



JSON/Dictionary/FILE IO

정보 기술을 위한 format

CSV : comma-separated values

EXCEL : 마이크로소프트사에서 개발해 판매하는 스프레드시트

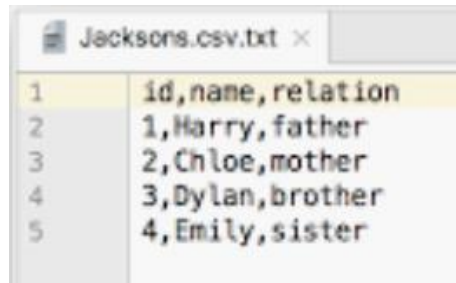
HTML : Hyper Text Markup Language

웹 페이지를 구조적 문서 표현 방법

XML : Extensible Markup Language, 마크업 언어를 사용해 데이터를 기술

JSON : JavaScript Object Notation

속성-값 쌍 으로 이루어진 데이터 오브젝트로 개방형 표준 포맷



id	name	relation
1	Harry	father
2	Chloe	mother
3	Dylan	brother
4	Emily	sister



	Maxwell	Chase	Thompson	Getsinger
January	\$4,212.64	\$5,624.12	\$4,321.89	\$4,796.06
February	\$5,545.08	\$7,654.00	\$4,456.98	\$4,765.84
March	\$5,465.98	\$6,566.74	\$3,567.92	\$6,651.12
April	\$5,099.07	\$5,543.23	\$6,546.43	\$7,765.34
May	\$6,609.99	\$6,543.23	\$4,410.12	\$7,899.19
June	\$6,195.00	\$4,565.78	\$4,123.11	\$9,564.21
July	\$5,567.04	\$4,567.89	\$5,567.34	\$4,321.75



```
<!DOCTYPE html>
<html>
<!-- created 2010-01-01 -->
<head>
<title>sample</title>
</head>
<body>
<p>Voluptatem accusantium
totam rem aperiam.</p>
</body>
</html>
```

HTML



```
<?xml version="1.0"?>
<quiz>
  <qanda seq="1">
    <question>
      Who was the forty-second
      president of the U.S.A.?
    </question>
    <answer>
      William Jefferson Clinton
    </answer>
  </qanda>
  <!-- Note: We need to add
  more questions later.-->
</quiz>
```

XML



```
{
  hey: "guy",
  anumber: 243,
  - anobject: {
    whoa: "nuts",
    - anarray: [
      1,
      2,
      "thr<h1>ee"
    ],
    more: "stuff"
  },
  awesome: true,
  bogus: false,
  meaning: null,
  japanese: "明日がある。",
  link: http://jsonview.com,
```

CSV(Comma Separated Values)

- 몇 가지 필드를 쉼표(,)로 구분한 텍스트 데이터 및 텍스트 파일
- 단순한 파일 포맷이며 여러 애플리케이션에서 널리 사용
- 호환되지 않는 포맷을 사용하는 프로그램 끼리 자료를 전달할 때 사용
- 각 컬럼의 이름을 포함시켜 헤더를 포함할 수 있음
- #을 사용해 주석을 포함시킬 수 있음

연도	제조사	모델	설명	가격
1997	Ford	E350	ac, abs, moon	3000.00
1999	Chevy	Venture "Extended Edition"		4900.00
1999	Chevy	Venture "Extended Edition, Very Large"		5000.00
1996	Jeep	Grand Cherokee	MUST SELL! air, moon roof, loaded	4799.00

연도,제조사,모델,설명,가격

1997,Ford,E350,"ac, abs, moon",3000.00

1999,Chevy,"Venture ""Extended Edition""",,,4900.00

1999,Chevy,"Venture ""Extended Edition, Very Large""",,5000.00

1996,Jeep,Grand Cherokee,"MUST SELL!air, moon roof, loaded",4799.00

XML(Extensible Markup Language)

