 인터페이스 제공이 중요
  - 클라우드 스토리지
    - 드롭박스, 구글드라이브, 원드라이브, icloud, ndrive 등
    - 파일 동기화
  - 클라우드 서비스
    - 소프트웨어 임대 방식(SaaS : Software as a Service)
    - 플랫폼 임대 방식(리눅스서버, 윈도우 서버, DB 서버 등)
    - 임대방식의 문제는 중요 정보가 외부 기관에 의해 노출될 수 있다는 것
    - 국가 기관이나 큰 기업에서는 자체 서버를 구축하여 사용

웹애플리케이션 아키텍처

웹서버
- 클라이언트와 통신
- 다중 클라이언트 접속관리 (소켓, 쓰레드 등)

위임

결과

WAS
- 비즈니스 로직
- 데이터 관리
- 사용자 접근 관리
- UI생성

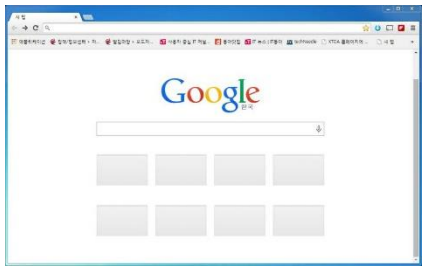
요청

응답

데이터

SQL 질의

웹브라우저
- 사용자와 상호작용
- UI 렌더링

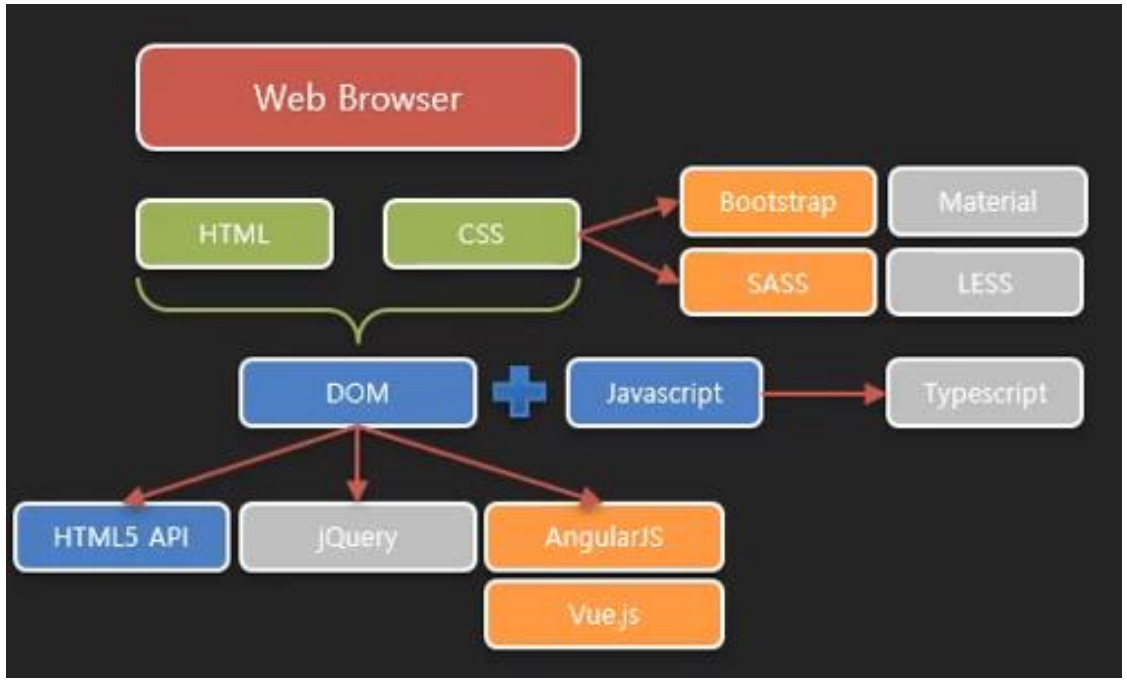


DBMS 서버
- 데이터 처리

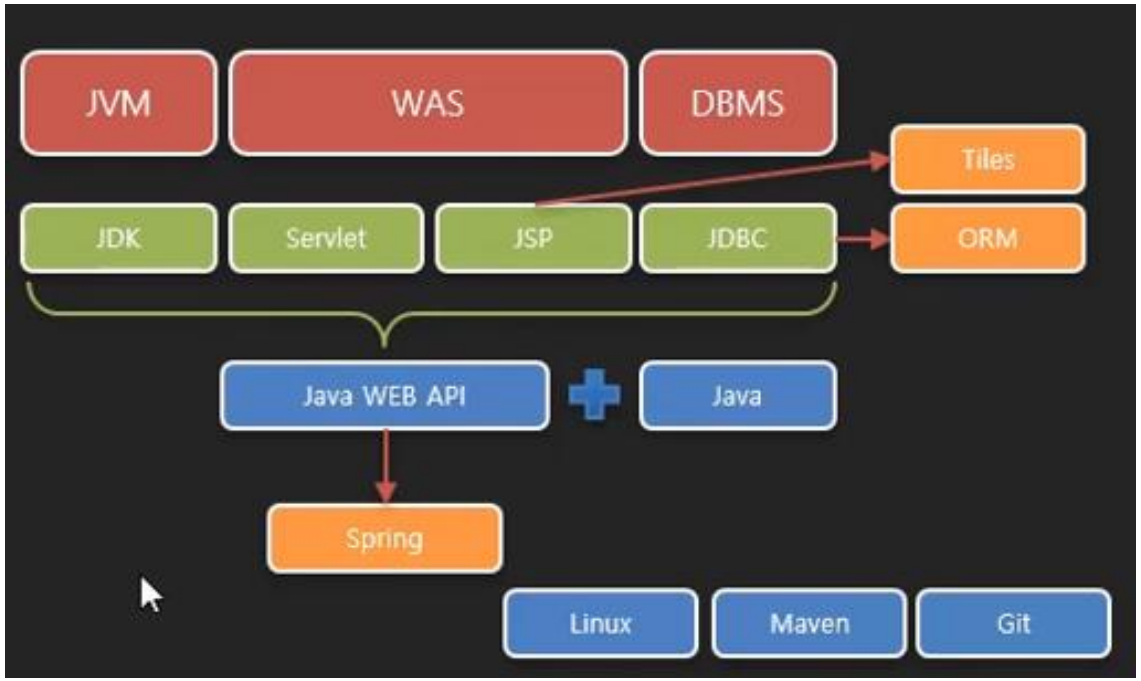
- 웹기술 도입으로 소켓, 멀티스레드 프로그래밍을 웹서버에서 처리
- 서버에 애플리케이션 배포 및 실행으로 기능 변경, 추가 용의

웹애플리케이션 구현

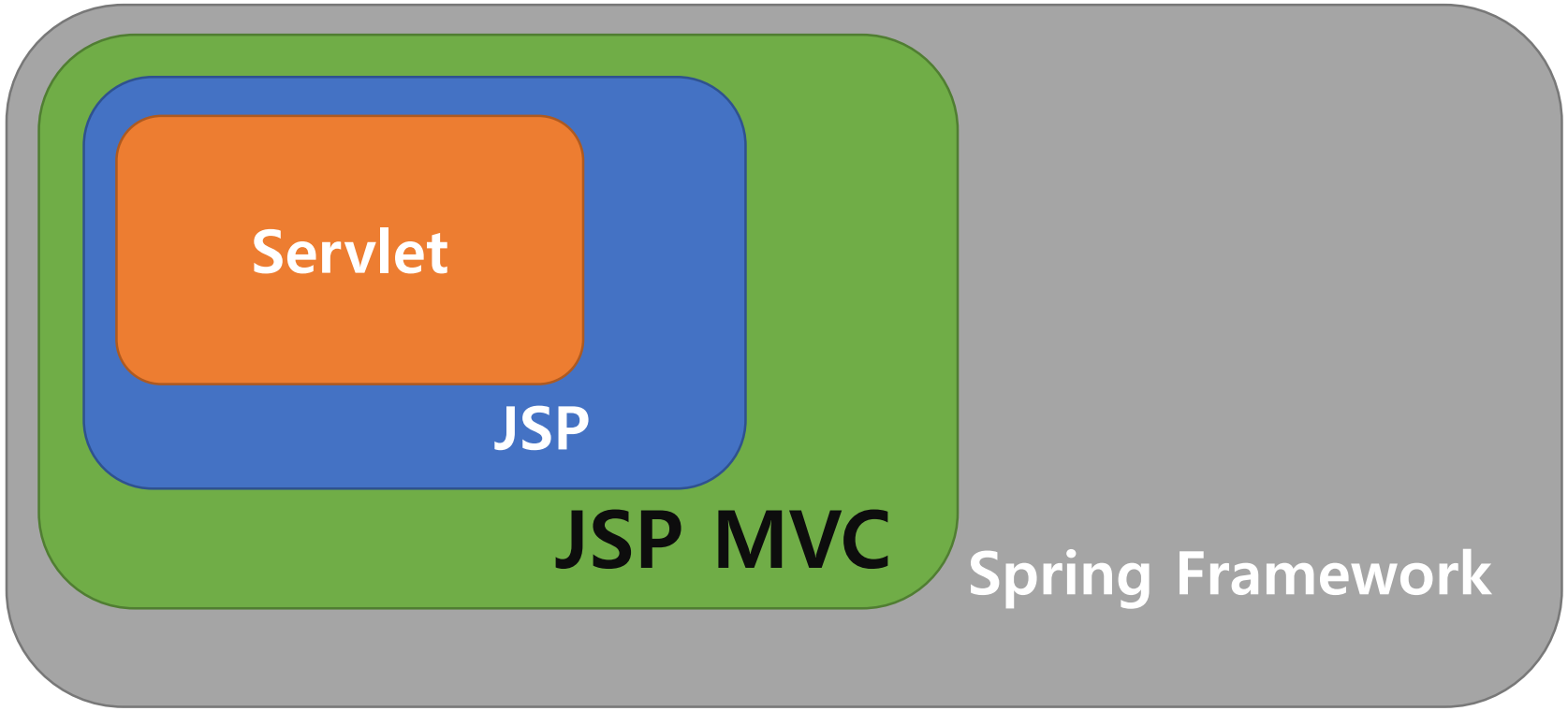
Front-End 개발자



Back-End 개발자



자바 웹 프로그래밍



Servlet : HTML 코드 출력 문제
JSP : 코드 유지보수 문제