

# 웹 상의 데이터

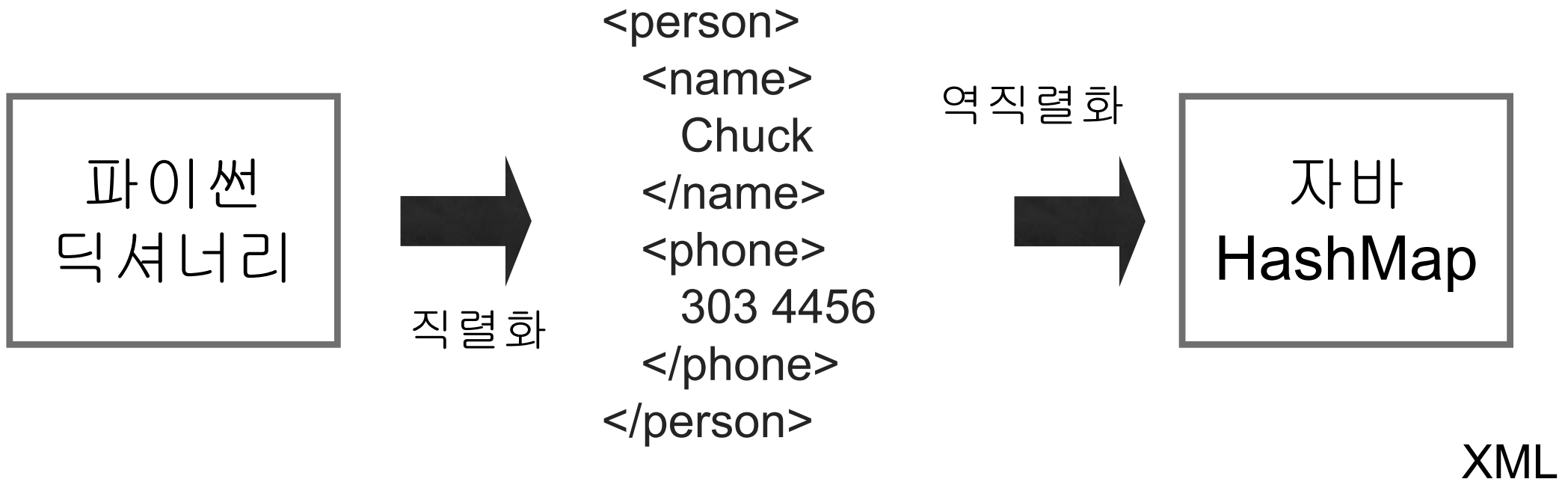
- HTTP 요청과 응답에 대한 이해와 지원을 바탕으로, 이 프로토콜을 이용한 프로그램 간의 정보 교환의 추세
- 네트워크와 응용프로그램 간의 데이터 표현 방식에 있어서 합의가 필요
- 가장 널리 사용되는 두 가지 포맷: XML과 JSON