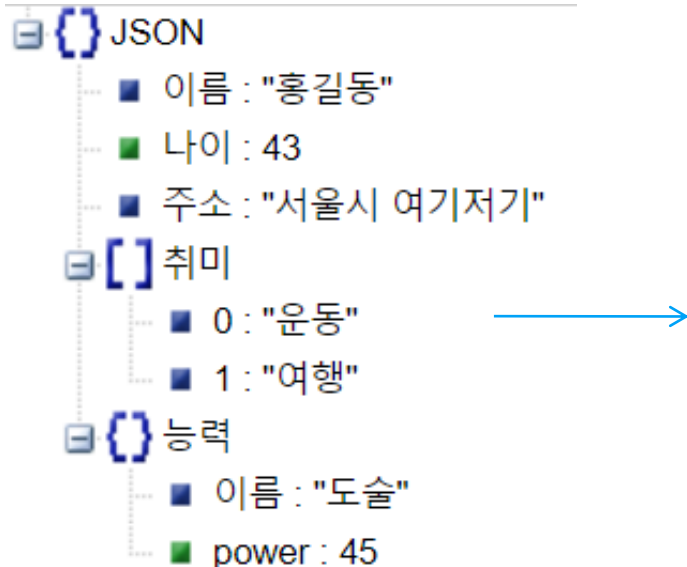
- 사람과 기계가 모두 읽을 수 있는 데이터를 기술하는 데 사용
- 태그/속성/text로 이루어짐
- 태그는 open, close로 구성
- 계층구조를 통해 복잡한 데이터 구조를 표현할 수 있음
- XML로 만들어진 문서가 주어진 문법을 준수하는지 문서 검증할 수 있는 XML을 정의할 수 있음

rss 태그 version 속성 text

```
<rss version="2.0">
  <channel>
    <title>Naver Open API - blog :: 'hello'</title>
    <link>http://search.naver.com</link>
    <description>Naver Search Result</description>
    <lastBuildDate>Wed, 13 Jun 2018 17:17:03 +0900</lastBuildDate>
    <total>625263</total>
    <start>1</start>
    <display>10</display>
  <item>
    <title><b>HELLO</b> HERO, 게이밍 마우스로지텍 G304</title>
    <link>
      http://blog.naver.com/feature?categoryId=681&logNo=221282216872
```

JSON(Java Script Object Notation)

- 자바스크립트 기반의 객체 지향 데이터 기술 방법
- 자바스크립트 언어로부터 파생되어 자바스크립트의 구문 형식을 따르지만 언어 독립형 데이터 포맷(프로그래밍 언어나 플랫폼에 독립적)
- Name/Value 형식으로 데이터 정의
- 이름/값 쌍의 집합으로, 중괄호{} 사용



name : value 표현

```
{
  "이름": "홍길동",
  "나이": 43,
  "주소": "서울시 여기저기",
  "취미": ["운동", "여행"],
  "능력": { "이름": "도술", "power": 45 },
}
```

array

객체

JSON(Java Script Object Notation)

- 데이터 양이 많고 사용 방식이 번거로운 XML 문제를 해결
- 비동기 브라우저/서버 통신 (AJAX)을 위해, 넓게는 XML을 대체하는 데이터 포맷
- 데이터 양을 줄이고 객체 접근과 문자열 객체화 방식이 용이함

XML 방식

```
<root> ↵
  <book> ↵
    <name>Mashup Guide</name> ↵
    <url>www.asjs.net</url> ↵
  </book> ↵
  <book> ↵
    <name>AJAX Guide</name> ↵
    <url>www.asjs.net</url> ↵
  </book> ↵
</root> ↵
```

JSON 방식

```
{ ↵
  book : [ ↵
    {"name" : "Mashup Guide", "url" : "www.asjs.net"}, ↵
    {"name" : "AJAX Guide", "url" : "www.asjs.net"} ↵
  ] ↵
} ↵
```


어떻게 가능하지?

