

스타트업 개발자를 위한

노드 백엔드 실무 속성 교육 과정

핸즈온랩 과정

JS

지니공공아카데미 강창훈
엠소프트웨어 대표

JINY
ACADEMY

T TimeToDev



개발은 처음이라

1)구글 드라이브 자료실

<http://bit.ly/3h95yKd>

2)실시간 소통 채널:화상통화

<https://meet.google.com/rsw-rudw-uvx>

3)WIFI

중계기: **beginmate**

암호:

4)Github Repository Address

<https://github.com/jinykang/nodefastdev.git>



엠소프트웨어 대표 강창훈

- 풀스택 개발자, SW Architect, 개발경력 22년
- Microsoft AI MVP(2019-2020-2021) Award 수상
- 도서 "인공지능 챗봇 개발 서비스 하기" 2018 저자
- 도서 "ASP.NET MVC5 반응형 웹사이트 개발 서비스하기" 2017 저자
- 융합기술정보제공 및 개발자플랫폼 "TimeToDev" 개발운영(베타 서비스 중)

- <https://jiny.academy>

- <http://www.timetodev.co.kr>

- <https://www.facebook.com/groups/1565990660262587>

- 연락처: 010-2760-5246

- 메일주소: master@jiny.academy
ceo@msoftware.co.kr

주요 커리큘럼

순번	주제	비고
1	개발환경 확인	1Hour
2	Node Express 소개 및 프로젝트 구성하기	
MVC 패턴 이해		
3	MVC 의 Controller – Node Routing	1Hour
4	MVC 의 View – Node View Engine	
ORM 이해		
5	MVC 의 Model – Node Model ORM DB Programming	1Hour
6	Authentication & Authorization (사용자 로그인 시스템 구현)	1Hour
프론트엔드/백엔드 JWT 토큰 기반 인증 프로세스 이해		
7	JWT Token 인증 서버 구축하기	1Hour
8	File Upload	
	CORS & DOT ENV	
	Node.js 분산 시스템 구축하기	

웹 프로그래밍 기초

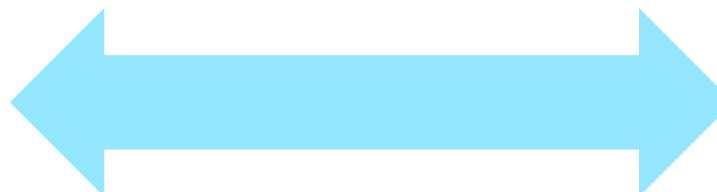
클라이언트 vs 서버

클라이언트 vs 서버



요청하는 서비스(웹페이지,메일,게임)
(웹브라우저,메일,게임,SNSApp..)서비스 요청 SW
OS(Windows10,MAC,IOS,Android,Ubuntu(Linux))
서비스 요청 컴퓨터/H/W-데스크탑/노트북/모바일폰
서비스 요청자-사용자,나,너,우리

CLIENT



SERVER

제공하는 서비스(웹페이지,메일,게임...)
(웹서버,메일서버,메시징서버,게임서버)서비스 제공 SW
OS(Linux(Centos,Ubuntu..),Windows Server)
서비스 제공 컴퓨터 H/W-데이터센터/전산실../용산..
서비스 제공자 :회사/기관/.../서비스 제공업체

Client Side and Server Side Environments



Client Side 환경

Server Side 환경

정적 웹페이지와 동적 웹페이지

웹 브라우저 와 웹 서버의 용도



HTML,CSS,JavaScript 가 보관되고 배포되는 중앙저장소가 웹서버이고 실제 해당 소스가 사용되고 작동되는 런타임 환경은 웹 브라우저이다.

정적 웹사이트 와 동적 웹사이트

- 정적 웹사이트/웹페이지

- 정적 웹사이트와 정적 웹페이지란? : HTML페이지 내용이 동적으로 변경이 안되는 웹페이지, 웹사이트
- 대표적인 정적 웹페이지들(회사소개, 대표이사 인사말, 서비스소개)
- 정적 웹페이지 구성기술들: HTML,CSS

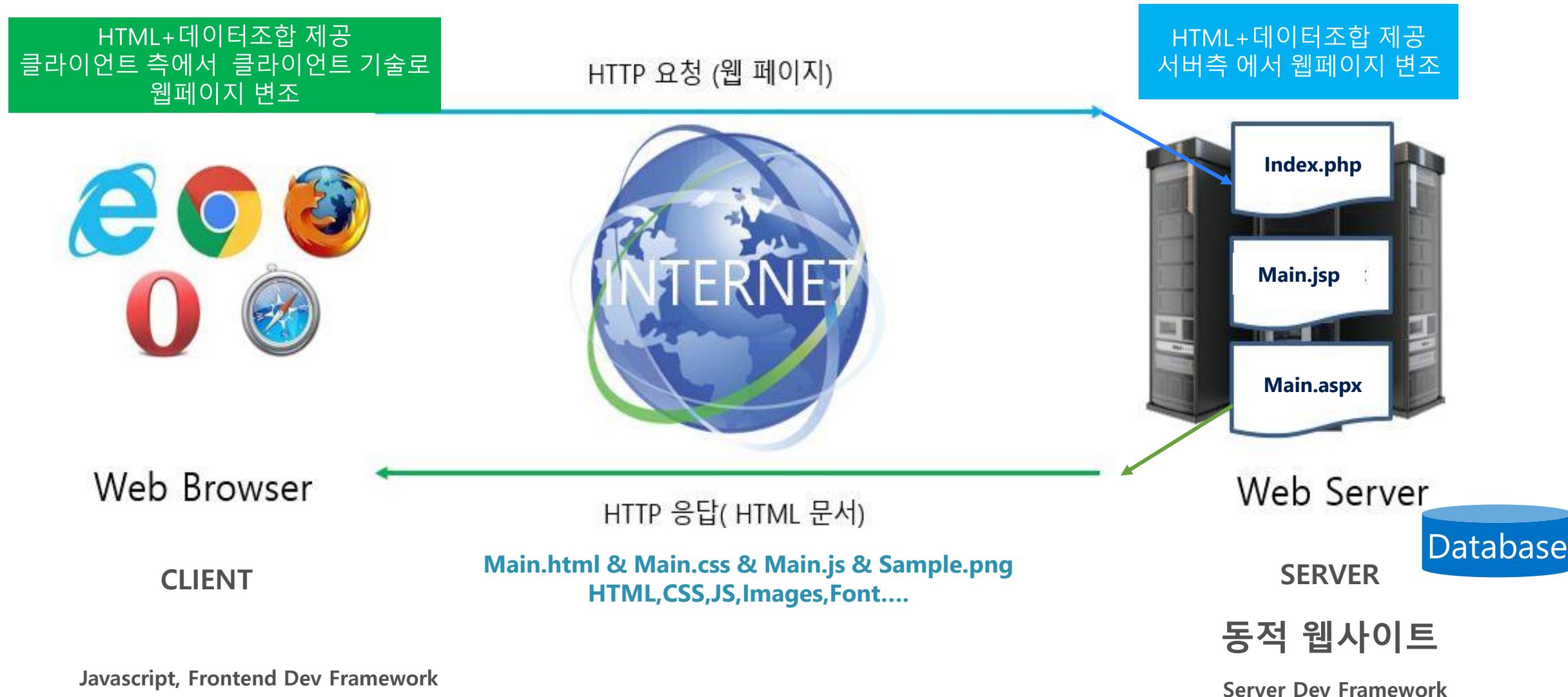
- 동적 웹사이트/웹페이지

- 동적 웹사이트와 동적 웹페이지란? 초기 개발된 웹페이지 내용이 프로그래밍에 의해 변경되는 페이지, 웹사이트
- 동적 웹 프로그래밍(HTML변조) 기술들
 - ↳ Client : HTML,CSS,Javascript,Javascript Libraries(Jquery), CSS Libraries(Plugins),Frontend Dev Frameworks
 - ↳ Server : PHP, Python, ASP.NET(C#), JSP(JAVA), Javascript(Node.js)

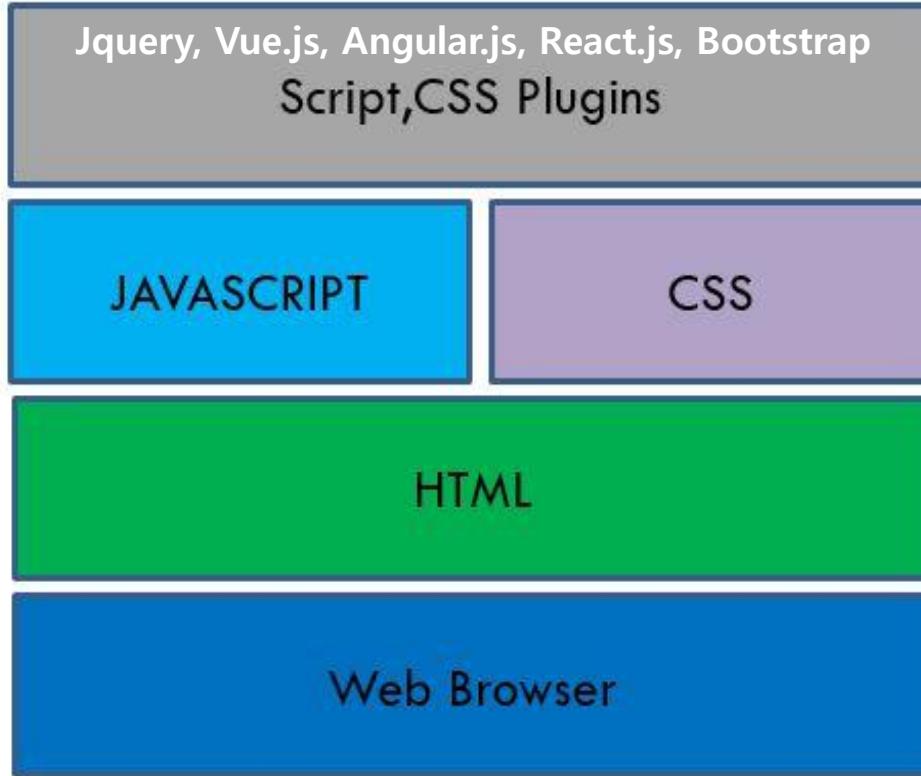
정적 웹사이트(정적 웹페이지)



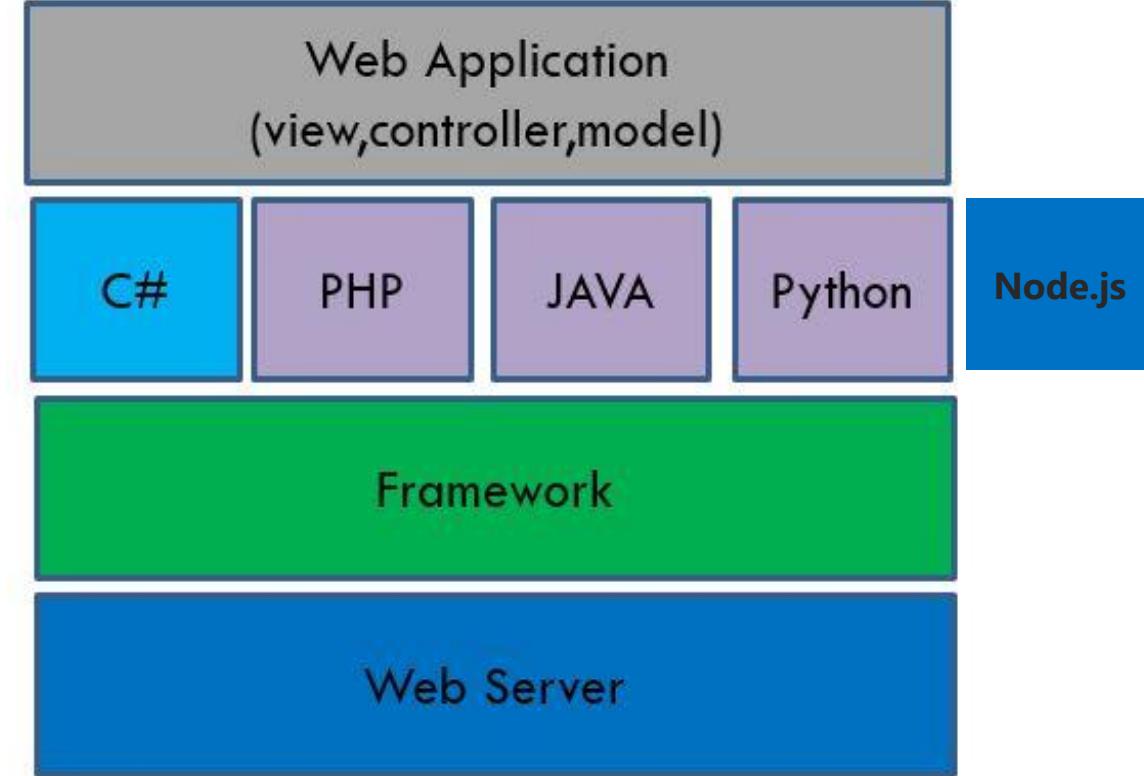
동적 웹사이트(동적 웹페이지-HTML변조여부)



Client and Server Web Technologies



Client Side Technologies



Server Side Technologies

HTML변조 작업을 어디서 웹 브라우저에서 하느냐? 웹서버에서 하느냐?
HTML변조 작업을 위해 서버측 기술을 사용할까 클라이언트 기술을 사용할까?

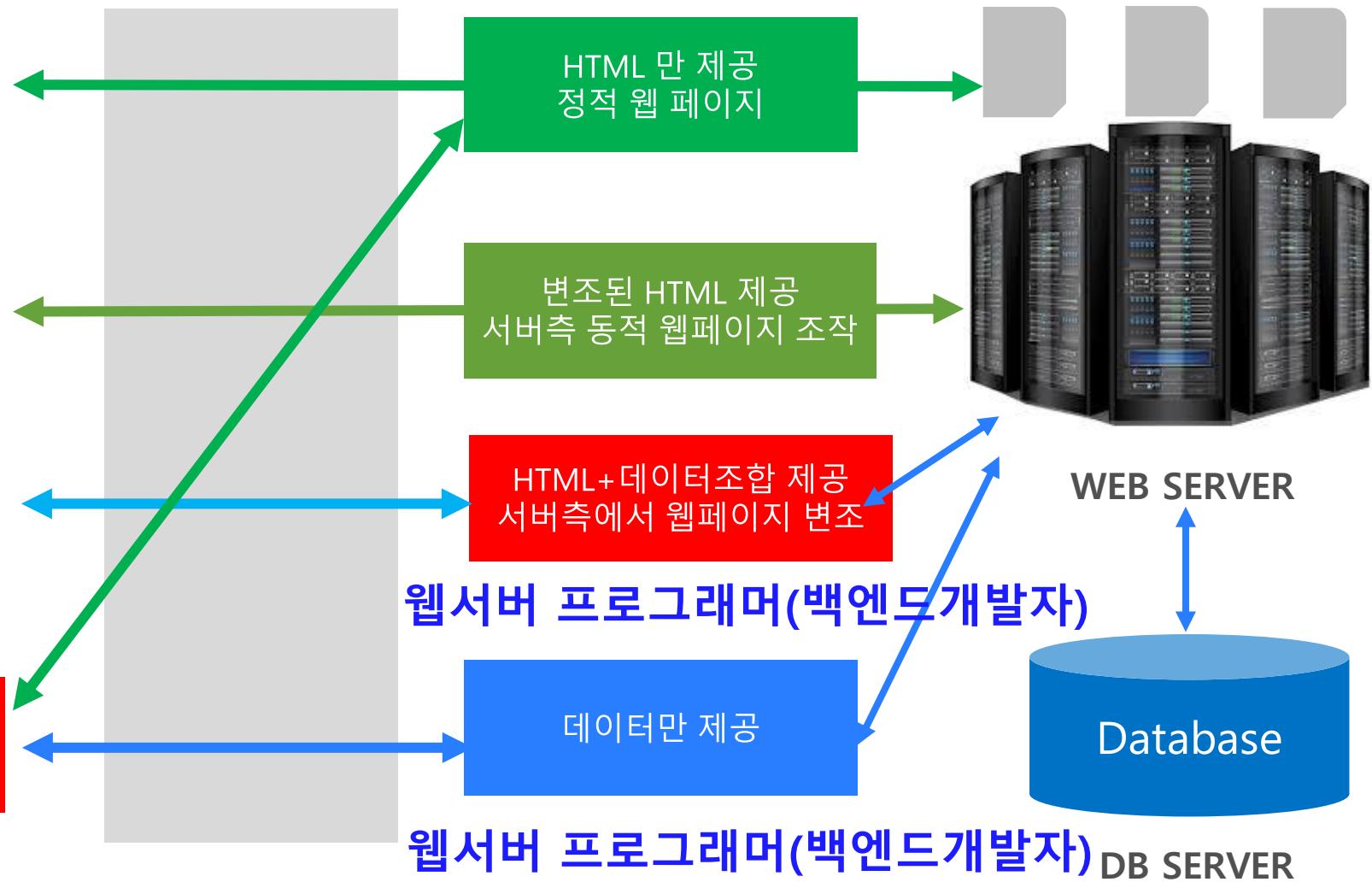
정적 웹사이트와 동적 웹사이트



Web Browser

HTML+데이터조합 제공
클라이언트측 웹페이지 변조조작

Frontend 개발자



웹 개발 직군

웹 기획자

웹디자이너

웹퍼블리셔

- HTML5,CSS3,Javascript
- 웹 표준 준수
- 웹 접근성 준수
- 반응형 웹 코딩
- 최초 코드 생성직군

웹(서버)개발자

풀 스택 개발자

프론트 엔드 개발자 : 웹 브라우저 기반 코딩

HTML5,CSS3,Javascript

Frontend Framework(Jquery,Vue.js,React,Angular.js)

백 엔드 개발자 : RESTful 서비스, 웹서버 기반 코딩

Node.js ,Python, JAVA, C# , PHP,

SQL,DB,Server,Architecture

데브옵스(SM, 배포, 유지보수, CI, CD)

DBA

인프라 관리자(서버, 네트워킹)

(시스템 엔지니어)

자바스크립트 와 자바스크립트 라이브러리

Javascript

- Script 언어

- 스크립트 언어는 매우 빠르게 배우고 작성하기 위해 고안됨.
- 짧은 소스 코드 파일이나 간단한 컴퓨터 프로그래밍 개발에 사용됨.
- Javascript, VBScript,..ShellScript..CoffeeScript,TypeScript 다양한 종류의 스크립트 언어가 존재..