# 네트워크를 통한 정보 전송

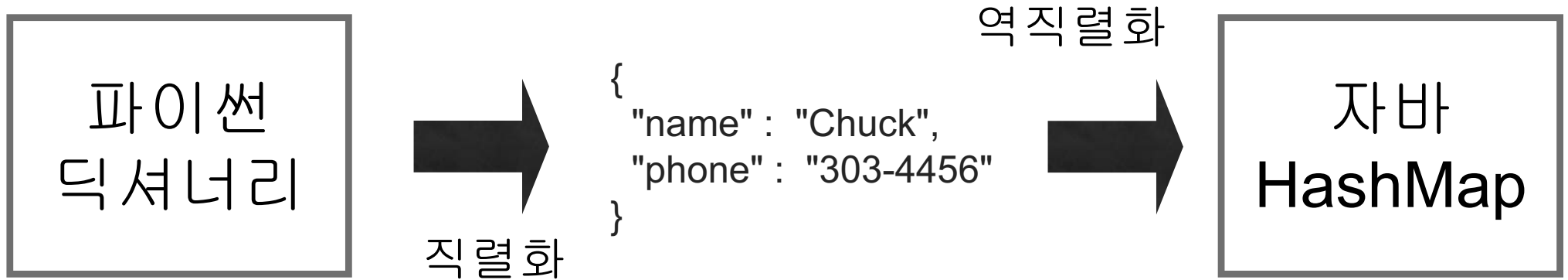


a.k.a. “와이어 프로토콜” - 우리가 “와이어” 상에 보내는 것

# “와이어 포맷” 합의하기



# “와이어 포맷” 합의하기



JSON

# eXtensible Markup Language

- 정보 시스템이 구조화된 데이터를 공유하는 것이 초기 목적
- 표준 범용 교정 용어 (**SGML**) 의 간소화된 버전으로 시작하였고, 조금 더 인간에게 친숙한 방향으로 디자인

<http://en.wikipedia.org/wiki/XML>

# XML의 기초

- 시작 태그 start tag

- 끝 태그 end tag

- 문자 정보 text element

- 속성 attributes

- 스스로 닫는 태그 self closing tag

```
<person>
  <name>Chuck</name>
  <phone type="intl">
    +1 734 303 4456
  </phone>
  <email hide="yes" />
</person>
```

# 공백

```
<person>
  <name>Chuck</name>
  <phone type="intl">
    +1 734 303 4456
  </phone>
  <email hide="yes" />
</person>
```

줄의 끝은 중요하지 않음.  
문자 요소에서 공백은 없어짐.  
오직 가독성만을 위해  
들여쓰기를 함.

```
<person>
  <name>Chuck</name>
  <phone type="intl">+1 734 303 4456</phone>
  <email hide="yes" />
</person>
```

# XML 용어

- 태그 **Tags** 는 요소의 시작과 끝을 알려줌
- 속성 **Attributes** - **XML**의 여는 태그에 위치한 키-값 쌍
- 직렬화 **Serialize** / 역직렬화 **De-Serialize** - 한 프로그램의 데이터를 특정 프로그램 언어에 제한되지 않은 채로 시스템 내에서 저장되고 전달되어질 수 있는 형식으로 변환하는 것