N

지도 홈

길찾기

버스

지하철

즐거찾기

더보기

경기도 화성시

도보 323m

5분

화성시청 승차

36-028

직행 G1003

7개 정류장 이동

지하철3호선교대역 하차

교대역 14번 출구까지 도보 197m

3분

3호선 교대역 승차

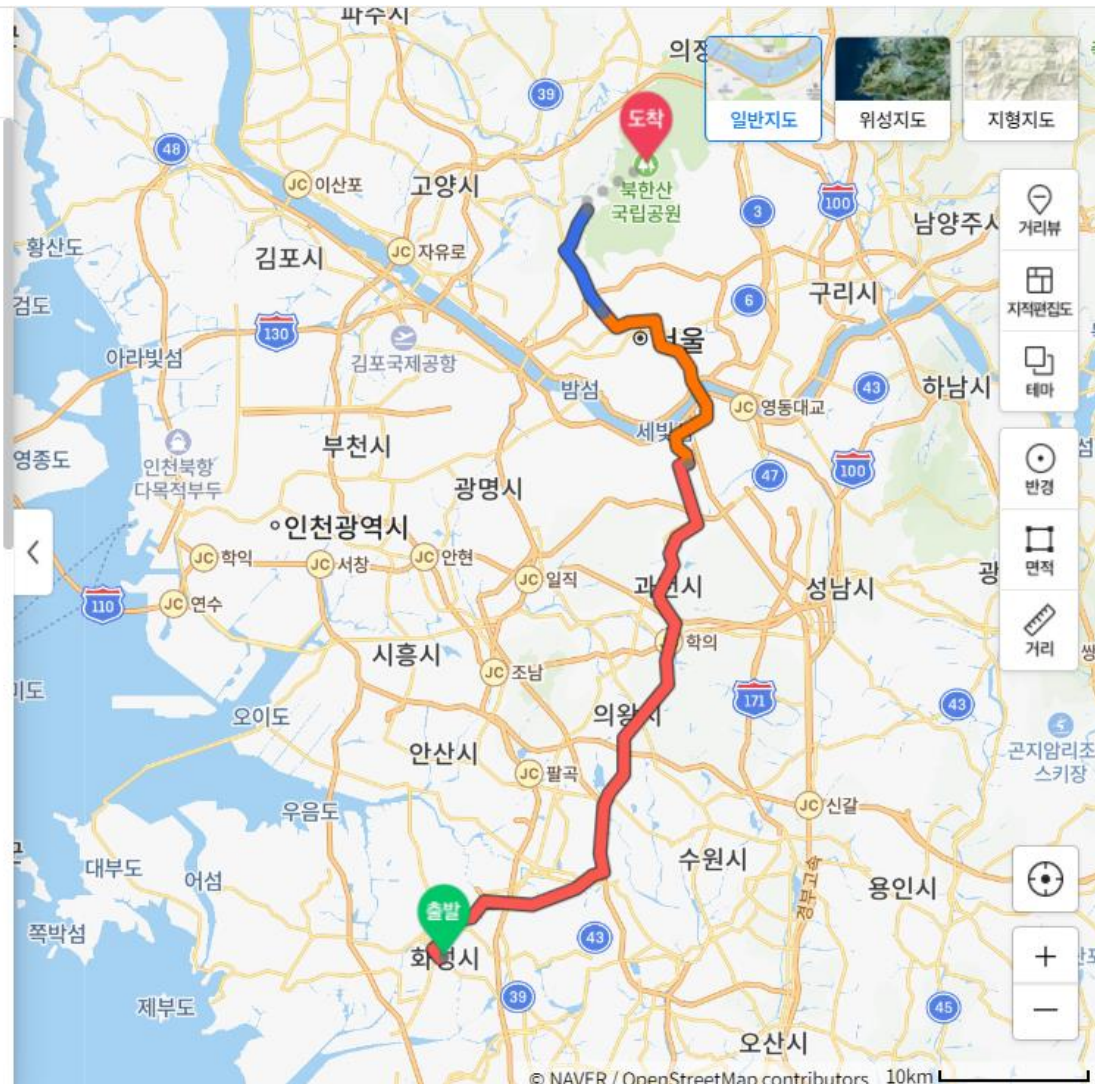
운행정보

고속터미널역 방면(대화행)