- Javascript 언어

- 객체 기반 스크립트 프로그래밍 언어
- 과거 주로 웹 브라우저 기반에서 동적 웹페이지 개발기술로 사용되다.
- 최근엔 서버측 동적 웹 프로그래밍(Node.js) 및 다양한 분야에서 사용됨.
- **ECMA**스크립트(**ECMAScript**, 또는 ES)는 **Ecma** 인터내셔널의 **ECMA-262** 기술 규격에 정의된 자바 스크립트 표준스펙 을 준수한 스크립트- 최신버전은 **ECMA6**

**ECMA1,2,3,4,6(2015년) ES6(객체지향언어로 향상=ES2015..2016..2017..),,,
년도식표기법으로 변경 ES7=X**

ECMAScript 6

A bright new future is coming

자바스크립트 라이브러리 & 플러그인

- **Javascript Library**

- 자바스크립트 언어로 개발된 재활용 가능한 기능의 집합체(프로그램덩어리)
- 각종 개발분야에서 자주 사용되는 기능을 미리 구현 쉽게 재활용 가능하게 해
- 개발의 생산성과 효율성을 제공한다.
- 코드파일> 라이브러리>프레임워크> 플랫폼
- Jquery.js, Bootstrap.js, Vue.js Angular.js, React.js, Node.js

- **Plugin**

- 외부에서 제공하는 각종 오픈소스 라이브러리를 쉽게 본인의 개발 소스에
- 쉽고 빠르게 추가하여 개발할 수 있는 각종 라이브러리 파일들
- 플러그인 라이브러리를 로컬로 가져와 직접 개발소스에 포함시켜 참조하거나 외부저장소에 있는 플러그인을 인터넷을 통해 참조(CDN)하는 방식 두가지 제공

자바스크립트로 할수 있는 일들

프로그래밍 기초

- **메모리 와 CPU(프로세서)**

- CPU(프로세서): 연산과 제어 담당 1+1 : 연산담당
- 메모리: 기억과 저장을 담당 = 2를 저장

- **프로그래밍 언어 와 기계어**

- **프로그래밍 언어**
 - ㄴ 개발자가 컴퓨터를 제어하기 위해 사용하는 언어(Javascript,C#,JAVA,Python,PHP)
- **기계어**
 - ㄴ 컴퓨터 기계가 이해하는 언어(0,1,0,1....),
- **컴파일이란?** 프로그래밍언어를 기계어로 변환하는 일, 컴파일러?

- **인터프리팅 언어 와 컴파일링 언어**

- **인터프리팅 언어?**

사용자 호출 시점에서 위에서부터 아래로 순차적으로 해석해가며 컴파일해 기계어로 변환 다소 실행 속도가 느리다.
(Javascript,python,php,asp,jsp,html,css)

- **컴파일일링 언어?**

프로그래밍 개발소스를 미리 컴파일 해 런타임 시 별도 기계어 변환과정없이 바로 실행해 실행속도가 인터프리터 언어보다 빠르다.
전체 소스코드를 보고 명령어를 수집하고 재구성, 속도가 빠르다.(C, C++,C#,JAVA)

프로그래밍 기초

- 프로그래밍 메모리 구조 (영역)

- **Code 저장 메모리 영역**

- └ 프로그램 코드 저장 영역
- └ 컴파일러가 메모리 영역 결정

- **Data 저장 메모리 영역**

- └ 전역변수, 정적변수, 배열, 구조체 등 저장
- └ 프로그램 시작시 생성, 종료시 반환
- └ 컴파일러가 메모리 영역 결정

- **Heap 영역 : 동적 메모리 영역 할당,**

- └ 메모리 주소값에 의해 참조되고 사용됨
- └ 클래스, 오브젝트를 저장, new 함수를 통해 생성
- └ 개발자가 정의 런타임시 결정

- **Stack 영역 : 프로그램 컴파일러가 자동 할당하는 임시 영역**

- └ 지역변수, 매개변수, 리턴값 등 잠시 사용되다 사라지는 데이터를 저장하는 영역
- └ 함수 호출시 생성되고 함수가 끝나면 시스템에 반환
- └ 컴파일 타임에 크기 결정

낮은 주소
(low memory)

높은 주소
(high memory)

런 타임에
크기가 결정됨

컴파일 타임에
크기가 결정됨

메모리

실행할
프로그램의
코드

전역 변수
정적 변수

사용자의
동적 할당

지역 변수
매개변수

메모리

실행할
프로그램의
코드

전역 변수
정적 변수

사용자의
동적 할당

지역 변수
매개변수

코드 영역

데이터 영역

힙 영역



스택 영역

브라우저 구조와 렌더링 프로세스

- **렌더링 엔진 역할**

- HTML문서 파싱
- DOM 구조 로드
- CSS 파싱, RenderTree 생성
- UI Backend Layer기반 노드표시

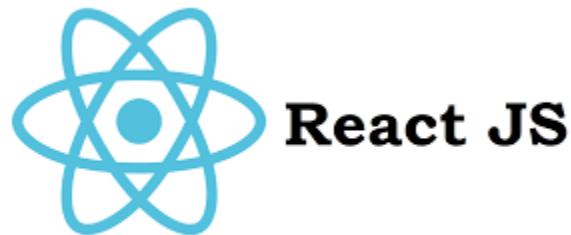
- **렌더링 엔진 유형**

- IE - Trident
- 크롬- Webkit
- 파이어폭스- Gecko
- 사파리 – Webkit(애플제작)
- 오페라- Presto



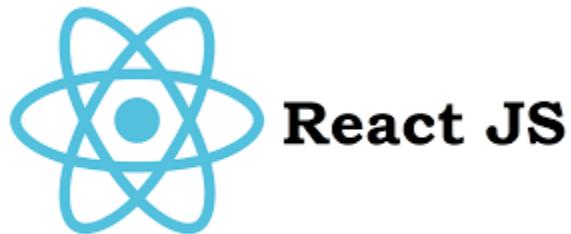
자바스크립트로 할 수 있는 일들

- 웹 프론트 엔드 개발



자바스크립트로 할 수 있는 일들

- 웹 백 엔드 개발



5 Node.Js Framework

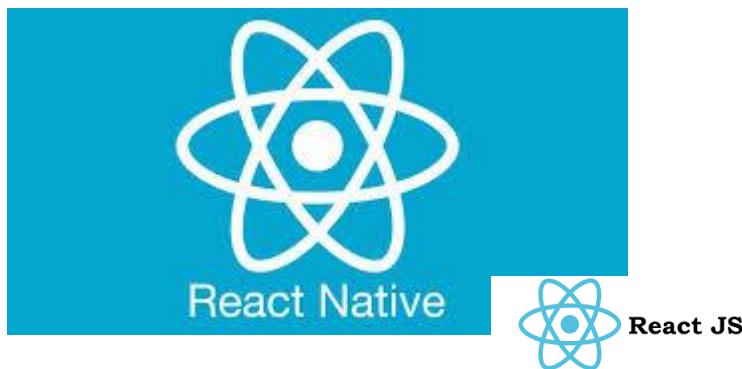
MOST POPULAR Node.Js Framework in 2018

The slide displays the logos of five popular Node.js frameworks: Express.js, Hapi.js, Mojito, Meteor, and Socket.io. Each logo is contained within a black hexagonal frame. Below each logo, the name of the framework is written in white text. The background of the slide is light green.

Framework	Logo Description	Name
Express.js	Black hexagon with a white outline; the word "express" is written vertically on the left side.	Express.js
Hapi.js	Black hexagon with a white outline; features a small orange bee icon above the word "hapi".	Hapi.js
Mojito	Black hexagon with a white outline; the word "mojito" is written vertically on the left side.	Mojito
Meteor	Black hexagon with a white outline; the word "METEOR" is written vertically on the left side.	Meteor
Socket.io	Black hexagon with a white outline; features a white lightning bolt icon.	Socket.io

자바스크립트로 할 수 있는 일들

- 모바일 하이브리드 앱 개발



네이티브 앱

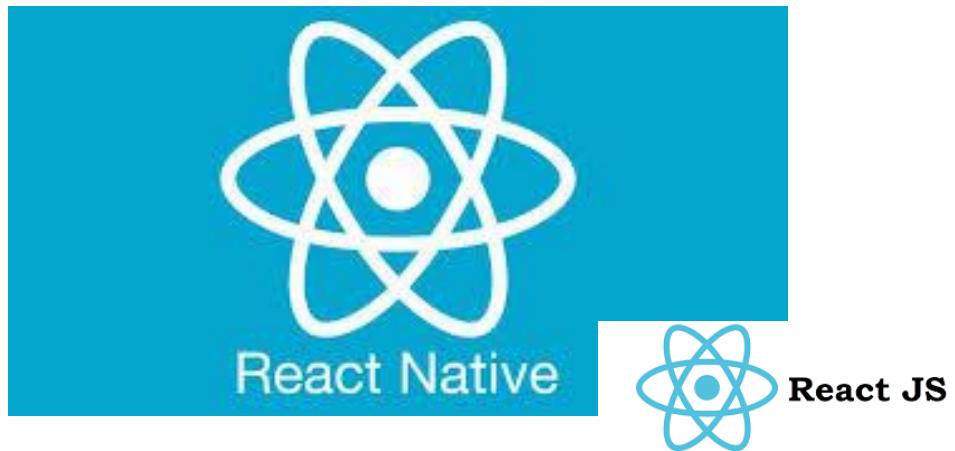
모바일 앱

웹 앱

하이브리드 앱

자바스크립트로 할 수 있는 일들

- 모바일 네이티브 앱 개발



네이티브 앱

모바일 앱

웹 앱

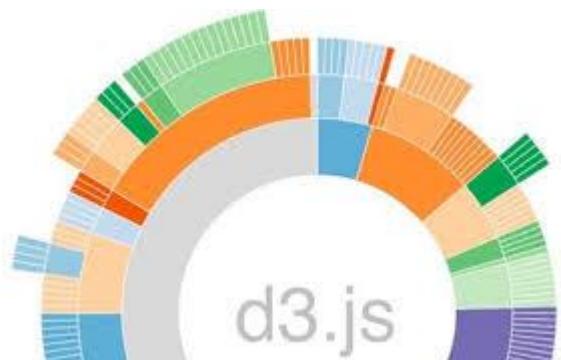
하이브리드 앱

자바스크립트로 할 수 있는 일들

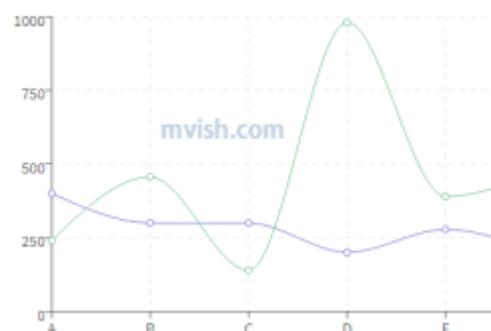
- AI(머신러닝, 딥러닝) 개발



- Data Visualization 개발

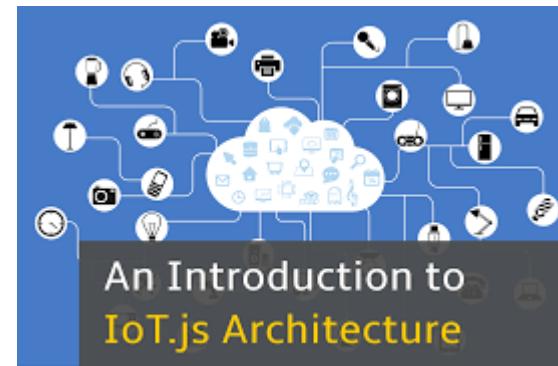
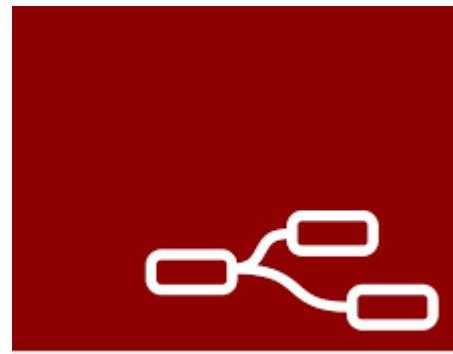


Recharts



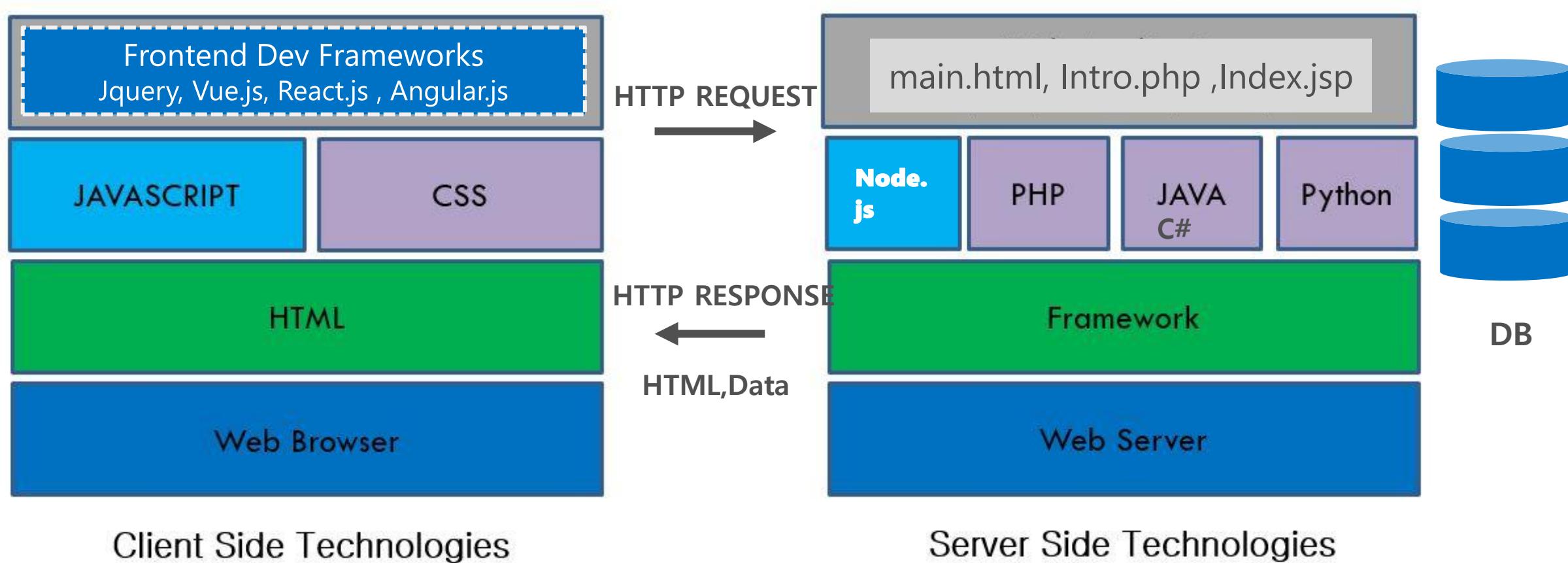
자바스크립트로 할 수 있는 일들

- IoT(사물인터넷 제어)



프론트엔드 VS 백엔드 웹 프로그래밍

프론트엔드 와 백엔드 웹 프로그래밍



Client Side Technologies

Server Side Technologies

웹 프로그래밍의 주업무는 **HTML** 과 **데이터**를 다루는 일
데이터를 제공하는 것은 서버측에서만 가능하지만

HTML과 데이터는 다루는 것은 클라이언트와 서버 양측에서 모두 가능하다

클라이언트에서 HTML과 데이터를 다루고 런타임 환경이 브라우저이면 프론트 엔드 웹 개발자
서버에서 HTML과 데이터를 조작하고 서버가 런타임 환경이면 백엔드 및 웹 서버 프로그래머(웹 개발자)

AJAX Vs Restful Open API : 데이터 처리기술

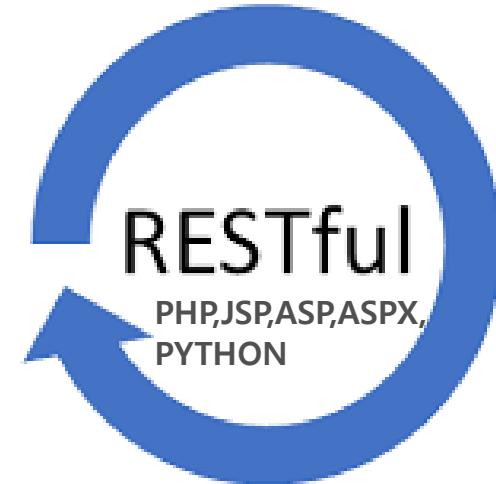
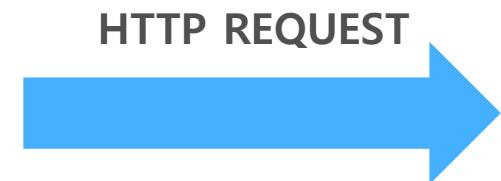
- AJAX : 에이잭스 -브라우저 기술
 - AJAX(Asynchronous JavaScript and XML)?
 - HTML Element & Data 바인딩
 - 화면 깜빡거림없이 자바스크립트 기반 순수 데이터 호출기술
- RESTful Service (Open API) :서버기술,오픈API
 - REST(Representational State Transfer)?
 - Restful Service? 서버측 데이터 처리기술 기반 서비스
 - Open API-시스템간 통합(SI) 기술로 각광-데이터교환
 - HTML 이 아닌 데이터만 처리한다(JSON & XML)