<http://en.wikipedia.org/wiki/Serialization>

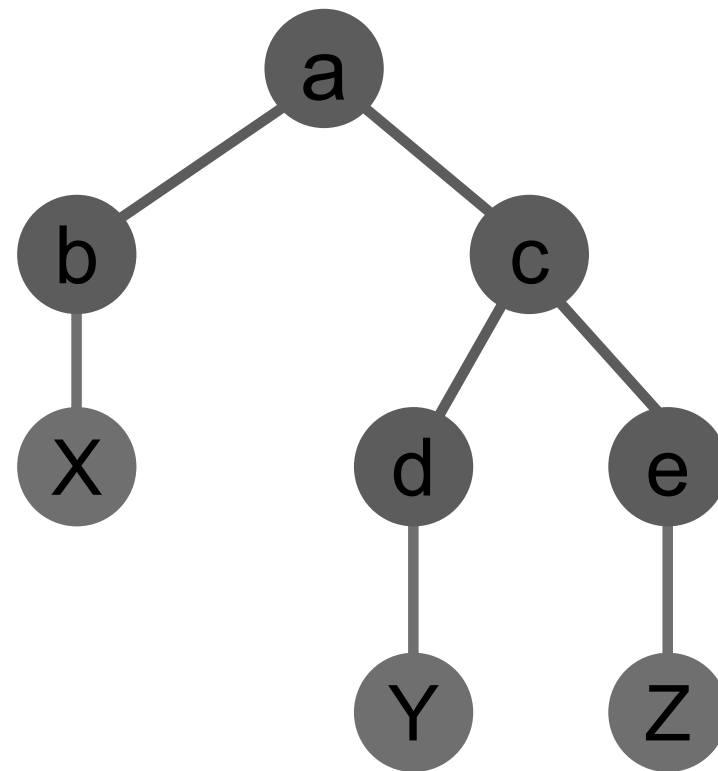


# XML 트리

```
<a>  
  <b>X</b>  
  <c>  
    <d>Y</d>  
    <e>Z</e>  
  </c>  
</a>
```

요소

문자

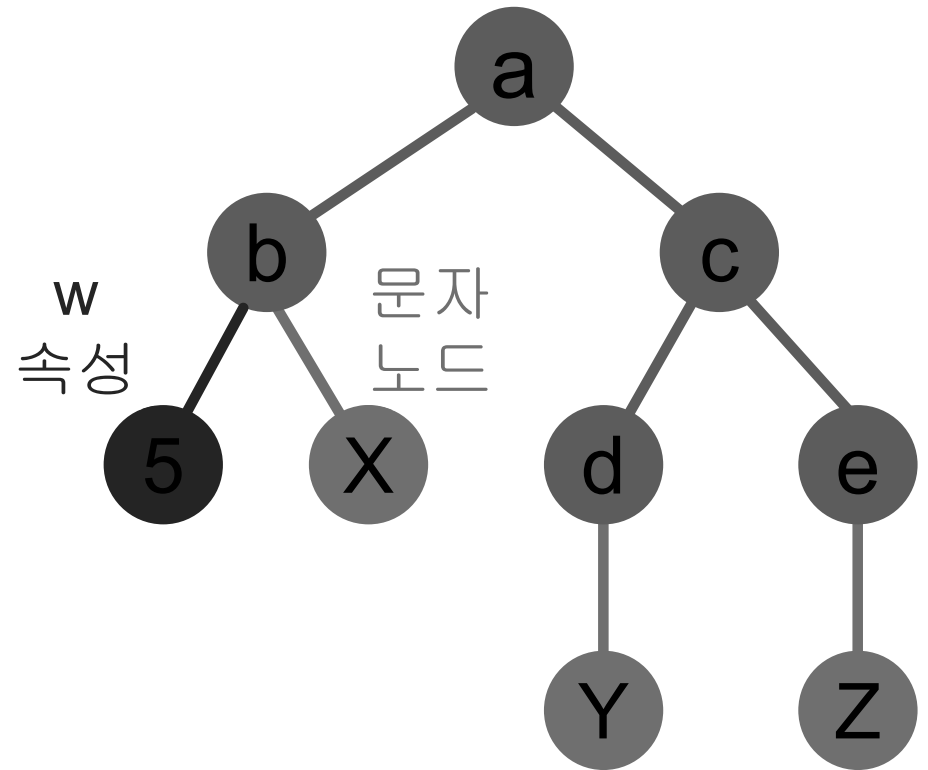


# XML 문자와 속성

```
<a>  
  <b w="5">X</b>  
  <c>  
    <d>Y</d>  
    <e>Z</e>  
  </c>  
</a>
```

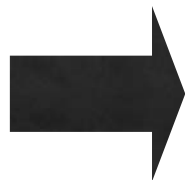
요소

문자



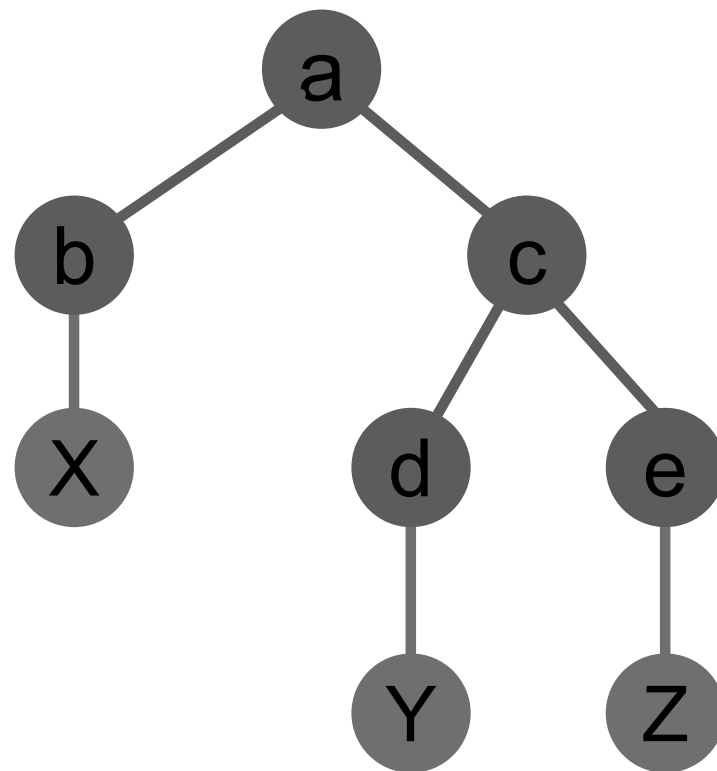
# XML 경로

```
<a>  
  <b>X</b>  
  <c>  
    <d>Y</d>  
    <e>Z</e>  
  </c>  
</a>
```



/a/b	X
/a/c/d	Y
/a/c/e	Z

Elements    Text



# XML 스키마

XML이 받아들이는 형태의 “약속” 을 설명

[http://en.wikipedia.org/wiki/XML\\_schema](http://en.wikipedia.org/wiki/XML_schema)

[http://en.wikibooks.org/wiki/XML\\_Schema](http://en.wikibooks.org/wiki/XML_Schema)

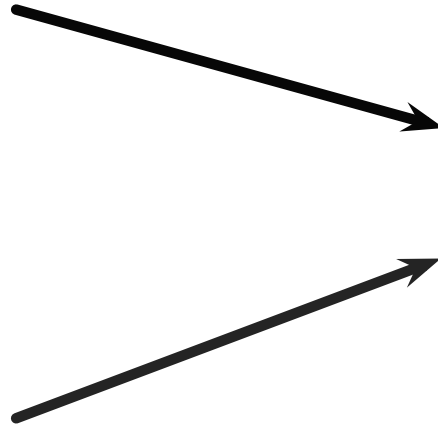
# XML 스키마

- XML 문서의 올바른 형식에 대한 설명
- 문서의 구조와 내용에 대한 제한의 형식으로 표현됨
- 시스템 간의 “약속”을 표현할 때 주로 사용됨 - “내 시스템은 이 스키마에 맞는 XML만 수용할 거야.”
- 특정 XML이 스키마의 사항들을 만족할 때 우리는 그것을 “타당하다 (validate)” 라고 한다