2시간 8분

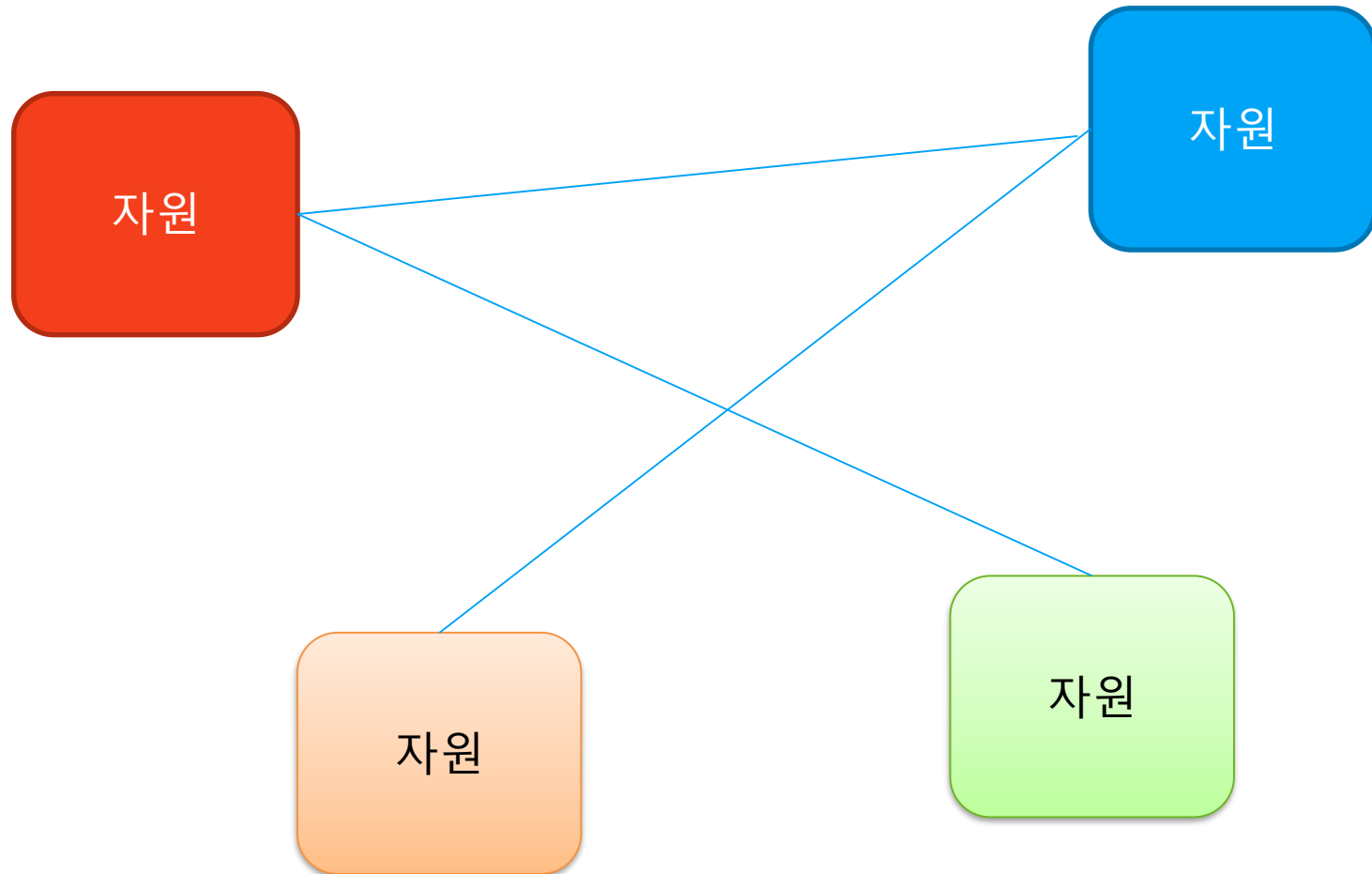
오후 8:26-오후 10:34 | 21분 | 3,700원

G1003 → 3호선 → 701



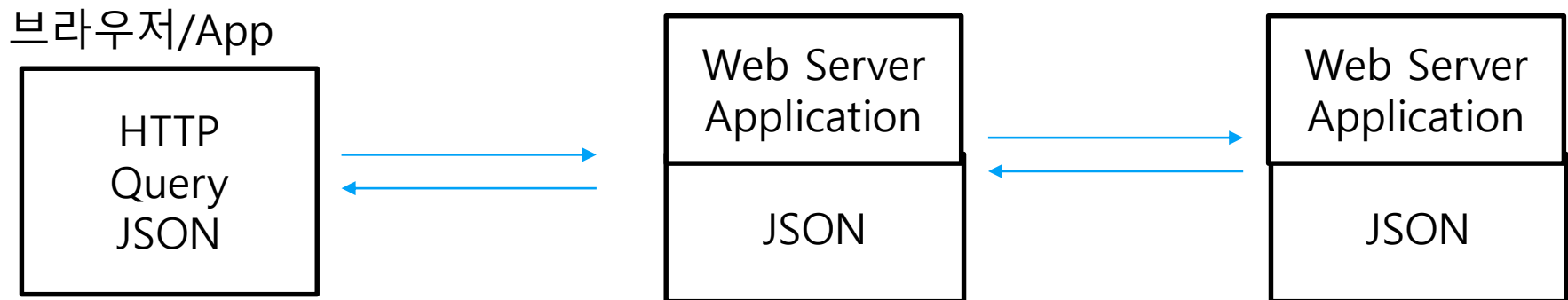
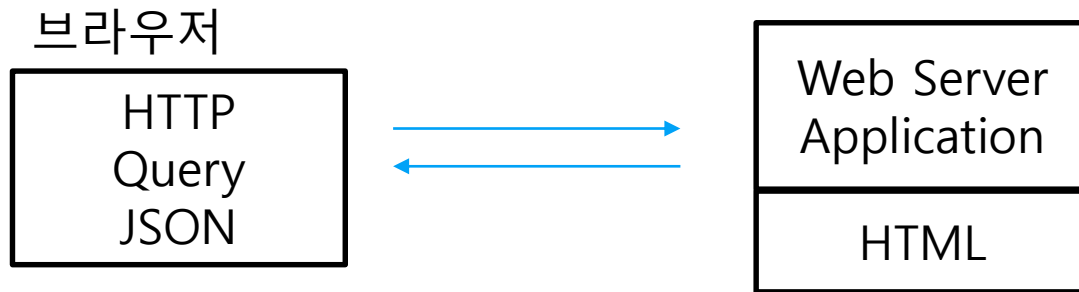
다양한 이종 시스템(플랫폼)에서 리소스 공유 문제