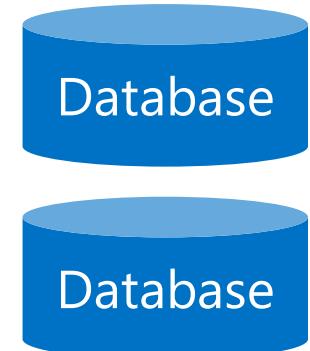
AJAX Vs Restful Open API



WEB Browser

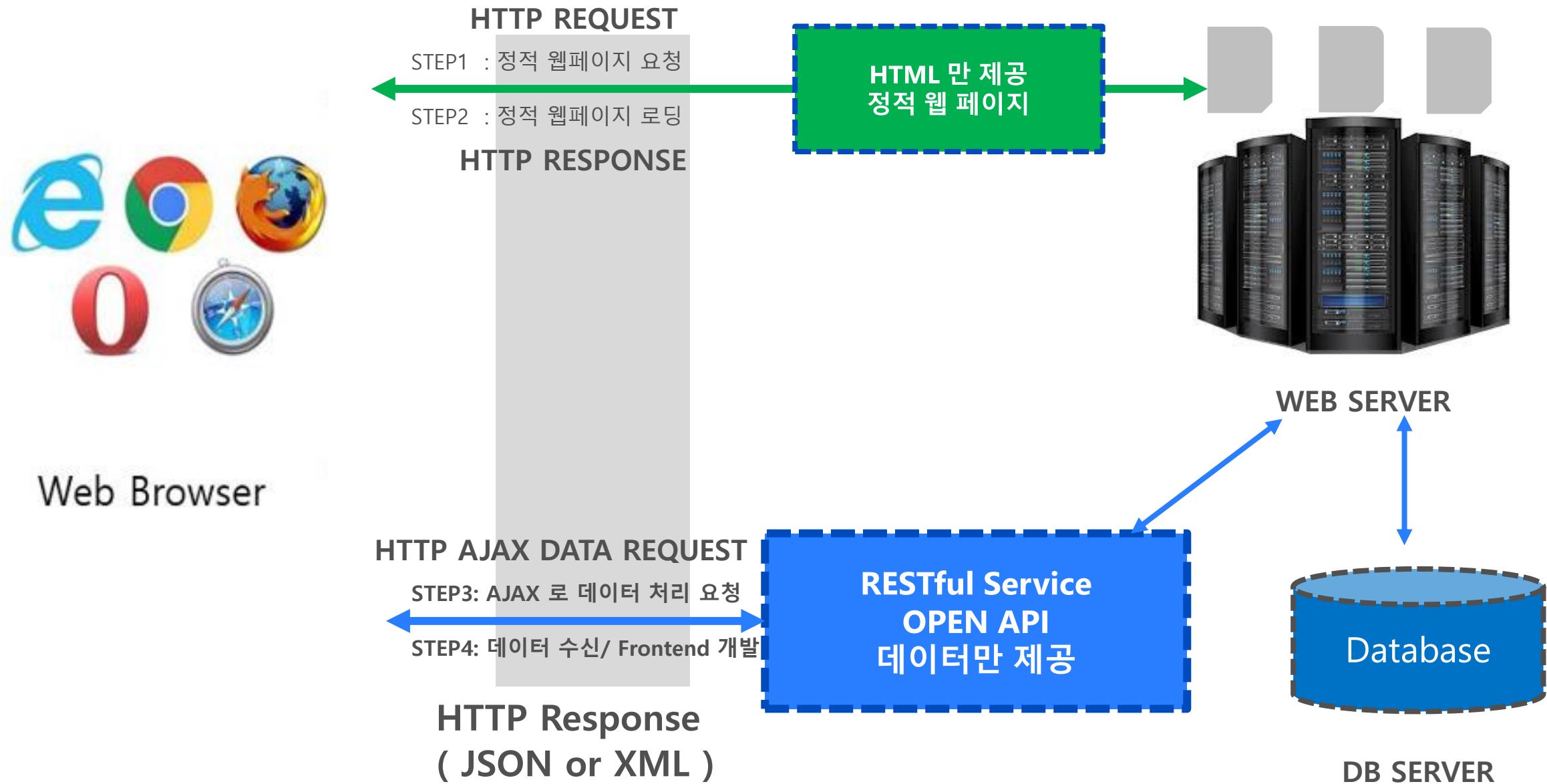


- GET
- POST
- PUT
- DELETE

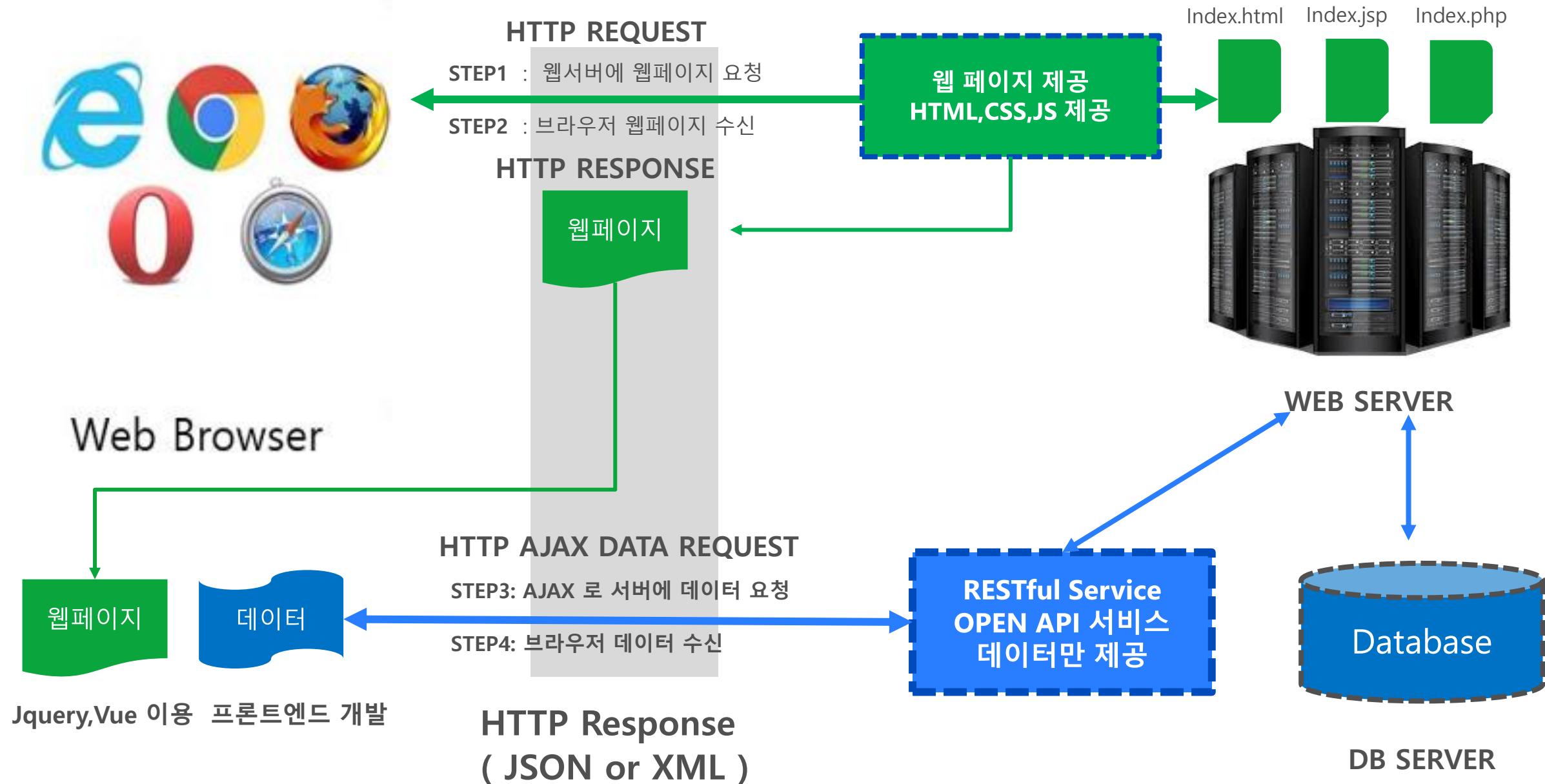


WEB SERVER

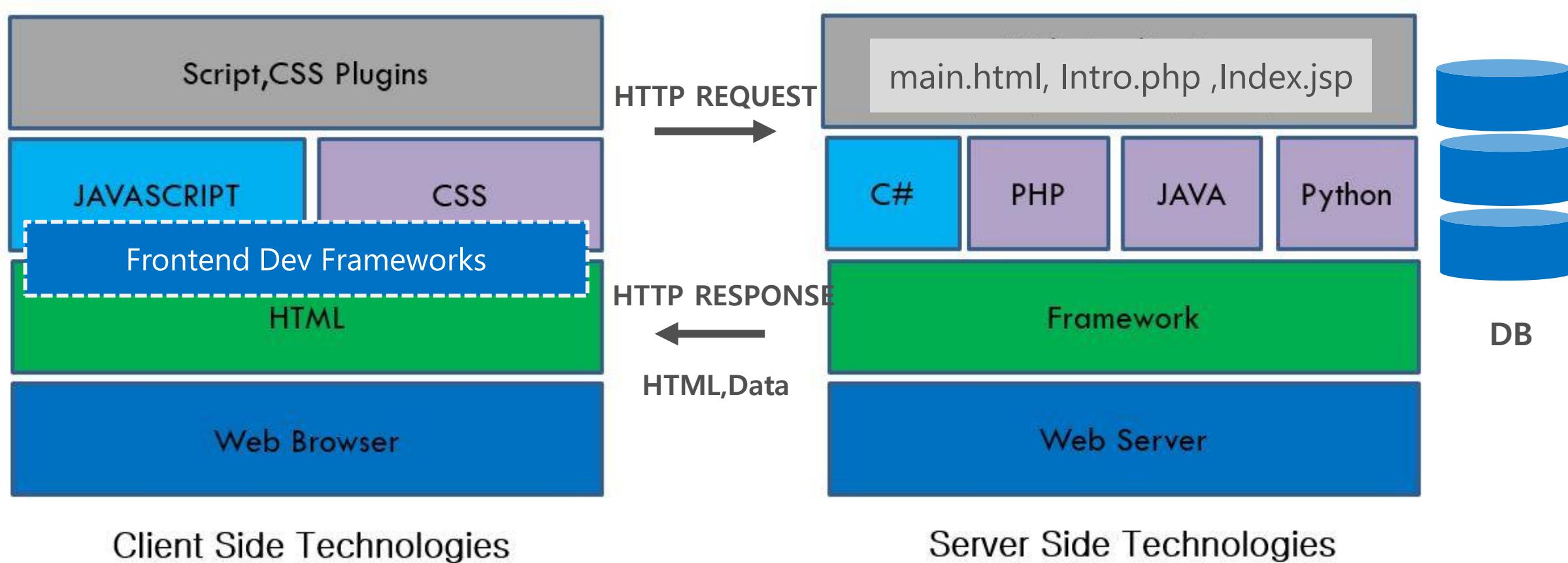
AJAX / RESTful 서비스 호출 프로세스



FRONTEND -통신 프로세스



프론트엔드 와 백엔드 웹 프로그래밍



웹 프로그래밍의 주업무는 **HTML** 과 **데이터**를 다루는 일
데이터를 제공하는 것은 서버측에서만 가능하지만

HTML과 데이터는 다루는 것은 클라이언트와 서버 양측에서 모두 가능하다
클라이언트에서 HTML과 데이터를 다루면 프론트엔드 웹 개발자
서버에서 다루면 백엔드 및 웹 서버 프로그래머(웹 개발자)

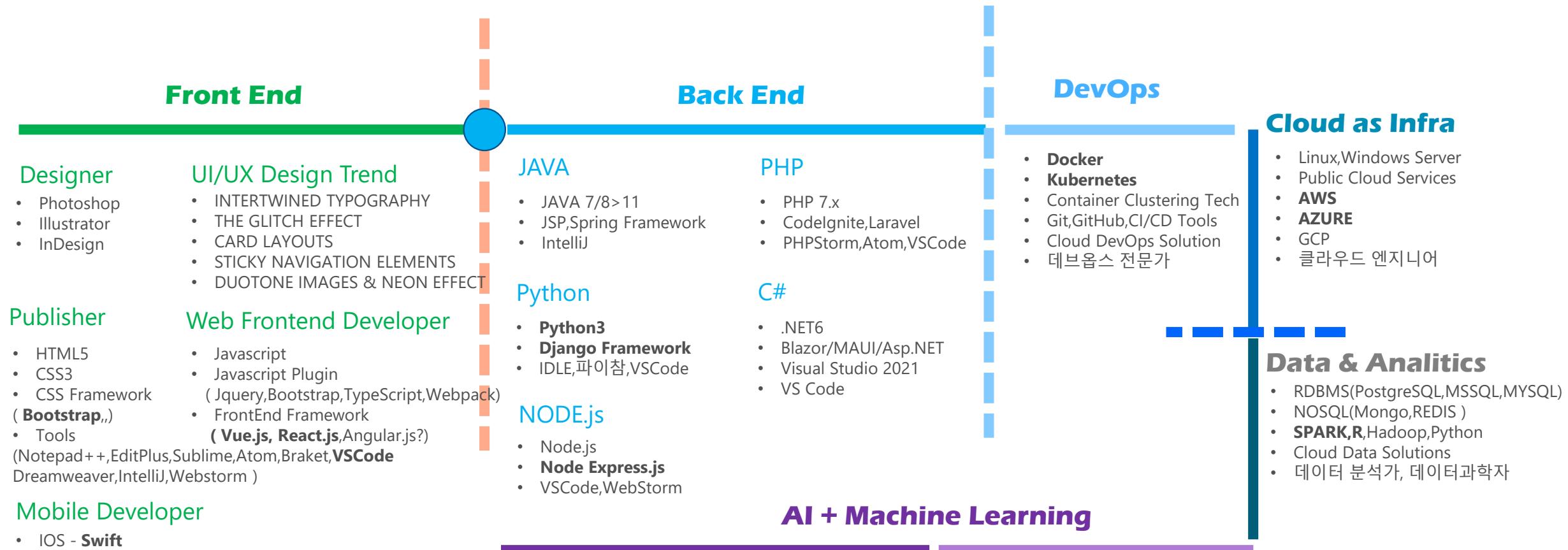
Frontend vs Backend Web Programming

- **Frontend 개발자**

- 웹 클라이언트 기반에서(브라우저) HTML문서 내용을 변조하는 일을 한다.
- 웹 클라이언트 기반 기술로 HTML 문서의 내용을 제어한다.
- HTML,CSS,Javascript,Javascript Library, Frontend 개발 Framework 을 활용해 개발한다.
- 주 개발언어는 Javascript(Jquery, Frontend Framework) 이다.

- **Backend 개발자**

- 웹 서버 기반에서 HTML문서를 변조하는 일을 한다.
- 클라이언트로부터 제공된 데이터를 관리하고 클라이언트에게 데이터를 제공한다.
- 데이터의 효율적 처리(CRUD) 업무가 주 업무이다.
- 각종 서버(웹서버,DB서버,메일서버 등등) SW에 대한 이해가 필요하며
- 서버 O/S에 대한 이해가 필수다
- 웹서버 프로그래밍 언어 **Python,C#(ASP.NET),JAVA(JSP),PHP,ASP,Node.js** 등 다양하다.



AI + Machine Learning

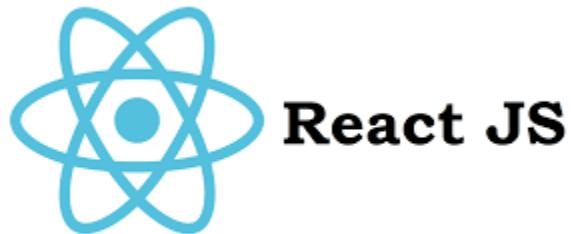
- **Tensorflow, Keras, Pytorch, Python, R**
- 자연어처리, 딥러닝, 강화학습, 오브젝트디텍션
- 추천시스템, 회귀분석, 분류
- Cloud Automated ML Tools
- AI Chatbot(Robotic Process Automation)
- 인공지능 엔지니어, 인공지능 활용전문가

MSoftware

Node.js

자바스크립트로 할 수 있는 일들

- 웹 백 엔드 개발



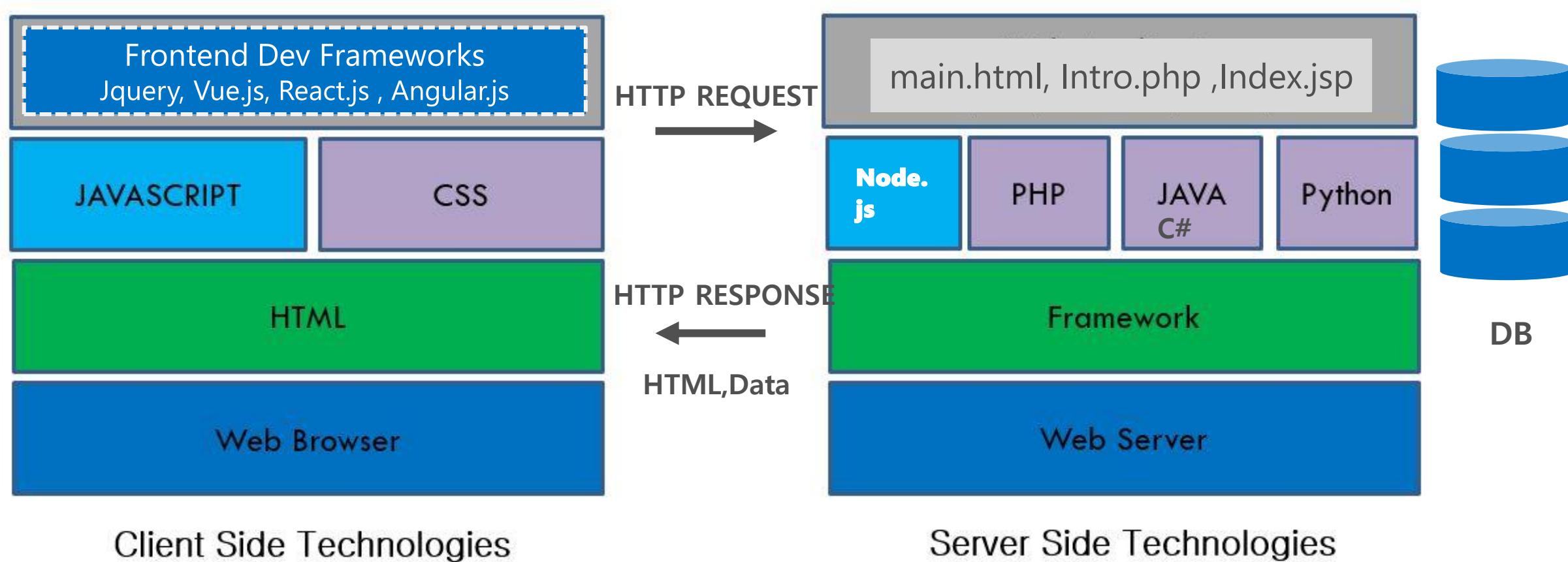
5 Node.Js Framework

MOST POPULAR Node.Js Framework in 2018

The slide displays the logos of five popular Node.js frameworks: Express.js, Hapi.js, Mojito, Meteor, and Socket.io. Each logo is contained within a black hexagonal frame. Below each logo, the name of the framework is written in white text. The background of the slide is light green.