XML 검증

XML 문서

XML 스키마  
계약서



## XML 문서

```
<person>
  <lastname>Severance</lastname>
  <age>17</age>
  <dateborn>2001-04-17</dateborn>
</person>
```

## XML 스키마 계약서

```
<xs:complexType name="person">
  <xs:sequence>
    <xs:element name="lastname" type="xs:string"/>
    <xs:element name="age" type="xs:integer"/>
    <xs:element name="dateborn" type="xs:date"/>
  </xs:sequence>
</xs:complexType>
```

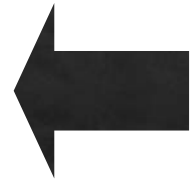
## XML 검증



검증기

# 여러 XML 스키마 언어

- 문서 타입 정의 Document Type Definition (DTD)
  - [http://en.wikipedia.org/wiki/Document\\_Type\\_Definition](http://en.wikipedia.org/wiki/Document_Type_Definition)
- 표준 범용 교정 용어 (ISO 8879:1986 SGML)
  - <http://en.wikipedia.org/wiki/SGML>
- XML 스키마 W3C - (XSD)
  - [http://en.wikipedia.org/wiki/XML\\_Schema\\_\(W3C\)](http://en.wikipedia.org/wiki/XML_Schema_(W3C))  
[http://en.wikipedia.org/wiki/Xml\\_schema](http://en.wikipedia.org/wiki/Xml_schema)