- 주소 시스템 + 도로 : INTERNET
- 운송 방법 : 웹 (HTTP)
- 문서시스템 : HTML, XML, JSON



Web 응용어플리케이션간 데이터 교환 방식

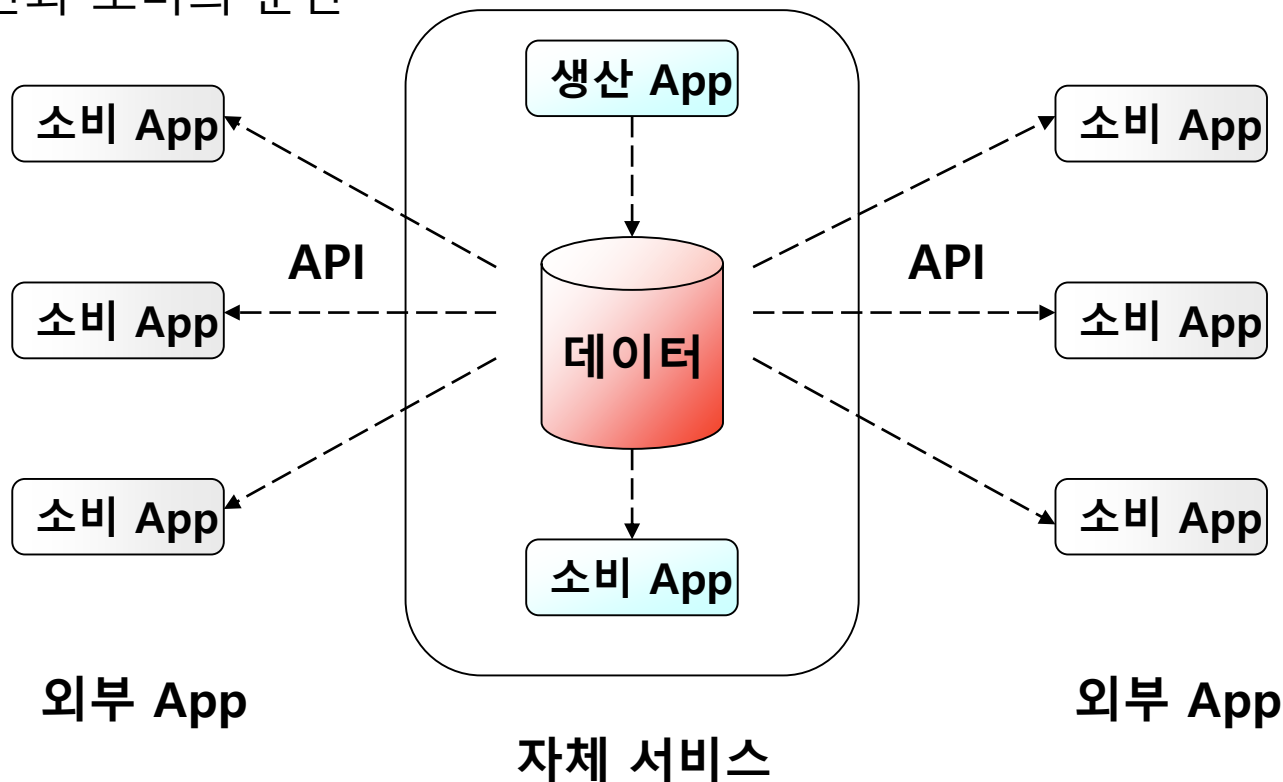
- 일반적인 웹 응용구조 : 사람이 해석할 수 있는 HTML이 생성됨
- 문제점 : 다양한 디자인 요소가 포함, 광고, 배너 포함, 빈번한 구조 변경 , 어플리케이션간 데이터 해석이 어려움



JSON 문서를 사용해 서로 통신 수행(공통문서언어)

Open API란

데이터 생산과 소비의 분산



Open API는 서비스, 정보, 데이터 등 언제, 어디서나 누구나 쉽게 이용할 수 있도록 개방된 API를 의미한다. 또한, 통신망의 구조 및 기술에 독립적으로 새로운 응용 서비스를 쉽게 개발할 수 있도록 한다. Open API는 데이터를 제어할 수 있는 간단하고 직관적인 인터페이스의 제공을 통해 사용자의 참여를 유도하는 사용자 중심의 비즈니스 모델이다.