Framework	Logo Description	Name
Express.js	Black hexagon with a white outline, containing the word "express" in white lowercase letters.	Express.js
Hapi.js	Black hexagon with a white outline, containing the word "hapi" in white lowercase letters with a small orange bird icon above the "a".	Hapi.js
Mojito	Black hexagon with a white outline, containing the word "mojito" in white lowercase letters.	Mojito
Meteor	Black hexagon with a white outline, containing the word "METEOR" in white uppercase letters with a small green lightning bolt icon above the "E".	Meteor
Socket.io	Black hexagon with a white outline, containing a white lightning bolt icon.	Socket.io

프론트엔드 와 백엔드 웹 프로그래밍



Client Side Technologies

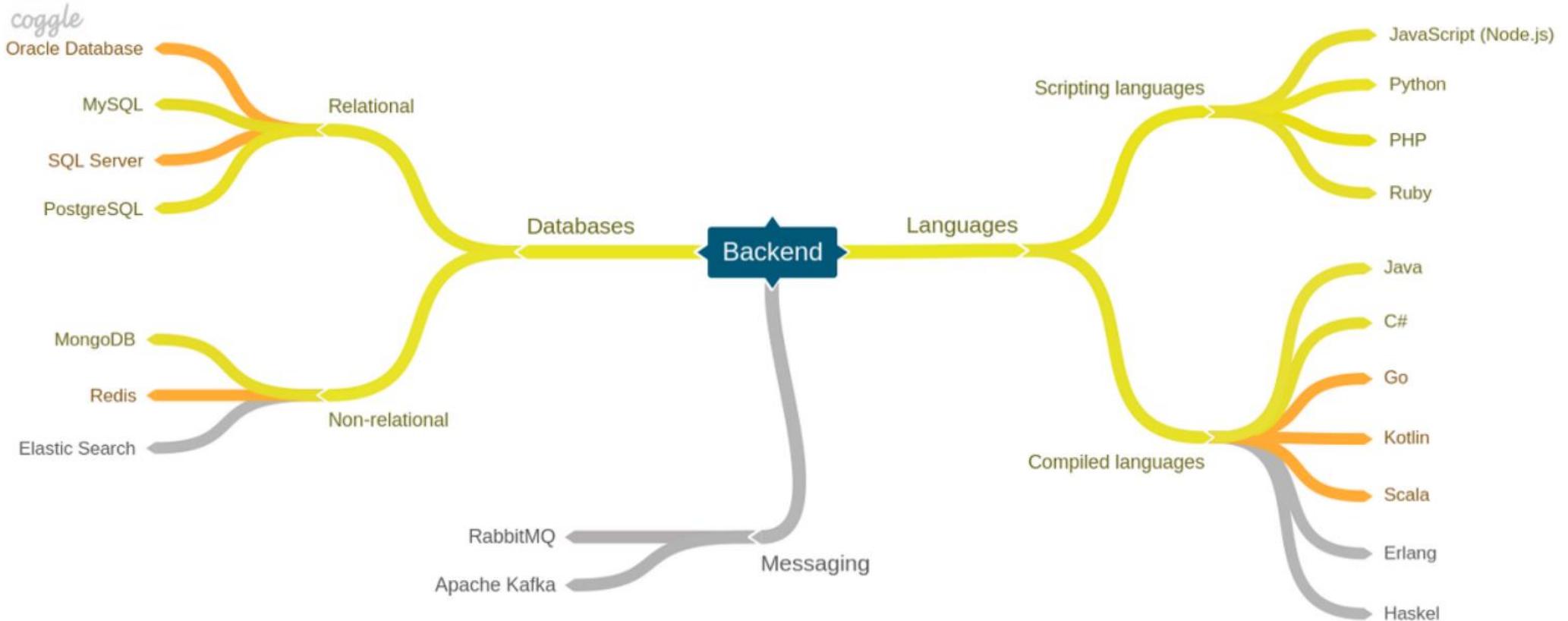
Server Side Technologies

웹 프로그래밍의 주업무는 **HTML** 과 **데이터**를 다루는 일
데이터를 제공하는 것은 서버측에서만 가능하지만

HTML과 데이터는 다루는 것은 클라이언트와 서버 양측에서 모두 가능하다

클라이언트에서 HTML과 데이터를 다루고 런타임 환경이 브라우저이면 프론트 엔드 웹 개발자
서버에서 HTML과 데이터를 조작하고 서버가 런타임 환경이면 백엔드 및 웹 서버 프로그래머(웹 개발자)

Backend Tech Tree



Node.js

- **Node.js**
 - 노드는 크롬 v8 자바스크립트 엔진으로 빌드된 자바스크립트 런타임
 - **런타임이란? 특정 개발언어로 만든 프로그램을 실행할수 있는 환경**
 - 노드는 자바스크립트로 개발된 프로그램을 각종 컴퓨터(서버포함)에서 실행할수 있게 런타임 환경지원
 - 2008년 구글이 v8 자바스크립트 엔진을 이용 크롬 브라우저를 출시, 오픈소스화
 - 2009년 v8엔진 기반 노드 프로젝트 시작
- **노드 주요 특징**
 - 이벤트 기반 개발 모델
 - 논 블록킹 I/O 모델 제공
 - 가볍고 효율적인 백엔드 스크립트 개발 프레임워크
 - **NPM**이라는 자바스크립트 오픈소스 라이브러리 저장소 생태계 제공

<https://nodejs.org/en/>

<https://nodejs.org/ko/>

Node = js 모듈 실행 환경

NVM = Node Version Manager

NPM = Node Package Manager

NVM(Node Version Manager)

- Node Framework 버전 설치/삭제/버전 관리 기능제공
 - nvm 설치 : <https://github.com/coreybutler/nvm-windows/releases>
 - [nvm-setup.exe](#) or [nvm-setup.zip](#) 다운로드 압축해제 설치
 - 동일한 컴퓨터에 여러 버전의 Node Framework을 설치하고 관리코자 할때 사용
 - NVM 주요 명령어
 - NVM버전 확인 : **nvm version**
 - 현재 NODE FRAMEWORK 버전 목록 : **nvm ls**
 - * 16.13.1 (Currently using 64-bit executable) : 현재 사용버전 *로 표시
 - 14.18.3
 - 특정 노드 프레임워크 버전 활성화 하기: **nvm use 버전명**
 - 특정 노드프레임워크 버전 설치하기 : **nvm install 버전명**
nvm install 14.18.3
 - 특정 노드프레임워크 삭제하기 : **nvm uninstall 버전명**
- ** nvm 명령어를 인식못하면 개발툴을 재시동하거나 터미널을 닫고 터미널을 다시 오픈한다.
- ** cmd나 vscode에서 nvm use시 에러나는경우는 권한문제로 관리자권한으로 cmd나 vscode를 실행한다.

Npm 설치 및 패키지 관리

- npm
 - Node Package : 특정 기능을 제공하는 노드 모듈들의 집합체
 - Node Package 저장소 : 세계 최대 재사용 가능한 노드(패키지들)와 자바스크립트 오픈소스 저장소
 - Npm(Node Package Manager): 각종 Node Package의 설치 및 설치된 팩키지들을 관리해주는 SW
 - 개발자는 노드 팩키지 저장소에서 NPM명령어를 이용해 각종 팩키지를 개발 또는 서비스 환경으로 다운로드 설치하여 빠르게 오픈소스 팩키지 기반 프로젝트 개발 및 서비스가 가능하다.
- Package.json
 - 해당 팩키지의 정보와 버전 및 의존 팩키지 정보를 관리하는 파일
 - 노드 프로젝트는 하나의 팩키지 파일이 반드시 필요하다.
 - 프로젝트 시작 전 packange.json 파일부터 생성 시작
 - **npm init** 실행 후 순차적 정보 입력 후 최종 yes 하면 폴더 또는 프로젝트에 package.json 파일 생성
- 팩키지 설치 : **npm install 팩키지명** 또는 **npm i 팩키지명** 또는 여러 팩키지를 동시설치시 **npm install [팩키지명] [팩키지명] [팩키지명]**
 - └ dependencies : 설치된 팩키지 정보 제공 - package.json
 - └ NODE-MODULES 폴더 : 설치된 노드 모듈 물리적 파일 위치 – 의존 관계파일도 함께 설치됨.
 - └ NODE-MODULES 폴더 삭제된 경우 packange.json파일만 있으면 **npm install** 또는 **npm i** 명령어 치면 자동 재설치됨.
 - └ package-lock.json : 설치된 팩키지들간의 의존관계 정보를 관리해줌
- 개발용 팩키지 설치: **npm install --save -dev [팩키지명]**
 - └ 실제 프로덕션 배포시에는 사용하지 않고 개발시에만 사용하는 팩키지일때
 - └ devDependencies : 설치된 개발용 팩키지 정보 제공 - package.json
- 전역으로 팩키지 설치: **npm install -global [팩키지명]** or **npm i -g 팩키지명**
 - └ 현재 개발 폴더에 팩키지를 설치하지 않고 npm이 설치된 공간에 팩키지를 설치하여 모든 프로젝트의 콘솔에서 실행가능하게 함
 - └ **c:\users\사용자계정\appdata\roamng\npm** 경로내 설치
 - └ 전역으로 설치한 팩키지는 package.json파일에 기록되지 않음

팩키지 버전

- **Npm 팩키지 버전**

- 팩키지간 의존관계로 버전관리가 중요
- 세자리로 버전 관리 : SemVer 버전관리 방식
- Major버전, Minor버전, 패치버전
- Major버전 1이상인 경우 정식버전 하위버전과 호환안됨, minor버전은 하위 호환되는 변경사항, 패치 버그수정
- ^ 버전 표시 : minor 버전까지만 설치 또는 업데이트 `npm install @^1.1.1` $1.1.1 \leq \text{버전} < 2.0.0$
- ~ 버전 표시 : 패치 버전까지만 설치
- @latest or @x : 항상 최신 버전만 설치

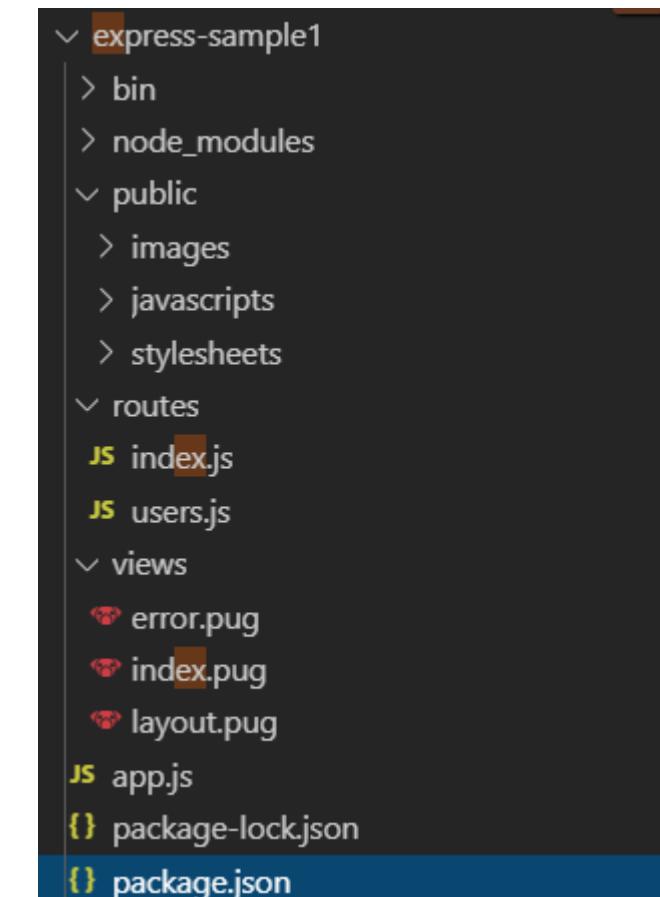
- **Npm 주요 기타 명령어**

- 업데이트 팩키지여부 조회 : `npm outdated`
- 팩키지 업데이트 : `npm update 팩키지명`
- 설치된 팩키지 삭제 : `npm rm 팩키지명` 또는 `npm uninstall 팩키지명`
- 팩키지 조회: `npm search` 검색어
- 팩키지 세부정보 확인 : `npm info 팩키지명`

Node Express로 웹서버 구현하기

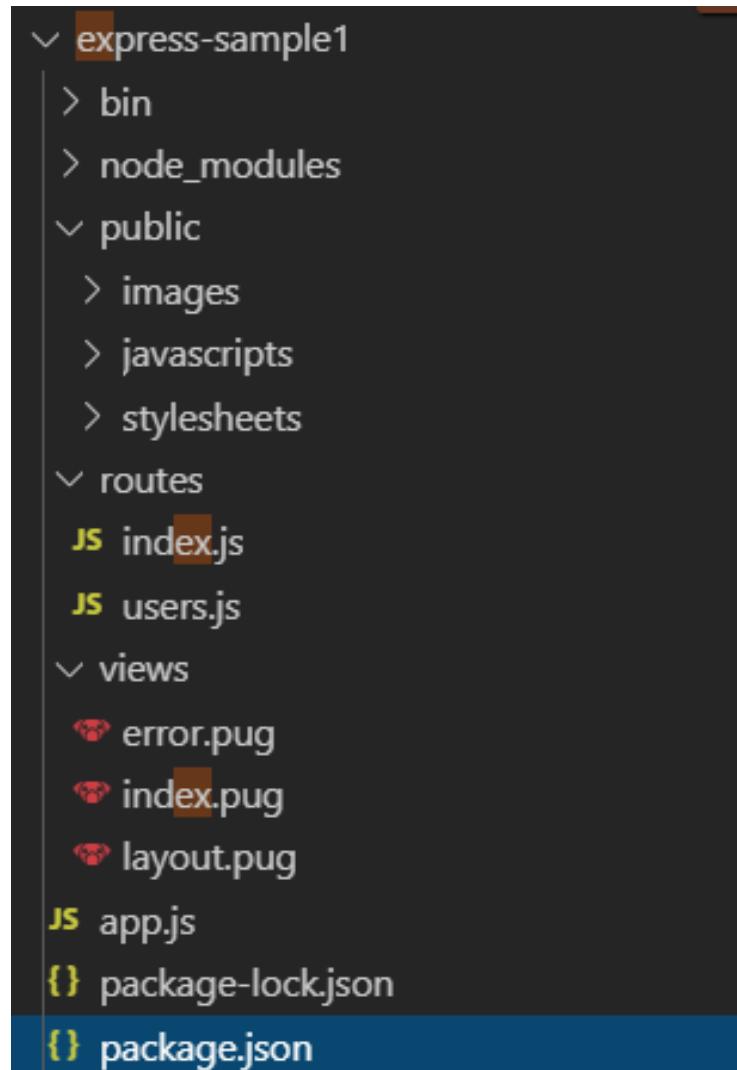
Node Express 프로젝트 구성하기

- Node Express.js
 - Node기반의 오픈소스 웹서버 프레임워크 라이브러리
 - http 전용 요청과 응답 객체 제공
 - Express와 다수 추가 팩키지 함께 설치해야함.
- Express-generator CLI 기반 웹 프로젝트 생성 및 실행
 - Express-generator 를 npm으로 전역으로 설치 : **npm i -g express-generator**
 - Express-generator 설치 후 CLI 를 통해 Node Express 기반 노드 웹프로젝트 생성 및 관리 가능
 - Node Express 웹 프로젝트 생성 : **express myfirstnodeapp --view=ejs**
 - └ MVC 패턴의 웹 프로젝트 생성
 - └ 뷰(화면) 엔진 기술로 Pug 또는 ejs 템플릿을 이용함.
 - 생성된 프로젝트 경로로 이동후 사용하는 node 팩키지 일괄 복원설치 : **npm i**
 - └ 프로젝트 폴더내 package.json 파일내 복원 설치 팩키지 확인
 - 웹프로젝트 빌드 산출물서버로 실행 : **npm start**
 - └ package.json 파일내 scripts에 start명령어 등록되어 있고 node ./bin/www.js 파일 실행되게 설정되어 있음
 - └ www.js파일내 설정된 포트를 통해 <http://localhost:3000> 웹브라우저에서 접속가능
 - 노드 프로세스 종료 : 콘솔에서 **Ctrl + C**



Node Express 웹서버 생성하기

- Node Express 프로젝트 폴더구조



- bin – www.js 노드프로젝트 시작 스크립트 파일저장
- node_modules – npm 설치 팩키지 모듈
- Public – 웹 개발 공통 요소 제공(이미지,CSS,Plugin)
- Routes –MVC 패턴에서의 Controller 역할제공-라우팅 및 사용자 요청 응답제어,DB호출,각종 비즈로직 처리
- views – 화면 템플릿 제공
- models – DB처리 기능 및 데이터 모델 정의
- app.js – 핵심적인 웹서버 역할제공 모듈
- package-json

라우팅 파일 호출 테스트 및 로그 파일 확인

<http://localhost:3000>

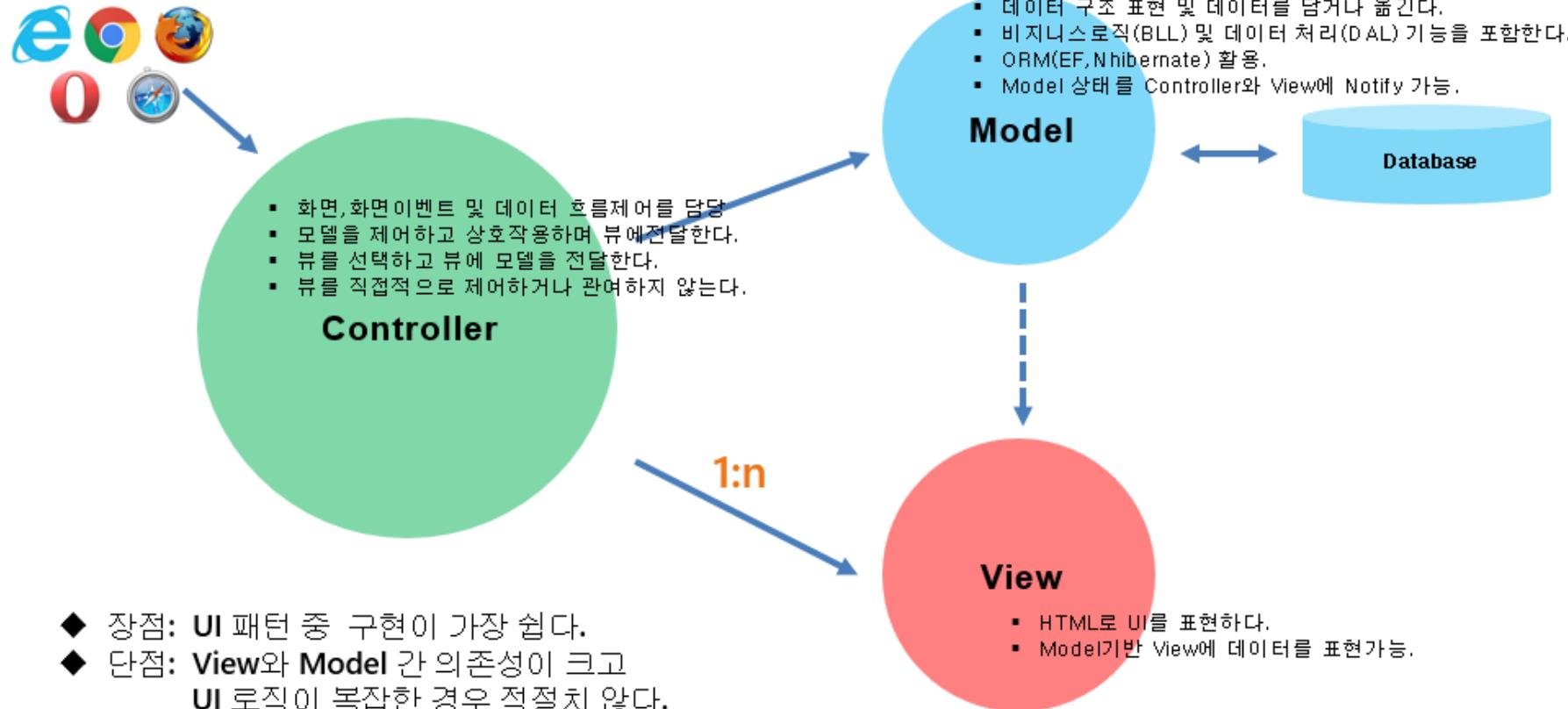
<http://localhost:3000/users>

Express 라우터 객체로
라우팅하기

UI Design Pattern

UI Design Pattern : MVC

MVC UI Design Pattern : WEB기반에서 주로 사용, ASP.NET MVC5 (Observer Pattern)
<http://test.co.kr/article/modify/1>