# XSD XML 스키마 (W3C spec)

- 우리는 World Wide Web Consortium (W3C) 버전에 집중할 것
- 때로 “W3C 스키마”라고 불리는 이유는 “스키마”가 포괄적인 표현이기 때문
- 흔히 XSD 라 불리는 이유는 파일 확장명이 .xsd 이기 때문

<http://www.w3.org/XML/Schema>

[http://en.wikipedia.org/wiki/XML\\_Schema\\_\(W3C\)](http://en.wikipedia.org/wiki/XML_Schema_(W3C))

# XSD

## 구조

- xs:element
- xs:sequence
- xs:complexType

```
<person>  
  <lastname>Severance</lastname>  
  <age>17</age>  
  <dateborn>2001-04-17</dateborn>  
</person>
```

```
<xs:complexType name="person">  
  <xs:sequence>  
    <xs:element name="lastname" type="xs:string"/>  
    <xs:element name="age" type="xs:integer"/>  
    <xs:element name="dateborn" type="xs:date"/>  
  </xs:sequence>  
</xs:complexType>
```

# XSD

## 제한

```
<xs:element name="person">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="full_name" type="xs:string"
        minOccurs="1" maxOccurs="1" />
      <xs:element name="child_name" type="xs:string"
        minOccurs="0" maxOccurs="10" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<person>
  <full_name>Tove Refsnes</full_name>
  <child_name>Hege</child_name>
  <child_name>Stale</child_name>
  <child_name>Jim</child_name>
  <child_name>Borge</child_name>
</person>
```

# XSD 데이터 타입

```
<xs:element name="customer" type="xs:string"/>  
<xs:element name="start" type="xs:date"/>  
<xs:element name="startdate" type="xs:dateTime"/>  
<xs:element name="prize" type="xs:decimal"/>  
<xs:element name="weeks" type="xs:integer"/>
```

서버가 세계에 퍼져있다는  
가정 하에, 흔히 시간을  
UTC/GMT 로 표현한다

```
<customer>John Smith</customer>  
<start>2002-09-24</start>  
<startdate>2002-05-30T09:30:10Z</startdate>  
<prize>999.50</prize>  
<weeks>30</weeks>
```

# ISO 8601 날짜/시간 형식

2002-05-30T09:30:10Z

↑  
년-월-일

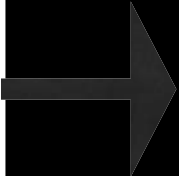
↑  
시간

↑  
시간대 - 주로 현지  
시간대가 아닌 UTC /  
GMT로 명시

[http://en.wikipedia.org/wiki/ISO\\_8601](http://en.wikipedia.org/wiki/ISO_8601)

[http://en.wikipedia.org/wiki/Coordinated\\_Universal\\_Time](http://en.wikipedia.org/wiki/Coordinated_Universal_Time)

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema elementFormDefault="qualified" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Address">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Recipient" type="xs:string" />
        <xs:element name="House" type="xs:string" />
        <xs:element name="Street" type="xs:string" />
        <xs:element name="Town" type="xs:string" />
        <xs:element minOccurs="0" name="County" type="xs:string" />
        <xs:element name="PostCode" type="xs:string" />
        <xs:element name="Country">
          <xs:simpleType>
            <xs:restriction base="xs:string">
              <xs:enumeration value="FR" />
              <xs:enumeration value="DE" />
              <xs:enumeration value="ES" />
              <xs:enumeration value="UK" />
              <xs:enumeration value="US" />
            </xs:restriction>
          </xs:simpleType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```



```
<?xml version="1.0" encoding="utf-8"?>
<Address
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="SimpleAddress.xsd">
  <Recipient>Mr. Walter C. Brown</Recipient>
  <House>49</House>
  <Street>Featherstone Street</Street>
  <Town>LONDON</Town>
  <PostCode>EC1Y 8SY</PostCode>
  <Country>UK</Country>
</Address>
```