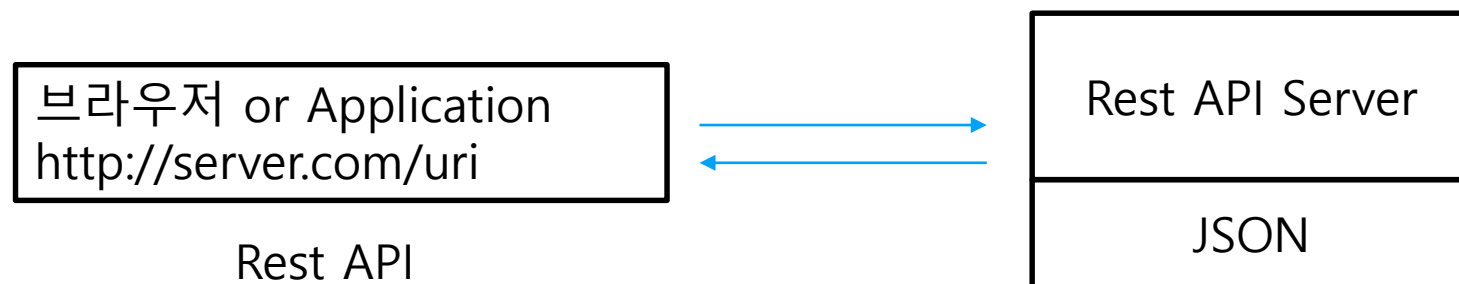
→ Open API의 장점이자 목적

웹 서비스 연동 방식의 장점

- 방화벽 문제 회피
 - Open API 웹 서비스를 제외한 다른 연동 프로토콜은 대부분 방화벽 문제를 야기(대부분의 서버들이 웹 포트 방화벽은 열어둠)
- 웹 서버로 해결 가능
 - 웹 서버 하나만 있으면 서비스가 가능해 복잡한 환경 구성이 필요 없음(대부분의 서버에는 웹 서버가 구축됨)
- HTTP 프로토콜 이용
 - 클라이언트 서버처럼 Tight Couple된 관계가 아니라서, HTTP에서는 콘텐츠 협상을 통해, 헤더, Mime Type 등의 HTTP 프로토콜의 특성을 그대로 이용할 수 있음

Rest(Representational State Transfer) API Service

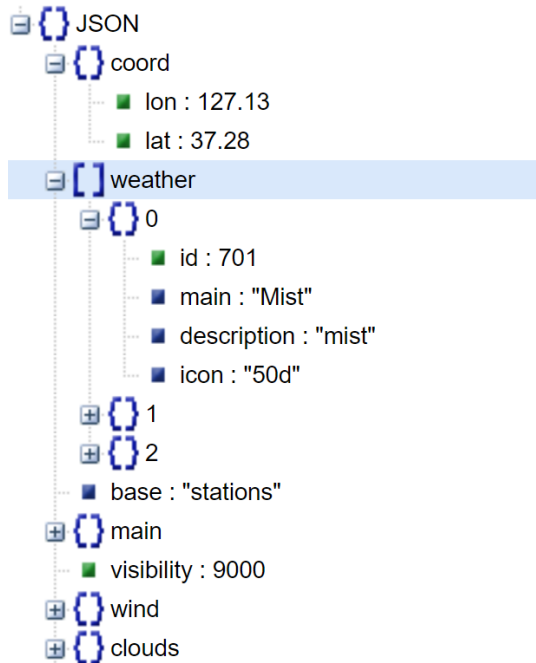
- 2000년, 로이 필딩 (Roy Fielding)의 박사학위 논문에서 최초로 소개
- 웹의 장점을 최대한 활용할 수 있는 아키텍처 디자인
- 인터넷 서비스 업체들이 응용 개발자들에게 손쉬운 데이터 제공을 목적으로 출발
- URI는 정보의 자원을 표현(리소스명은 동사보다는 명사 사용)
- 자원에 대한 행위는 HTTP Method(GET, POST, PUT, DELETE)로 표현
- 회원 id가 5인 사람 지우는 URI
 - GET /members/delete/5 -> DELETE /members/5
- 회원 id가 5인 사람 정보를 가지고 오는 URI
 - GET /members/display/1 -> GET /members/1
- 회원 추가하는 URI
 - GET /members/insert/1 -> POST /member/1
- REST API에서는 메시지 바디 내용의 포맷을 나타내기 위한 파일 확장자를 URI 안에 포함시키지 않음
 - GET /member/photo/1.jpg (X) -> GET /member/photo/1