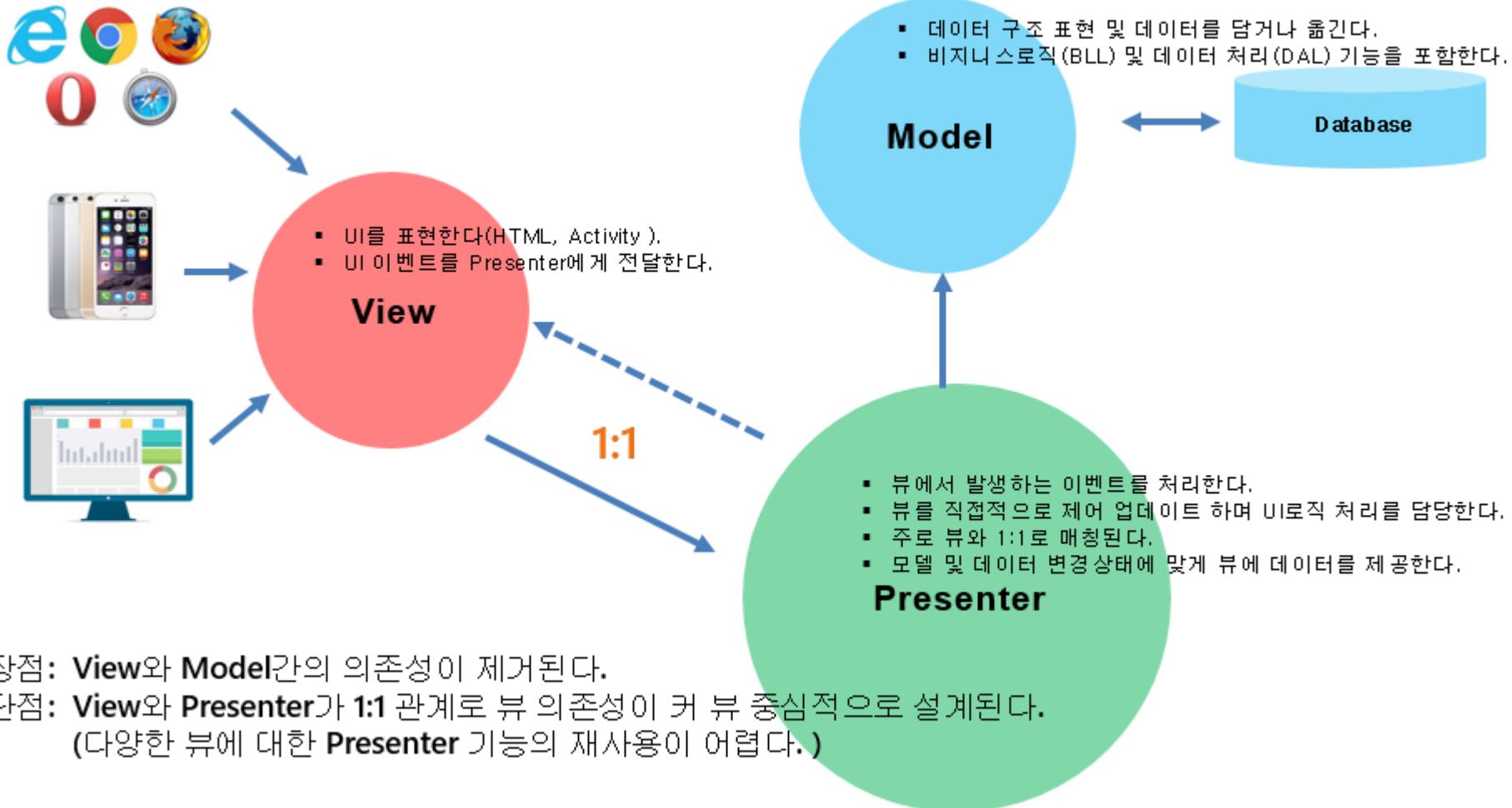


- ◆ 장점: UI 패턴 중 구현이 가장 쉽다.
- ◆ 단점: View와 Model 간의 혼성이 크고 UI 로직이 복잡한 경우 적절치 않다.

<http://mixedcode.com/Article/Index?aidx=1180>
<http://mixedcode.com/Article/Index?aidx=1184>

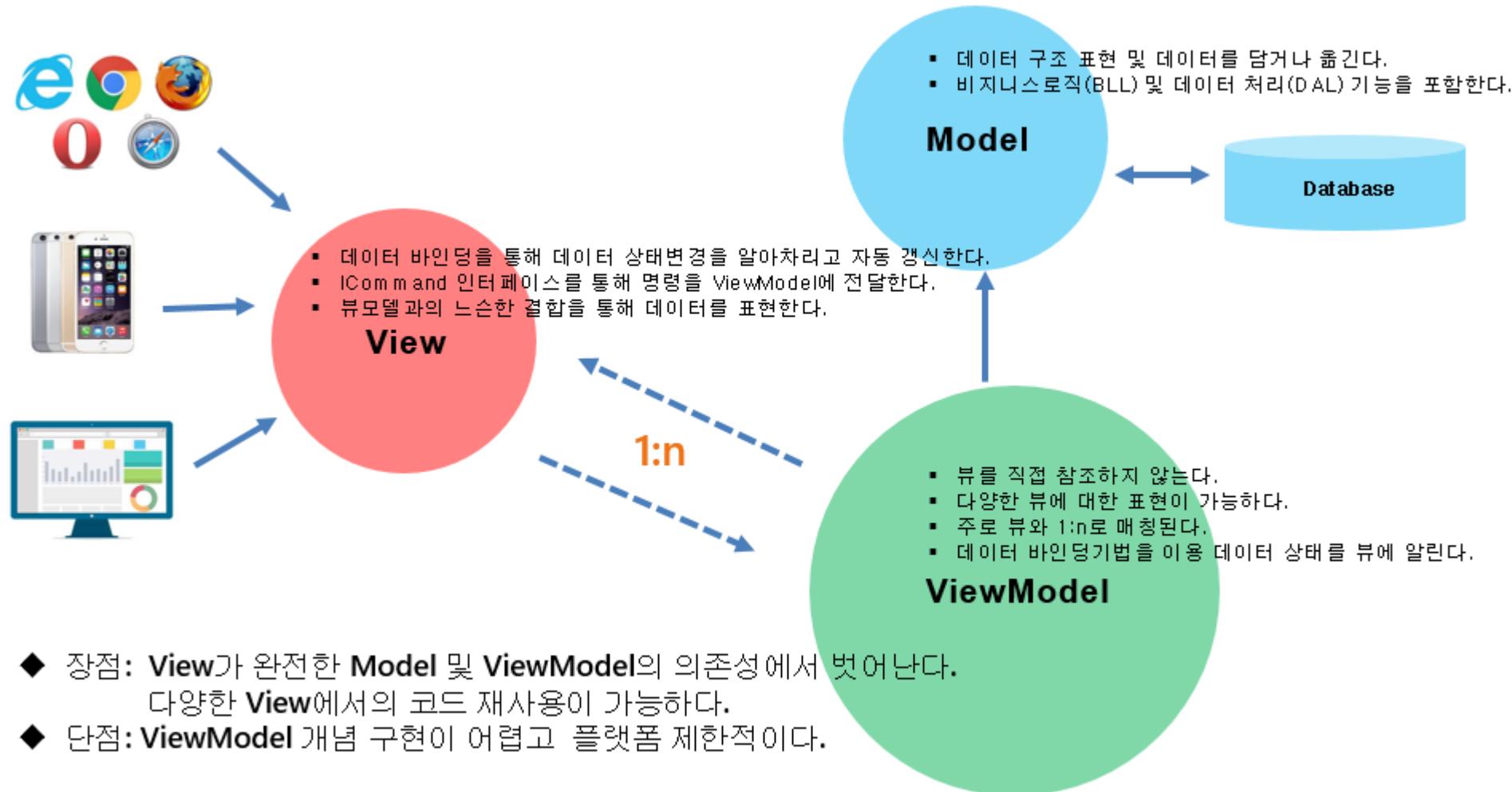
UI Design Pattern : MVP

MVP UI Design Pattern : DeskTop Application, Web, Mobile :
WinForms , ASP.NET Forms, Apps(IOS with XCODE, Andoid)



UI Design Pattern : MVVM

MVVM UI Design Pattern : Web, Windows, Apps : WPF , Xamarin.Forms , UWP, Front-END Web



Node Express – 라우터로 라우팅하기

- Routing 미들웨어 구현하기

- STEP1) Routes 폴더내 라우팅 파일(컨트롤러)을 생성한다.
- STEP2) 라우팅 파일에 라우팅 메소드를 구현한다.
- STEP3) 라우팅파일을 App.js에 라우팅 파일을 임포트하고 해당 라우팅파일의 기본접속 주소정보를 설정한다.
- 라우팅 메소드를 통해 사용자 입력주소와 맵핑되는 라우팅 메소드를 만들어 사용자의 요청 기능을 구현하고 응답하는 기능을제공함
- 라우팅 메소드 요청방식은 요청 URL주소와 메소드 유형(get/post/put/delete...)이 일치하는 경우 자동으로 해당 메소드를 검색되며 호출된다.

App.js 파일내

```
//라우터 모듈 추가  
var indexRouter = require('./routes/index');  
var usersRouter = require('./routes/users');
```

```
//라우팅 처리를 위한 라우터 미들웨어 설정하기  
app.use('/', indexRouter);  
app.use('/users', usersRouter);
```

Node Express – 라우터로 라우팅하기

- Router 요청유형-`httprequest` 객체 요청방식
 - `Router.get('요청주소',콜백함수);` //처음 웹페이지 호출시
 - `Router.post('요청주소',콜백함수);` //신규데이터 등록 처리 //http 본문을 통해 전달된 값 추출: `req.body.html요소 name`값으로 추출
 - `Router.put('요청주소',콜백함수);` //기존데이터 수정처리시
 - `Router.delete('요청주소',콜백함수);` //기존데이터 삭제 처리시
- `req.query`.쿼리스트링 키명
 - `req.params`.파라메터아이디,
 - `req.body.html요소 name`값
- 라우팅 와일드 카드
 - url 요청경로에 고유값을 포함해 전달하는 경우 아래와 같이 와일드 카드 적용가능
 - `/users/100?limit=5` 와 같이 호출하는 경우
 - 100값 추출은 `req.params.id`로 추출하고 `?limit` 쿼리스트링값은 `req.query.limit`으로 추출할수 있다.
 - 와일드카드 라우팅 방법
- Router 응답유형-`httpresponse` 주요 메소드
 - `res.render('뷰템플릿파일경로',뷰파일에 전달할 JSON 데이터);`
 - `res.json(응답할 json데이터)`
 - `res.redirect('바로이동주소경로')`
 - `res.send('버퍼,문자열,html,json모두 전송가능')` : 만능메소드
 - `res.sendFile('전송할 파일경로')`

화면 구성을 위한 뷰(View) 템플릿 엔진 사용하기

View Template Engine

- Pug(Jade) View Engine
 - 웹페이지를 동적으로 구성해주는 뷰 템플릿 엔진
 - Html 페이지 생성 및 제어를 위한 별도의 Pug 문법 제공, 어렵지는 않지만 별도로 Pug문법을 익혀야 한다.
 - **h1 = title => <h1>제목입니다</h1>**

app.set('views', path.join(__dirname,'views')); // View템플릿 파일이 존재하는 폴더위치 지정
app.set('view engine','pug');//

 - **block content**
 - **h1= title**
 - **p Welcome to #{title}**

//Views폴더를 기준으로 index.pug를 찾아 Pug엔진에 의해 pug문법을 해석하고 html내용을 동적으로 생성해서 브라우저로 전달

- EJS View Engine
 - 기본 html문서에 자바스크립트 문법을 추가해 html 내용을 동적으로 조작하는 구조
 - 전통적인 인터프리팅 웹개발언어 방식과 유사한 방식채택(jsp,asp,asp.net mvc,php,python)Npm 팩키지에서 다운로드 설치 필요: **npm i ejs**

app.set('views', path.join(__dirname,'views')); // View템플릿 파일이 존재하는 폴더위치 지정
app.set('view engine','ejs');

 - **<h1><%= title %></h1>**
 - **<p>Welcome to<%= title %></p>**

View Template Engine

- EJS View Engine 주요 코딩 규칙
 - <%= title %> : 라우팅 메소드에서 뷰에 전달된 데이터 출력하기 <%= 데이터속성명 %>
 - If 가정문 사용 예시

```
<% if(userName == "강창훈"){%>
  <h3>환영합니다. 운영자님~~</h3>
<% } else{ %>
  <h3>환영합니다. 사용자님~</h3>
<% }%>
```
 - For 반복문 사용 예시

```
<% for(var i =0;i<children.length;i++){ %>
  <%=i+1%> 번째 이름은 <%=children[i].childName%> 나이는<%=children[i].age%> 입니다.<br>
<% } %>
```
 - 입력요소 사용 예시

```
<input type="text" id="userName" name="userName" value="<%=userName%>">
```

View Template Engine

- EJS 레이아웃 페이지 구현하기
 - 마스터(레이아웃) 페이지 개념 이해하기
 - STEP1) EJS Layout Page 지원 노드 팩키지 추가 설치하기 : `npm i express-ejs-layouts`

STEP2) Views\레이아웃뷰 파일 생성하기

-layout.ejs

-전체 개별 페이지 콘텐츠 페이지 내용을 제외한 레이아웃 페이지 리소스를 구성한다.(헤더/레프트/풋터 등등)

```
<%- body%>
<%- script %>
```

- STEP3) 레이아웃 페이지 사용 app.js 설정하기

~ app.js 파일내 하기내용을 설정한다.

//expressLayouts 모듈을 참조한다.

- var expressLayouts = require('express-ejs-layouts');
- //ejs-layouts setting
- app.set('layout', 'layout'); //기본 레이아웃 페이지 뷰 설정하기
- app.set("layout extractScripts", true);
- app.use(expressLayouts);

STEP4) 필요시 include 사용하기

<% include './header.ejs' %> //ejs :ver 2.x버전

<%- include('./header.ejs',{data:"aaa"}) %> //ejs :ver 3.x버전

View 파일 재사용

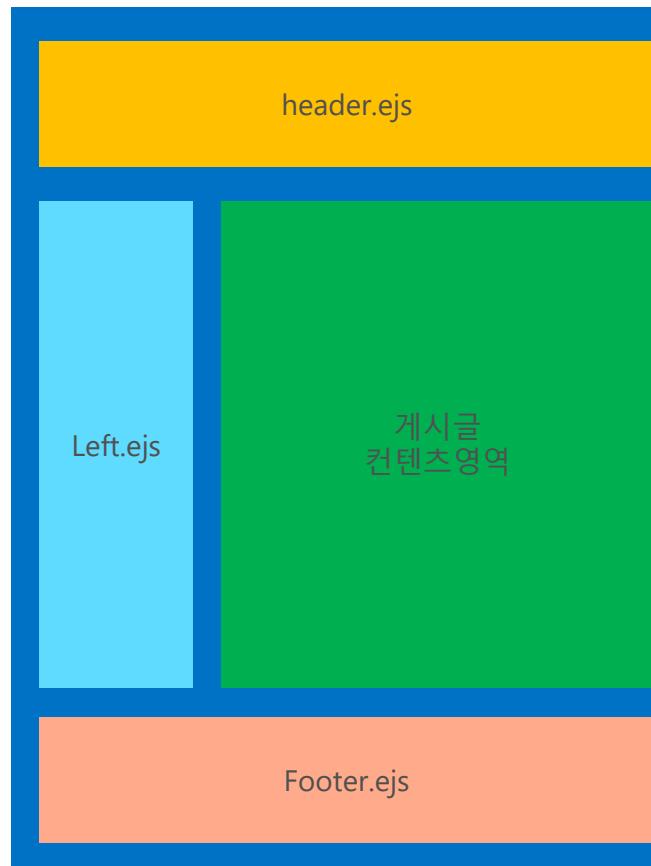
- 인클루드 기술

1) 인클루드 방식

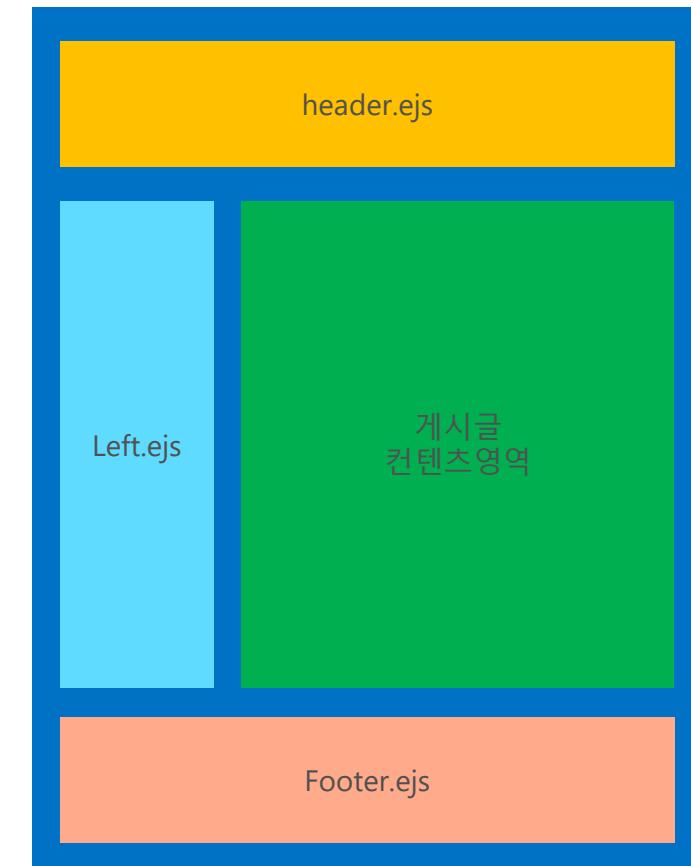
- 헤더 인클루드 파일 : views\header.ejs
- 좌측메뉴 인클루드 파일 : views\leftmenu.ejs
- 풋터 인클루드 파일 : views\footer.ejs