xml1.py

```
import xml.etree.ElementTree as ET
data = '''<person>
  <name>Chuck</name>
  <phone type="intl">
    +1 734 303 4456
  </phone>
  <email hide="yes"/>
</person>'''

tree = ET.fromstring(data)
print('Name:', tree.find('name').text)
print('Attr:', tree.find('email').get('hide'))
```

xml2.py

```
import xml.etree.ElementTree as ET
input = '''<stuff>
  <users>
    <user x="2">
      <id>001</id>
      <name>Chuck</name>
    </user>
    <user x="7">
      <id>009</id>
      <name>Brent</name>
    </user>
  </users>
</stuff>'''

stuff = ET.fromstring(input)
lst = stuff.findall('users/user')
print('User count:', len(lst))
for item in lst:
    print('Name', item.find('name').text)
    print('Id', item.find('id').text)
    print('Attribute', item.get("x"))
```



# JavaScript Object Notation

json1.py

```
import json
data = '''{
    "name" : "Chuck",
    "phone" : {
        "type" : "intl",
        "number" : "+1 734 303 4456"
    },
    "email" : {
        "hide" : "yes"
    }
}'''

info = json.loads(data)
print('Name:', info["name"])
print('Hide:', info["email"]["hide"])
```

**JSON** 은 데이터를  
중첩된 “리스트”와  
“딕셔너리” 로 표현

json2.py

```
import json
input = '''[
    { "id" : "001",
      "x" : "2",
      "name" : "Chuck"
    } ,
    { "id" : "009",
      "x" : "7",
      "name" : "Chuck"
    }
]'''
```

```
info = json.loads(input)
print('User count:', len(info))
for item in info:
    print('Name', item['name'])
    print('Id', item['id'])
    print('Attribute', item['x'])
```

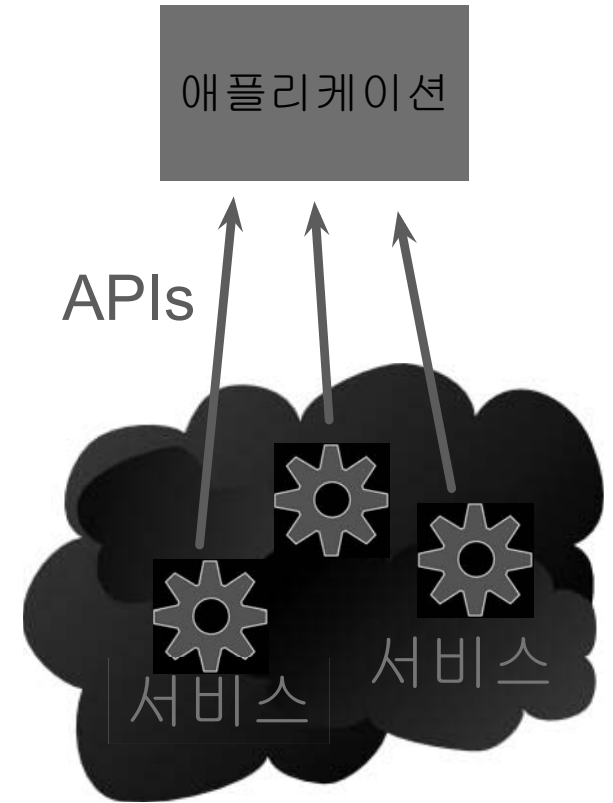
JSON 은 데이터를  
중첩된 “리스트”와  
“딕셔너리” 로 표현

# 서비스 지향적 접근

[http://en.wikipedia.org/wiki/Service-oriented\\_architecture](http://en.wikipedia.org/wiki/Service-oriented_architecture)