사례 : 날씨 OpenAPI

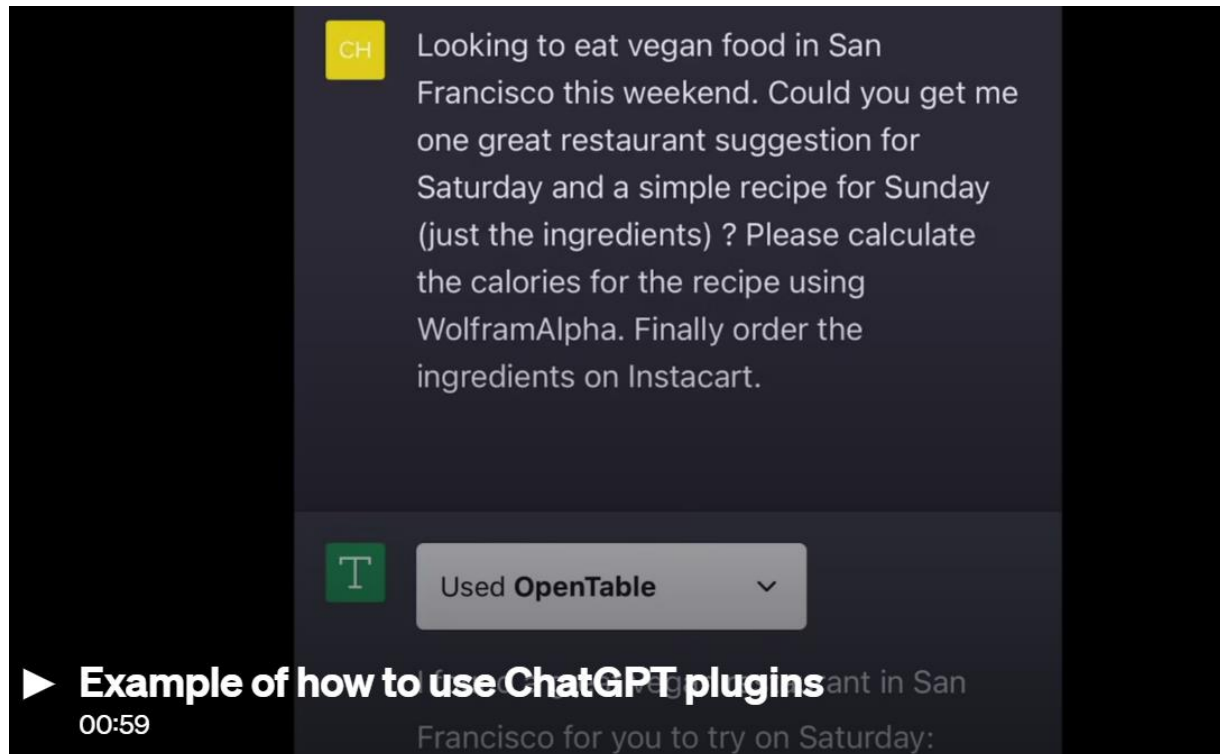
- 위도,경도 좌표에 대한 날씨 정보 제공
- JSON 포맷 사용

<http://api.openweathermap.org/data/2.5/weather?lat=37.276101&lon=127.130824&APPID=342bd9672f19bbc63b63ec3b629cb610>



ChatGPT Plugin

- ChatGPT Plugin 시나리오
- 주말에 비건채식을 찾고 있어
- 토요일에 갈 수 있는 좋은 레스토랑 알려주고
- 일요일을 위한 간단한 레시피와 재료를 알려주고
- WolframAlpha를 사용해 칼로리를 계산해주고
- Instacart에서 재료를 주문해줘



<https://openai.com/blog/chatgpt-plugins>



```
import requests
import json

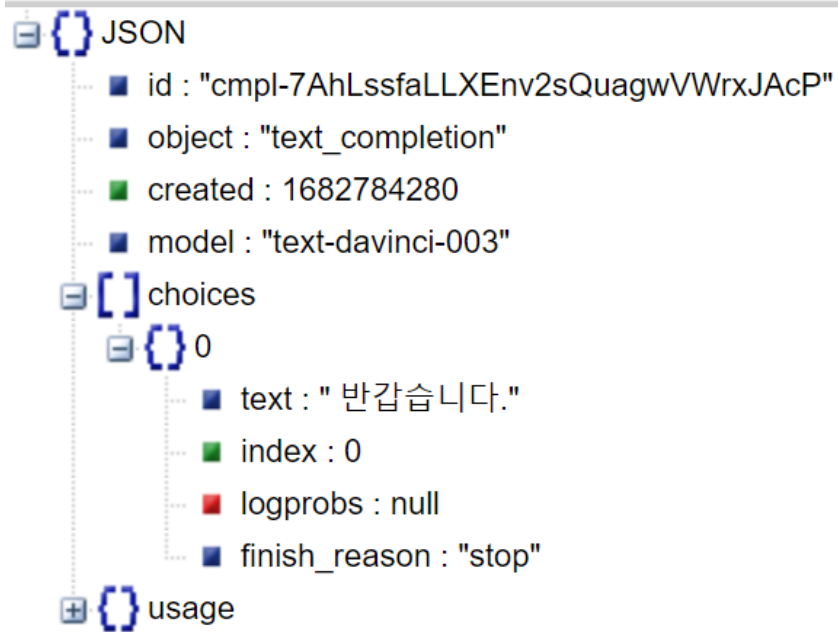
api_key = "sk-9W"

headers = {"Authorization": f"Bearer {api_key}"}
data = {"model": "text-davinci-003", "prompt": "안녕하세요.",
        "max_tokens": 200 }

response = requests.post(
    "https://api.openai.com/v1/completions",
    headers=headers, json=data,
)

res = response.json()["choices"][0]["text"]
print(res)
```