Main.html



Article.ejs



Detail.ejs

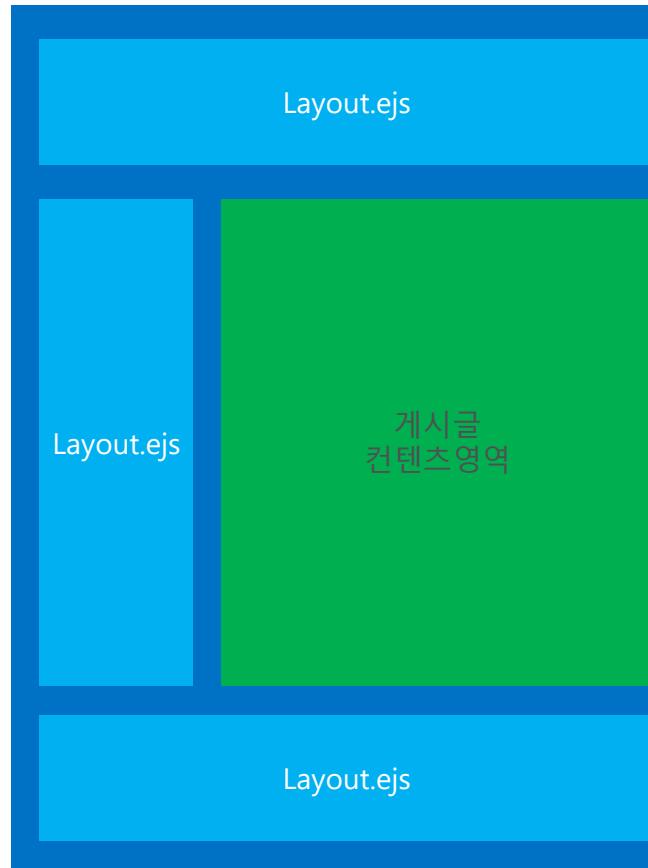
View 파일 재사용

- 마스터(레이아웃) 기술

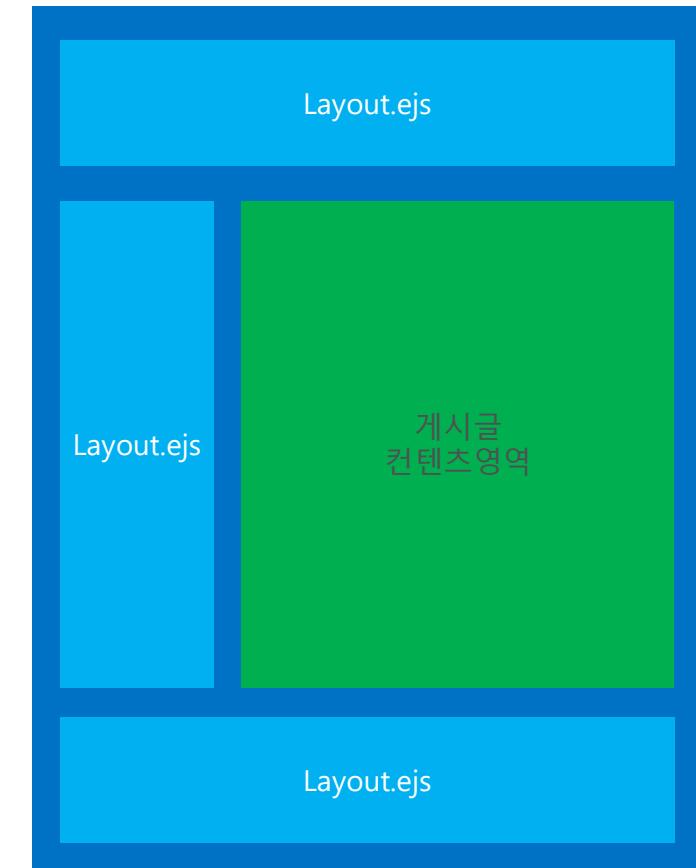
1) 레이아웃 기술
- 레이아웃 파일 : views\Layout.ejs



Main.html



Article.ejs



Detail.ejs

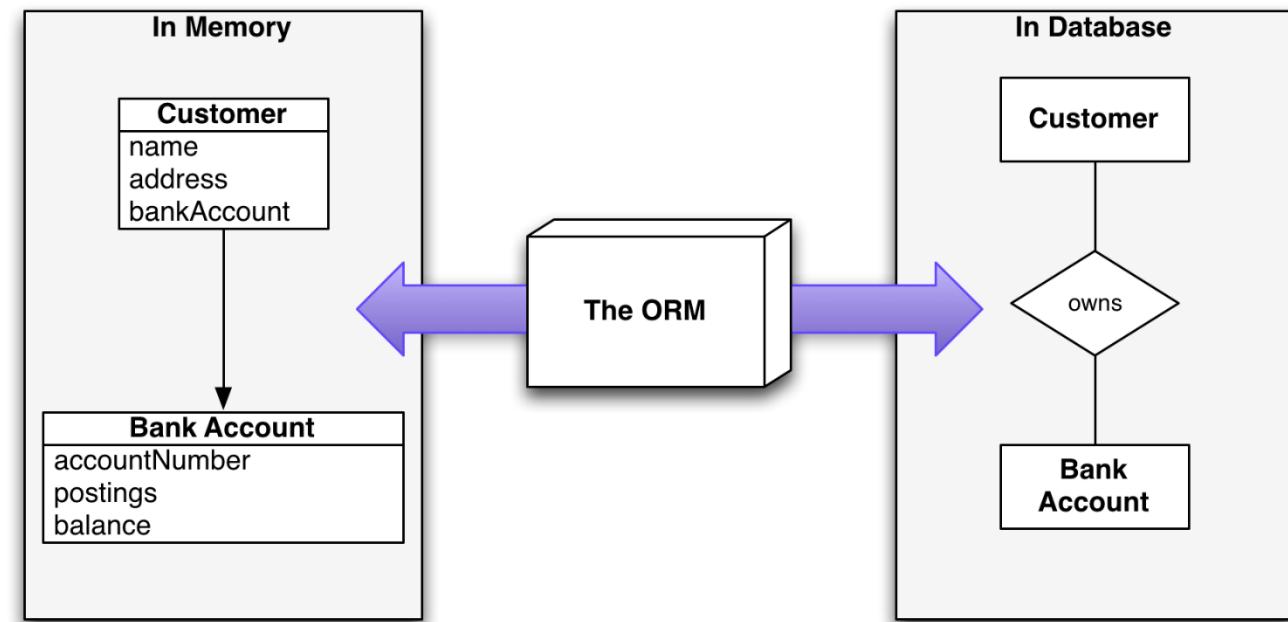
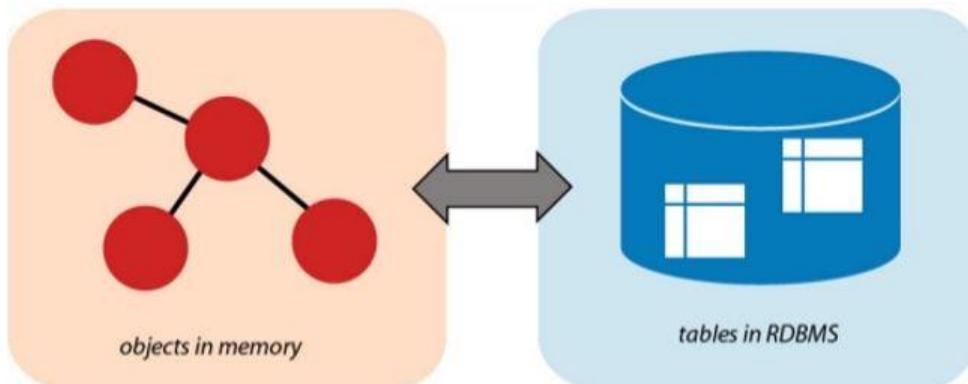
ORM(Object Relational Mapping)

ORM 개념도

What is ORM?

Object Relational Mapping

ORM은 객체지향 프로그래밍언어를 이용해 DB 프로그래밍을 할 때 RDBMS의 데이터베이스 구조 및 데이터를 관리하기 위한 SQL(Structured Query Language)를 잘 모르더라도 프로그래밍적인 방법을 이용 쉽게 데이터 모델 객체 기반으로 DB 프로그래밍을 할 수 있는 환경을 제공합니다.



ORM

- **ORM 장점**
 - ORM이란 DB의 테이블과 맵핑되는 프로그램의 데이터 모델 클래스를 이용해 DB프로그램을 구현하는 방법
 - **CRUD작업을 위해 SQL 언어를 직접 사용하는것이 아닌 프로그램 언어를 이용해 CRUD 작업이 가능**
 - SQL을 몰라도 어플리케이션 개발언어를 이용해 데이터 처리업무를 개발한다(내부적으로는 SQL로 자동변환 RDBMS에서 실행)
 - ORM이 나온 배경은 데이터 처리 업무를 위해 별도로 표준 SQL(ANSI-SQL)을 배워야하고 RDBMS마다 표준 SQL(ANSI-SQL)을 사용하지 않는 상황이 많아 RDBMS마다 특화된 SQL사용법을 별도 알아야 하는 개발자들에게 개발자 친숙한 개발언어로 DB프로그래밍을 할수 쉽게 처리할수 있는 환경을 제공코자 함.
 - **ORM을 이용하면 RDBMS 종속적인 어플리케이션이 아닌 RDBMS를 변경해도 쉽게 대응이 가능하다.**
 - **최소한의 SQL 언어는 알아야 DB에 저장된 데이터에 대한 조작이나 고급 쿼리 작성시 도움된다.**
- **Model?**
 - Model이란 데이터의 구조를 프로그래밍 언어로 표현한 클래스
 - 일반적으로 데이터를 저장하는 개별 테이블의 구조와 맵핑되는 모델 클래스를 생성한다.
- **Model의 유형**
 - **Data Model** : 데이터의 구조를 클래스의 속성으로 표현하고 데이터를 저장하는것이 목적, RDBMS Table과 맵핑
 - **Domain Model** : 실제 업무 기준으로 데이터 구조를 표현하며 테이블과는 구조가 다소 상이할수 있다.
 - **View Model** : 뷰(화면)에서 데이터 바인딩을 위해 전문적으로 사용하는 뷰 전용 모델
 - **DTO Model** : Data Transfer Object ;프로그래밍 계층간 대량(복합) 데이터 전송을 위해 여러 모델을 하나의 전송모델로 재구성한 모델

시퀄라이즈 ORM 활용하기

시퀄라이저 설치

- 시퀄라이저(Sequelize)
 - Node(Javascript) 기반의 ORM 지원 팩키지 모듈
 - NPM 을 통해 별도 팩키지 설치 필요
 - Express 프로젝트 생성 이후 추가 설치
- STEP1) 시퀄라이저(Sequelize) ORM 관련 팩키지 설치 및 프로젝트 ORM 구성하기
 - **npm install sequelize mysql2** : sequelize,mysql2 노드팩키지 2개동시 설치
 - **npm install -global sequelize-cli** : sequelize 명령어 팩키지 전역설치
 - **sequelize init** : 현재 노드 프로젝트에 ORM 프로그램 환경 구성해주는 sequelize 명령어
- STEP2) 시퀄라이저 ORM 프로젝트 주요폴더 설명
sequelize init 명령어 실행이후 시퀄라이즈 ORM은 현재 노드 프로젝트에 ORM 프로젝트 구조를 추가 구성해줍니다.
 - └ config : DB연결정보 환경설정 파일
 - └ models : 테이블과 맵핑되는 모델 정의 파일 보관 및 모델 기반 프로그래밍 처리 모듈 제공
 - └ migrations : 구조변경 마이그레이션 계획정보 기록
 - └ seeders : 테이블 생성시 초기 데이터 생성설정

시퀄라이저 개발환경 구성

- STEP3) 시퀄라이저 ORM 모델 관리 index.js 모듈 코딩하기

- Models\index.js 모듈파일내 자동 생성코드에 에러가 있으니 기존소스 삭제하고 하기소스를 붙여넣기 저장합니다.
- Models\index.js 모듈은 sequelize orm의 핵심파일로 DB연결 및 DB제어 모델파일 연결/관계설정 등을 관리하는 핵심 모듈임.

```
const path = require('path');
const Sequelize = require('sequelize');

//개발모드 환경설정
const env = process.env.NODE_ENV || 'development';

//DB연결 환경설정정보 변경처리//관련정보 수정
const config = require(path.join(__dirname,'..','config','config.json'))[env];

//데이터 베이스 객체
const db= {};

//DB연결정보로 시퀄라이즈 ORM 객체 생성
const sequelize = new Sequelize(config.database,config.username,config.password,config);

//DB 처리 객체에 시퀄라이즈 정보 맵핑처리
//이후 DB객체를 통해 데이터 관리 가능해짐
db.sequelize = sequelize; //DB연결정보를 포함한 DB제어 객체속성(CRUD)
db.Sequelize = Sequelize; //Sequelize팩키지에서 제공하는 각종 데이터 타입 및 관련 객체정보를 제공함

//회원모델 모듈파일 참조하고 db속성정의하기
//db.Member = require('./member.js')(sequelize,Sequelize);

//db객체 외부로 노출하기
module.exports = db;
```

시퀄라이저로 DB 프로그래밍 하기

- STEP4) MY SQL DB 연결정보 환경설정 수정
 - Config 폴더내 config.json 파일 오픈 후 환경에 맞는 DB연결정보(development) 수정 후 저장
- STEP5) Node Application에 MY SQL 연결 설정하기- app.js
 - App.js 파일에 시퀄라이즈 index.js 모듈 참조 연동하기
 - `var Sequelize = require('./models/index.js').Sequelize;`
`var app = express();`
`//mysql과 자동연결처리 및 모델기반 물리 테이블 생성처리제공`
`Sequelize.sync();`

시퀄라이저로 DB 프로그래밍 하기

- STEP6) 물리 테이블과 1:1 맵핑되는 모델모듈파일 생성하기
 - models\테이블에 맵핑되는 모델 모듈파일 생성하고 정의하기 ex) models\member.js

```
module.exports = (sequelize, DataTypes) => {  
  
    //member 테이블과 맵핑되는 member모델 정의  
    //return sequelize.define()메소드를 통해 물리테이블과 맵핑되는 모델클래스를 생성하고 반환한다.  
    //sequelize.define('맵핑되는 물리 테이블명',{테이블의 데이터구조 정의},{테이블생성 옵션정보})  
    //맵핑되는 물리 테이블명은 단수형태로 정의할것..실제생성되는 물리테이블은 복수형태로 생성됨. member(모델명) ->members(물리테이블명)  
    //{테이블의 데이터구조 정의} = {속성(컬럼)명:{각종세팅정보정의(데이터타입,null허용여부,primaryKey여부,자동채번여부.)},속성(컬럼)명:{}},속성(컬럼)명:{}  
    return sequelize.define('member',  
    {  
        userid: {  
            type: DataTypes.STRING(100),  
            allowNull: false  
        },  
        username: {  
            type: DataTypes.STRING(100),  
            allowNull: true  
        },  
    },  
    {  
        timestamps: true,  
        paranoid: true  
    });  
  
    //timestamps 는 물리적 테이블 createdAt,updatedAt컬럼을 자동추가하고  
    //데이터 신규생성일시,수정일시 데이터를 자동으로 마킹해줍니다.  
    //paranoid가 트루이면 deletedAt컬럼이 자동추가되고  
    //삭제시 삭제일시정보가 자동 마킹되고 데이터는 실제 삭제되지 않습니다.  
};
```

시퀄라이저로 DB 프로그래밍 하기

- STEP7) 생성한 모델 모듈파일 index.js 에 통합하기
 - Models\index.js 파일내에 생성한 모델파일 참조하여 db속성으로 노출하기

```
//회원모델 모듈파일 참조하고 db속성정의하기  
db.Member = require('./member.js')(sequelize,Sequelize);
```

시퀄라이저로 DB 프로그래밍 하기

• 모델 정의하기

- 데이터베이스에 존재하는 테이블과 맵핑되는 시퀄라이즈 모델 자바스크립트 모듈 작성
- 모델 이름은 단수형, 테이블명은 복수형으로 관리된다.
- 모델을 이용하면 데이터베이스에 해당 모델구조로 테이블을 자동 생성하고 데이터를 관리해준다.(DB에 기존 생성 테이블 삭제)
- 시퀄라이즈는 id를 기본키로 연결하므로 id 컬럼은 정의하지 않는다.
- Timestamp 옵션이 true이면 테이블에 자동으로 createdAT,updatedAT 컬럼이 추가되고 등록과 수정정보를 자동관리해줌.
- Paranoid 옵션이 true이면 테이블에 자동으로 deletedAt 컬럼이 생성되고 데이터 삭제시 삭제일시가 기록되며
실제 물리 테이블에서 데이터는 삭제되지 않으며 어플리케이션에서는 삭제된것으로 인식함
- DB 자료형과 프로그래밍 처리 자료형의 차이
`VARCHAR,CHAR,TEXT => STRING(문자열길이) STRING(100)`
- TEXT => TEXT(긴문자열)
- INT => INTEGER
- TINYINT => TINYINT
- BIT => BOOLEAN
- DATETIME => DATE
- NOT NULL = allowNull:false

예시

```
ARTICLE_PSEQ: {  
  autoIncrement: true, //자동 채번  
  type: DataTypes.INTEGER, //숫자형  
  allowNull: false, //null 허용안함-필수입력  
  primaryKey: true, //pk설정  
  comment: '게시글고유번호', //컬럼 코멘트  
},  
REG_DT: {  
  type: DataTypes.DATE,  
  allowNull: false,  
  defaultValue: DataTypes.NOW  
  comment: '등록일시',  
},
```

시퀄라이저로 DB 프로그래밍 하기

- 정의된 모델 파일 index.js에 연결하기
 - /models/index.js 파일에 모델 파일들을 추가해준다.
- 관계 정의하기
 - 테이블간 관계를 모델에서도 정의해준다.
 - 모델을 통한 관계정의시 참조키는 해당 모델파일에서 정의하지 않는다.
 - 1:N 관계 : **hasMany** 메소드(내값을 참조하고 있는 테이블의 목록) , **belongsTo**메소드(참조 테이블의 특정 로우값)
 - 1:1 관계: 하나의 주문에 하나의 배송정보만 존재한다. **hasOne**메소드 과 **belongsTo**메소드를 사용
 - N:M 관계: 게시글과 해시태그 테이블 관계 **belongsToMany**로 표현
 - 관계정보는 /models/index.js파일에 정의한다.
- 모델 정의하기 주의사항 정리
 - 모델을 이용하면 데이터베이스에 해당 모델구조로 테이블을 자동 생성하고 데이터를 관리해준다.
 - 모델 정의시 Timestamp옵션이 true이면 테이블에 자동으로 createdAT,updatedAT컬럼이 추가되고 등록과 수정정보를 자동관리해줌.
 - 시퀄라이즈는 id를 기본키로 연결하므로 id 컬럼은 정의하지 않는다.
 - 테이블 참조키 컬럼은 해당 모델에서 정의할수도 있고 테이블간의 관계설정 정보를 별도 정의할수도 있다.
 - 모델을 모두 만들고 나면 DB에 기존 생성 테이블들을 삭제해준다.(테이블존재시 에러발생)
 - 관계정보는 /models/index.js파일에 정의한다.