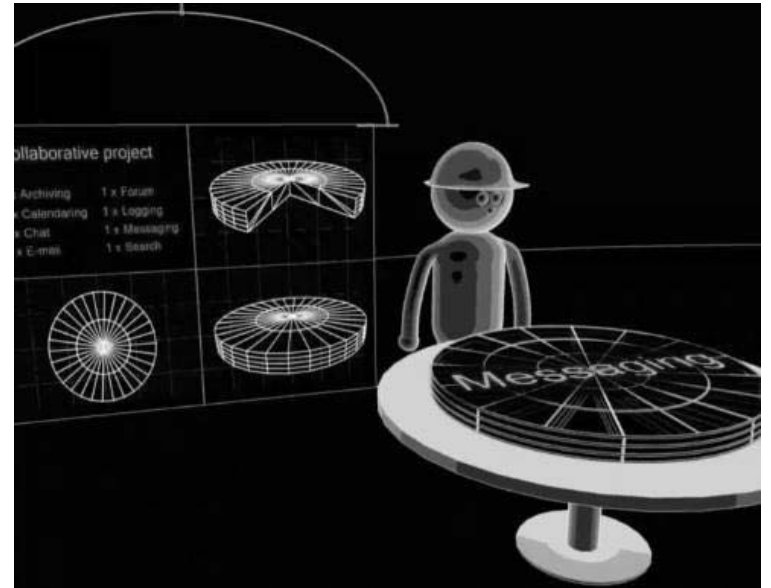
# 서비스 지향적 접근

- 대부분의 대형 웹 애플리케이션은 서비스를 이용
- 다른 애플리케이션으로부터 서비스를 사용
  - 신용카드 청구
  - 호텔 예약 시스템
- 서비스는 애플리케이션이 서비스를 이용하기 위해 따라야하는 “규칙”을 만듦 (API)



# 다수의 시스템

- 초기에는 두 시스템이 협력하여 문제를 나눔
- 데이터와 서비스가 유용해지며 다수의 애플리케이션이 정보를 이용하려 함



<http://www.youtube.com/watch?v=mj-kCFzF0ME>

5:15

# 응용 프로그램 인터페이스(API)

**API**는 인터페이스를 지정하고 그 인터페이스의 객체의 행동을 제어한다는 점에서 매우 추상적. **API**에 명시된 기능을 제공하는 소프트웨어를 **API**의 “실행”이라고 하며, **API**는 대체로 애플리케이션을 구성하게 되는 언어로 정의됨.

<http://en.wikipedia.org/wiki/API>

The screenshot shows the Google Maps APIs documentation page for the Geocoding API. The browser's address bar displays the URL <https://developers.google.com/maps/documentation/geocoding/start>. The page header includes the Google Maps APIs logo, navigation links for Home, Documentation, and Pricing and Plans, a search bar, and links for All Products and a user profile. Below the header, the breadcrumb trail reads 'Web Services > Geocoding API', followed by buttons for 'GET A KEY' and 'VIEW PRICING AND PLANS'. A secondary navigation bar contains links for GUIDES, SUPPORT, and a 'SEND FEEDBACK' button. The main content area is titled 'Getting Started' with a five-star rating. It explains that the Google Maps Geocoding API provides geocoding and reverse geocoding services. A callout box notes that the service is also available as part of the client-side Google Maps JavaScript API or for server-side use with the Java Client, Python Client, Go Client, and Node.js Client. The page defines geocoding as converting addresses into geographic coordinates and reverse geocoding as converting coordinates into human-readable addresses. A 'Sample request and response' section introduces the HTTP interface. A 'Contents' sidebar on the right lists topics such as 'Sample request and response', 'Geocoding request and response (latitude/longitude lookup)', 'Reverse geocoding request and response (address lookup)', 'Start coding with our client libraries', 'Authentication, quotas, and policies', 'Activate the API and get an API key', 'Quotas', 'Policies', and 'Learn more'. A left sidebar provides a comprehensive list of links, including 'Get Started', 'Developer's Guide', 'Best Practices', 'Geocoder FAQ', 'Get a Key', 'Usage Limits', 'Optimizing Quota Usage', 'Policies', 'Terms of Service', 'Google Maps Web Services', 'Introduction', 'Client Library', 'Other APIs', 'Directions API', 'Distance Matrix API', 'Elevation API', 'Geolocation API', 'Places API Web Service', 'Roads API', and 'Time Zone API'.