OpenAI - Chat API





```
import requests

api_key = "sk-"

while True :
    prompt = input()
    if prompt == "." : break
    headers = {"Authorization": f"Bearer {api_key}"}
    data = {"model": "text-davinci-003", "prompt": prompt,
           "max_tokens": 200 }

    response = requests.post(
        "https://api.openai.com/v1/completions",
        headers = headers, json = data,
    )

    res = response.json()["choices"][0]["text"]
    print("User :" + prompt)
    print("GPT :" + res)
```

Dictionary 개요

- Dictionary 정의 : 중괄호를 사용해 name(key)/value 정의
- 키는 문자열로 정의하며 반드시 "(더블 쿼테이션)을 사용해야 함
- ,를 사용해 여러 개의 name(key)/value를 정의
- obj = {"name": "이순신", "age":56, "check": True}
- 대괄호를 사용해 키값 조회
- obj["name"] // 키가 존재하지 않을 때 예외 발생
- obj.get("name") // 키가 존재하지 않으면 None 리턴
- obj.get("name", "기본값") // 키가 존재하지 않으면 "기본값" 리턴
- "name" in obj // obj에 "name" 키가 있는지 조사
- 대괄호를 사용해 키값 설정
- obj["name"] = "홍길동"
- JSON과 비교
 - name정의 시 " 사용하지 않아도 됨, Boolean 타입다름, 속성방식(.)으로 값을 읽고 쓰기 가능
 - JSON 객체를 JSON 스트링으로 표 변환
 - {"name":"임꺽정","age":50,"check":true}

```
obj = {name:'이순신', age:50, check:true}
obj.name = "홍길동"
obj["name"] = "임꺽정"
console.log(obj)                                     javascript
```

Dictionary : 요소 접근



dict1.py

```
obj = {"name": "이순신", "age": 56, "check": True}
print(obj)
```

```
key = "이름"
obj = {key: "이순신"}      #key가 변수이기 때문에 "이름": "이순신 "
print(obj)
```

```
#print(obj[ " age " ])    # 키가 없으면 예외발생
print(obj.get("age"))     # 키가 없으면 None 리턴
print(obj.get( " age " , 0)) # 키가 없으면 0 리턴
print( "이름" in obj )    # 이름이라는 키가 있는지 조회
```

```
obj["score"] = 100
obj[key] = "홍길동"
print(obj)
```

```
{'name': '이순신', 'age': 56, 'check': True}
{'이름': '이순신'}
None
0
True
{'이름': '홍길동', 'score': 100}
```

Dictionary : 데이터표현

- table형태의 데이터는 Dictionary나 Dictionary의 리스트 형태로 표현할 수 있음
- Dictionary의 리스트 형태를 일반적으로 사용함

name	age	check
이순신	56	True
홍길동	45	False
임꺽정	23	True

- Dictionary만 사용해 정의

```
datas = { "이순신" : {"age":56, "check": True},  
          "홍길동" : {"age":45, "check": False},  
          "임꺽정" : {"age":23, "check": True}}  
print(datas["홍길동"]["age"]) # 홍길동으로 검색 가능
```

dict3.py

- Dictionary의 리스트 사용해 정의

```
datas = [{"name": "이순신", "age":56, "check": True},  
         {"name": "홍길동", "age":45, "check": False},  
         {"name": "임꺽정", "age":23, "check": True}]  
print(datas[0]["name"], datas[0]["age"]) # index 가지고 검색 가능
```


Dictionary : Save/Load



- pickle : 파이썬 객체 자체를 파일로 저장하는 라이브러리
- 거대 용량도 효과적으로 저장, 적재할 수 있음
- 모든 타입에 대해서 적용가능
- binary파일이기 때문에 한글과 상관없음

dict9.py

```
import pickle
```

```
obj = {"name": "이순신", "age":56, "check": True}
```

```
with open("data.pickle","wb") as f:  
    pickle.dump(obj, f)
```

```
with open("data.pickle","rb") as f:  
    obj2 = pickle.load(f)
```

```
print(obj2)
```