JWT(Json Web Token) 프로세스

JWT Process

opennaru

CLIENT

SERVER

1

사용자가 id와 password를 입력하여 로그인

3

클라이언트에 JWT 전달

4

서비스 요청과 권한을 확인하기 위해
헤더에 JWT 전달

6

클라이언트에 요청에 대한 응답 전달

2

서버는 요청을 확인하고
secret key를 통해
Access token을 발급

5

JWT 서명을 체크하고 JWT에서
사용자 정보를 확인

추가 팩키지 설치

- Nodemon 팩키지
 - 개발환경에서 노드 서버를 항상 실행되는 환경을 제공한다.
 - 노드 소스 변경 후 매번 재실행 시켜주는 번거로움을 줄여준다.
 - 팩키지를 전역으로 설치 후 프로젝트에는 개발용으로만 설치한다.
 - 코드 수정 후 서버를 자동으로 재시작해준다.
 - **npm i -g nodemon**
 - **npm i -D nodemon**
- Dotenv 팩키지
 - 주요 키값을 소스상에 하드코딩 하지 별도 환경설정 파일에 저장하고 사용관리하기 위한 용도
 - 전역 환경 설정파일인 .env파일을 이용해 노드 어플리케이션 전역에서 접근가능한 process.env 객체에 정보를 맵핑하여 활용한다.
 - .env 파일을 만들고 주요 비밀키를 해당 파일에 저장후 dotnet 팩키지가 자동으로 .env파일을 읽어 process.env 객체 넣어줍니다.
 - **npm i dotenv 팩키지 설치 후 프로젝트 Root에 .env파일을 생성합니다.**
 - **.env파일내에 키=값 형식으로 설정값을 추가합니다.**
 - **app.js 설정 처리**
require('dotenv').config();
 - 설정된 env 키값은 process.env.키명 으로 정보를 추출하여 사용한다.

JWT(Json Web Token)?

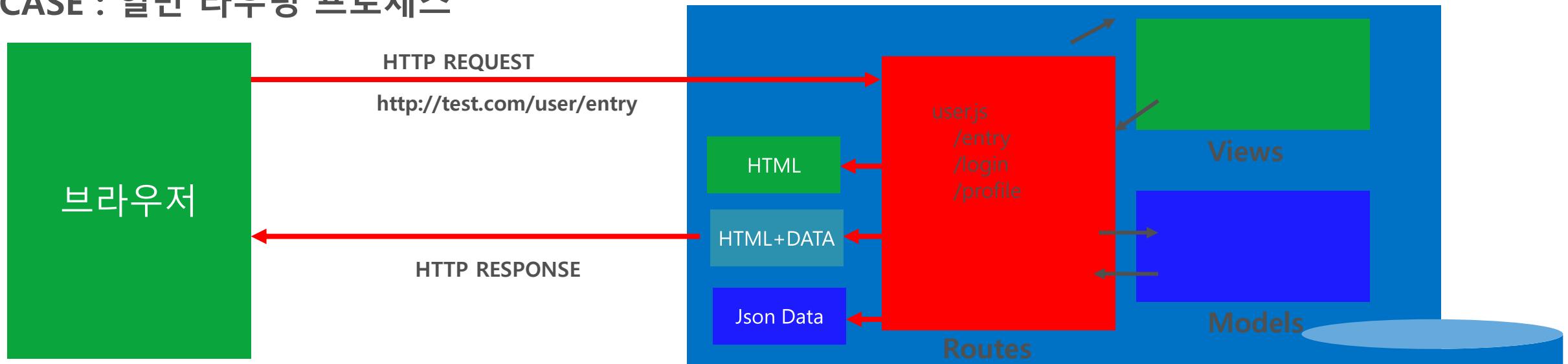
- OPEN API Server 사용 인증
 - 제공하는 OPEN API 서비스를 허가된 사용자에게만 서비스하기 위한 수단 제공
 - 제공되는 OPEN API 서비스(라우트)는 크게 모든이에게 공개되는 OPEN API 기능과 회원으로 로그인한 사용자에게만 제공하는 OPEN API 기능 두가지 유형으로 나뉜다.
 - 인증된 사용자에게만 제공되는 OPEN API는 일반적으로 서버에서 인증토큰을 사용자 브라우저에게 발급하고 매번 서비스 요청시 발급된 토큰을 통해 인증과정을 거쳐 OPEN API를 사용할수 있게 처리한다.
- JWT 토큰
 - JSON 형식의 특정 데이터를 암호화 하여 저장할 수 있는 토큰
 - 회원으로 인증된 사용자에게 JWT토큰을 발급하고 발급된 토큰을 기반으로 허가가 필요한 OPEN API 라우팅 호출시 인증수단으로 활용
 - **JWT 토큰의 형식**
 - └ 헤더: 토큰 종류와 해시알고리즘 정보 제공
 - └ 페이로드: 토큰의 내용인 실제 전송 데이터를 인코딩 하여 저장하는 영역(주로 로그인 사용자의 사용자아이디/이름/기타정보 저장)
 - └ 시그니처 : 일련의 문자열로 서버에서 발급해준 특정 문자열로 시그니처 값을 통해 서버의 값과 비교해 토큰이 변조되었는지 여부를 확인할수 있는값 설정.
- **JWT 토큰값 확인하기**
 - <https://jwt.io> 사이트를 통해 발급된 토큰값들을 확인할수 있다

JWT(Json Web Token) 발급 과 확인

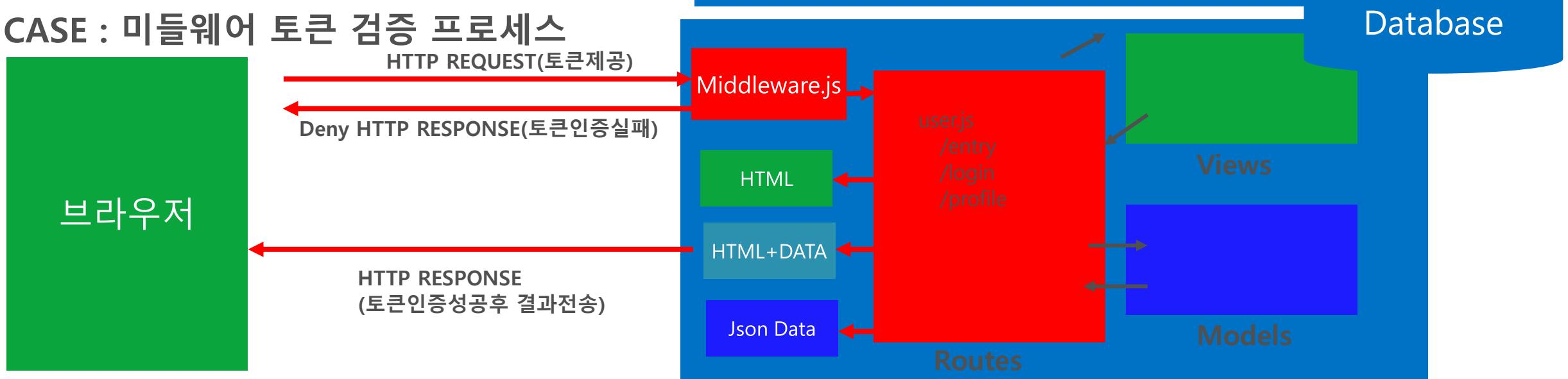
- **JWT 토큰 발급하기**
 - 토큰의 발급은 해당 시스템의 로그인 시점에서 주로 발급한다.
 - 인증된 사용자에 대해서만 토큰을 발급하고 발급된 토큰을 통해 로그인한 사용자 정보를 확인할수 있다.
 - 발급된 토큰은 쿠키처럼 브라우저에저장되어 HTTP요청시마다 헤더에 담겨 서버로 전송된다.
- **JWT 토큰 검사하고 라우팅 서비스 제공하기**
 - 헤더에 담겨 서버로 전송된 토큰내의 인증값을 서버에 저장해둔 인증값과 비교하여 토큰의 유효성을 검증한다.
 - URL 요청에 대한 토큰이 검증된 경우 해당 URL 라우팅 서비스를 제공한다.
- **회원가입 처리시 단방향 암호화 팩키지 bcryptjs 팩키지 설치하기**
 - `npm i bcryptjs`

라우팅 호출 및 미들웨어 인증 과정

CASE : 일반 라우팅 프로세스

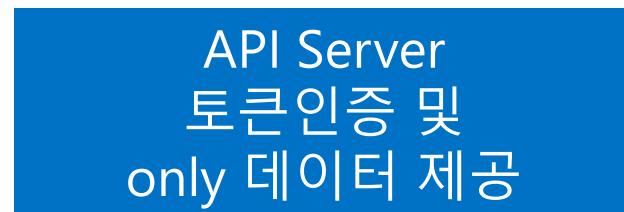


CASE : 미들웨어 토큰 검증 프로세스

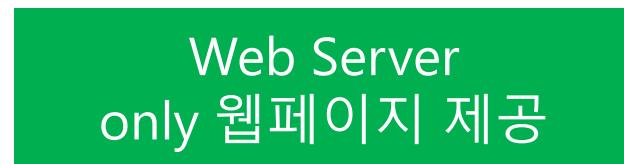


JWT(Json Web Token) 활용 방식

CASE1 :Backend Base



CASE2 : FRONTEND/ BACKEND Base



사용자 로그인

토큰발급처리

브라우저 저장된 토큰과 함께 API 호출

API 데이터 제공

API Server
사용자 인증 및 토큰발급
토큰인증,데이터제공

JWT(Json Web Token) 인증서버 구축하기

- **JWT 토큰 사용하기**

- **npm i jsonwebtoken** JWT 모듈 팩키지 설치
- 모든 데이터 처리 OPEN API 라우터에서 공용으로 사용하기 위한 공통 JWT 인증 라우팅 미들웨어 추가
- **middlewares.js**
- 클라이언트에게 발급하는 인증코드(키) 저장용 env 키 설정

1).env 파일내 클라이언트에게 발급할 인증값 설정

- **npm i dotenv** 팩키지 설치
- **app.js 설정 처리**
`require('dotenv').config();`
- 프로젝트 Root에 .env파일을 생성합니다.
- .env파일내에 키=값 형식으로 비밀키를 추가합니다.

- OPEN API 서버에 접근시 반드시 클라이언트는 해당 키값을 가지고 접근하고 해당 값을 OPEN API 서버에서는 그 값을 비교하여 인증처리한다.

JWT_SECRET = jwtSecret //값은 임의의 텍스트값으로 설정하고 클라이언트에게 제공

2)라우팅 공용 인증기능 제공 미들웨어 모듈 추가 (routes\middlewares.js)

CORS 적용하기

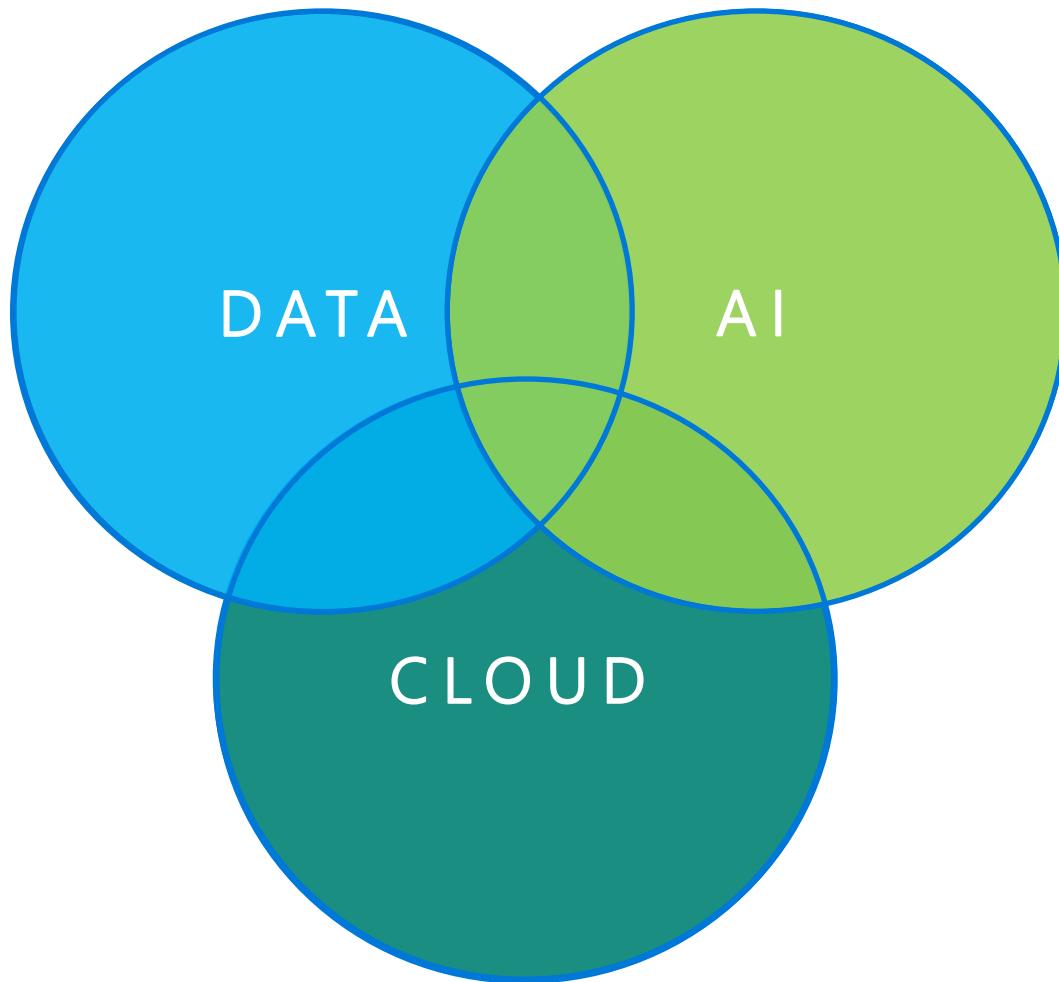
- CORS (Cross-Origin-Resource-Sharing)
 - OPEN API를 요청하는 웹페이지(리소스) 주소가 OPENAPI 서비스 와 동일한 도메인주소가 아닌경우 크로스 도메인 문제 발생
 - 클라이언트에서 OPEN API서버로 데이터 처리 요청을 보낼때 클라이언트 제공서버와 OPEN API서버의 도메인이 일치하지않을 때 발생
 - 기본적으로 웹브라우저는 OPENAPI 주소와 해당 주소를 사용하는 웹페이지가 같은 서비스 제공자(위치)가 아니면 OPEN API호출불가함
 - OPEN API 서버를 로컬 컴퓨터의 로컬 HTML파일이나 다른서버,모바일앱(네이티브, 하이브리드앱(내장))의 경우 주로 크로스 도메인 문제
 - 발생함
 - CORS 문제를 해결하려면 서버에서 클라이언트로 응답을 보낼때 응답헤더에 Access-Control-Allow-Origin이라는 헤더를 넣어주어 서버 도메인과 다른 클라이언트 도메인의 요청을 허락하게 한다.
- NODE에서 CORS 문제 해결
 - Cors 팩키지설치 후 라우팅 공통 미들웨어에 core 미들웨어 기능추가
 - **npm i cors 설치**
 - 인증 라우팅파일에
const cors = require('cors');
 - **router.use(cors());**
 - **//cors 커스터마이징을 통해 특정 도메인만 cors적용가능하게 처리가능 book page: 426p**
 - **Router.use((req,res,next)=>{**
 - **Cors()(req,res,next);**
 - **})**

Open API 서버 기반 인증/권한 구축하기

- CORS (Cross-Origin-Resource-Sharing)
 - OPEN API를 요청하는 웹페이지(리소스) 주소가 OPENAPI 서비스 와 동일한 도메인주소가 아닌경우 크로스 도메인 문제 발생
- NODE에서 CORS 문제 해결
 - Cors 팩키지설치 후 라우팅 공통 미들웨어에 core 미들웨어 기능추가

지금은 클라우드 세상

Cloud Computing & Services



Organizations that harness Data, Cloud, and AI outperform

Cloud Computing & Services

Cloud Computing?

- 글로벌 클라우드 기업의 스토리지, 실제 서버 리소스를 대여 받아 사용하는 방식
- 사용한 양만큼만 요금 지불
- 클라우드 업체는 고객사 컴퓨팅 작업을 위한 실제 하드웨어를 제공
- 클라우드 업체는 하드웨어 가상화 기반 하드웨어 사용을 극대화하여 이윤창출

Cloud Computing 제공 서비스

- 컴퓨팅 능력 : 서버 또는 웹 어플리케이션
- 스토리지: 파일 및 데이터 베이스
- 네트워킹: 클라우드 공급기업과 회사간의 보안연결
- 분석 : 원격 분석데이터 및 성능 데이터 시각화

1. Cloud Computing & Services

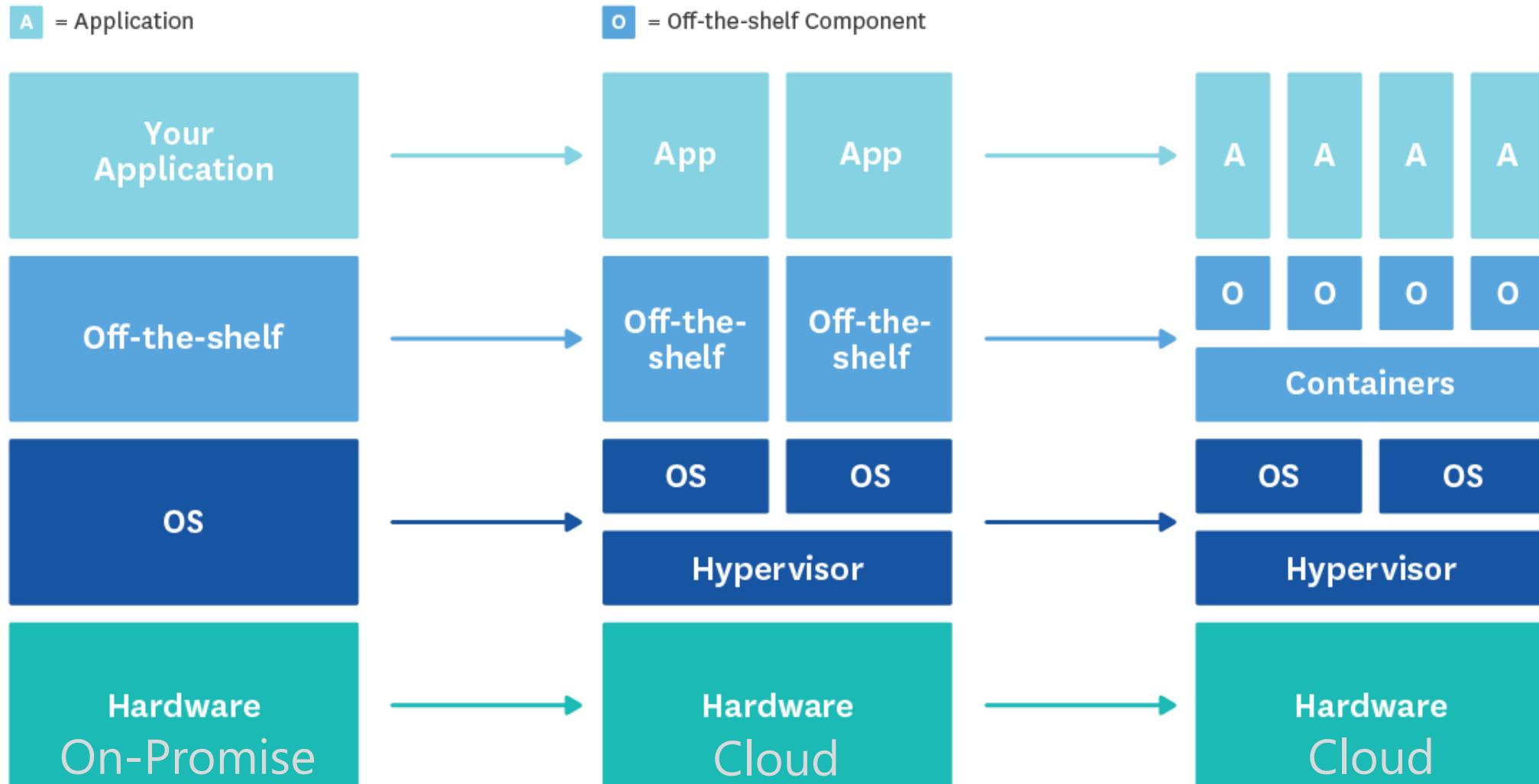
1.1 Cloud Computing?

“클라우드는 하드웨어 가상화 기술로 H/W자원활용을 S/W적으로 극대화 시킬 수 있는 기술”



Cloud Computing & Services

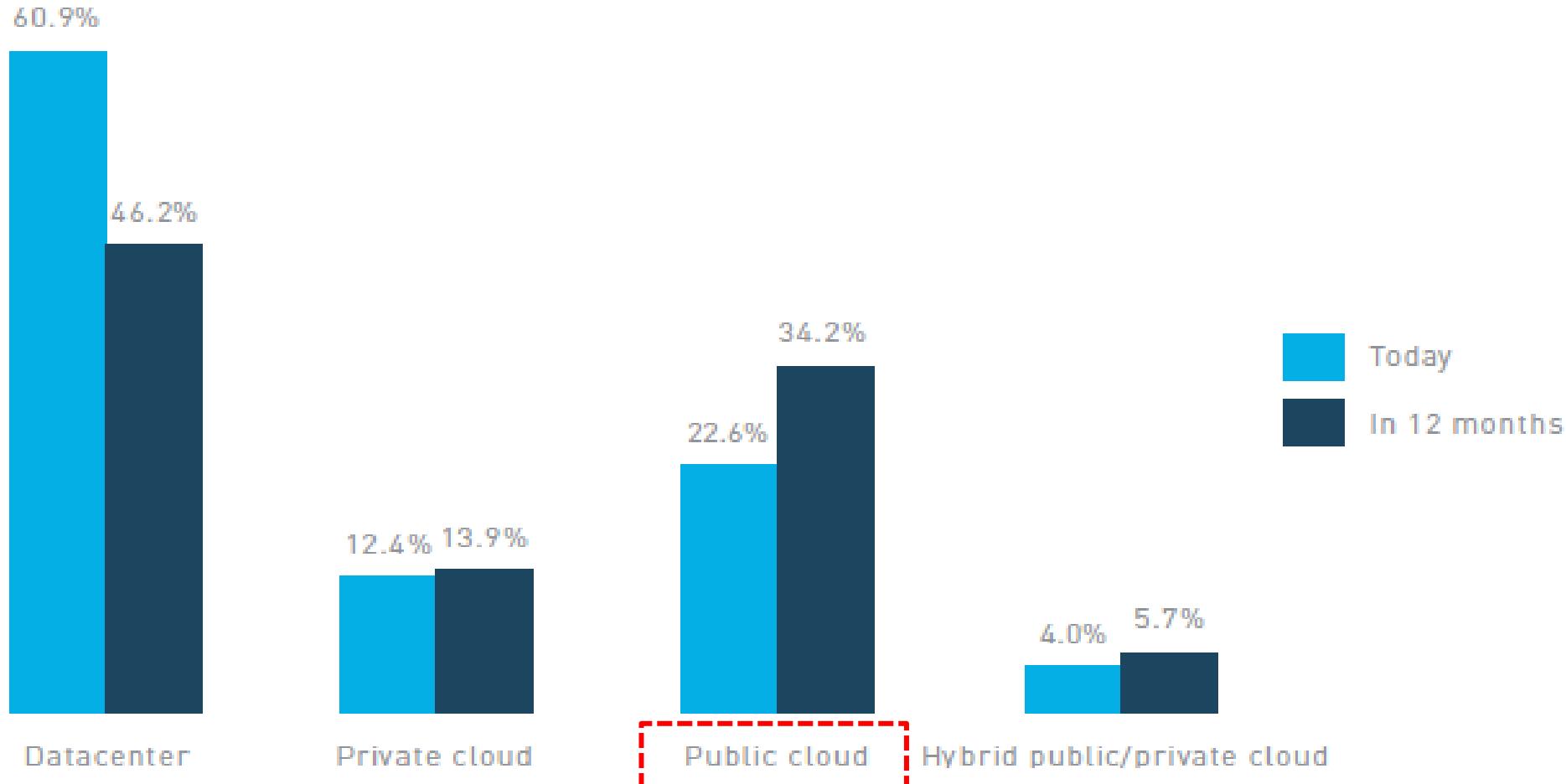
Cloud Computing History



클라우드는 하드웨어 가상화 기술로 H/W자원활용을 S/W적으로 극대화 시킬 수 있는 기술

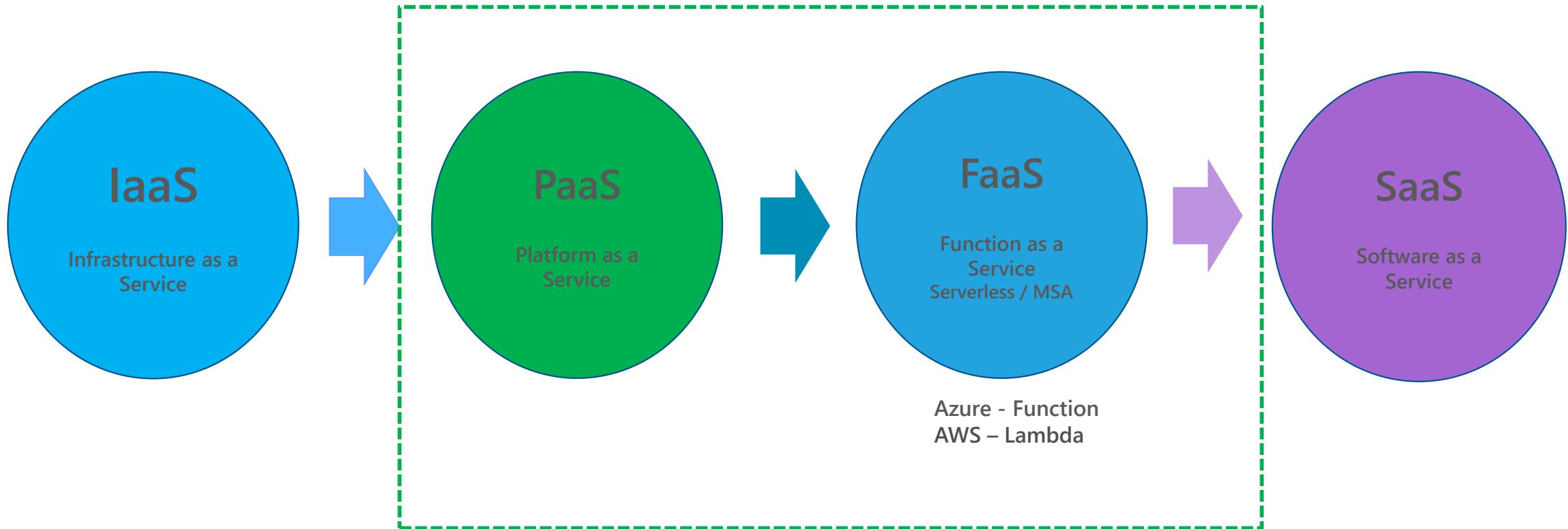
Cloud Computing & Services

Cloud Service Types



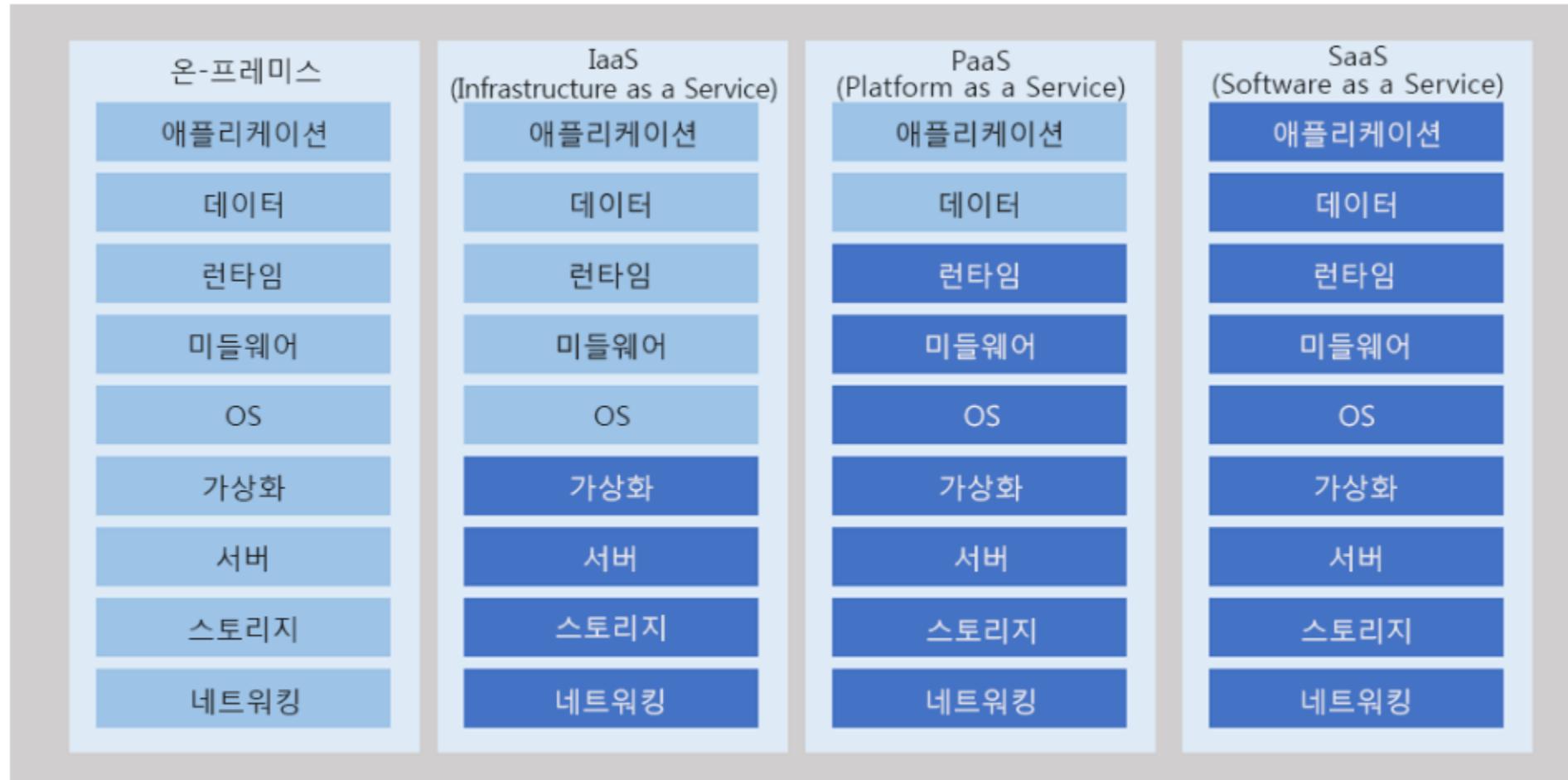
Cloud Computing & Services

Cloud Services Types



Cloud Computing & Services

Cloud Services Types



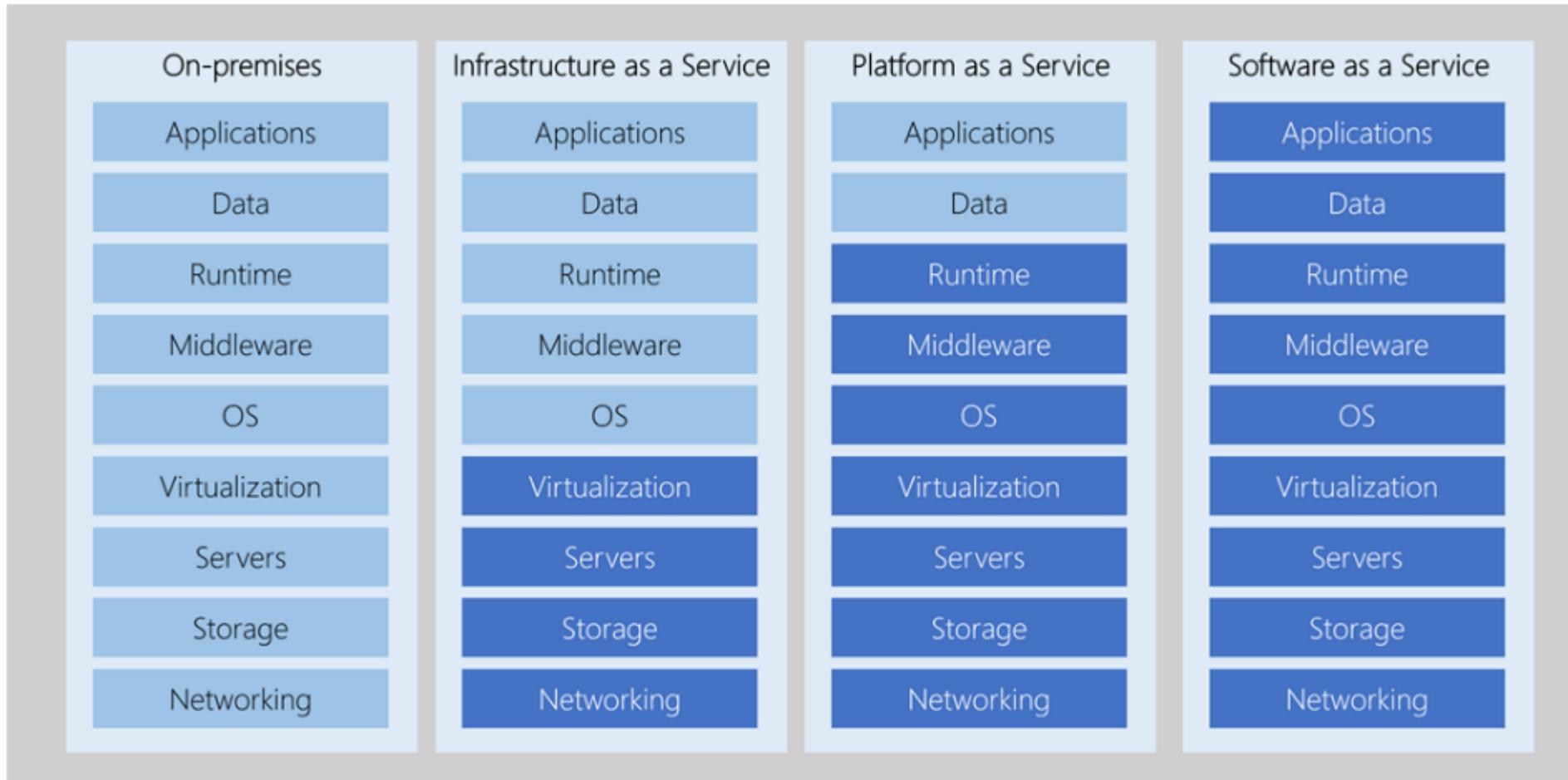
사용자 관리



공급자 관리

Cloud Computing & Services

Cloud Services Types



You Manage



Provider Manages

오늘 하루도
수고 많으셨습니다.

재미있고 행복한 코딩하세요.