Getting Started

The Google Maps Geocoding API is a service that provides geocoding and reverse geocoding of addresses.

★ This service is also available as part of the client-side [Google Maps JavaScript API](#), or for server-side use with the [Java Client](#), [Python Client](#), [Go Client](#) and [Node.js Client](#) for Google Maps Services.

**Geocoding** is the process of converting addresses (like a street address) into geographic coordinates (like latitude and longitude), which you can use to place markers on a map, or position the map.

**Reverse geocoding** is the process of converting geographic coordinates into a human-readable address. The Google Maps Geocoding API's reverse geocoding service also lets you find the address for a given place ID.

Sample request and response

You access the Google Maps Geocoding API through an HTTP interface. Following are examples of geocoding and reverse geocoding requests.

Geocoding request and response (latitude/longitude lookup)

The following example requests the latitude and longitude of "1600 Amphitheatre Parkway, Mountain View, CA", and specifies that the output must be in JSON format.

Contents

- Sample request and response
- Geocoding request and response (latitude/longitude lookup)
- Reverse geocoding request and response (address lookup)
- Start coding with our client libraries
- Authentication, quotas, and policies
- Activate the API and get an API key
- Quotas
- Policies
- Learn more

Get Started

Developer's Guide

Best Practices

Geocoder FAQ

Get a Key

Usage Limits

Optimizing Quota Usage

Policies

Terms of Service

Google Maps Web Services

Introduction

Client Library

Other APIs

Directions API

Distance Matrix API

Elevation API

Geolocation API

Places API Web Service

Roads API

Time Zone API

<https://developers.google.com/maps/documentation/geocoding/>



```
{
  "status": "OK",
  "results": [
    {
      "geometry": {
        "location_type": "APPROXIMATE",
        "location": {
          "lat": 42.2808256,
          "lng": -83.7430378
        }
      },
      "address_components": [
        {
          "long_name": "Ann Arbor",
          "types": [
            "locality",
            "political"
          ],
          "short_name": "Ann Arbor"
        }
      ],
      "formatted_address": "Ann Arbor, MI, USA",
      "types": [
        "locality",
        "political"
      ]
    }
  ]
}
```

<http://maps.googleapis.com/maps/api/geocode/json?address=Ann+Arbor%2C+MI>

geojson.py

```

import urllib.request, urllib.parse, urllib.error
import json

serviceurl = 'http://maps.googleapis.com/maps/api/geocode/json?'

while True:
    address = input('Enter location: ')
    if len(address) < 1: break

    url = serviceurl + urllib.parse.urlencode({'address': address})

    print('Retrieving', url)
    uh = urllib.request.urlopen(url)
    data = uh.read().decode()
    print('Retrieved', len(data), 'characters')

    try:
        js = json.loads(data)
    except:
        js = None

    if not js or 'status' not in js or js['status'] != 'OK':
        print('==== Failure To Retrieve ====')
        print(data)
        continue

    lat = js["results"][0]["geometry"]["location"]["lat"]
    lng = js["results"][0]["geometry"]["location"]["lng"]
    print('lat', lat, 'lng', lng)
    location = js['results'][0]['formatted_address']
    print(location)

```

```

Enter location: Ann Arbor, MI
Retrieving http://maps.googleapis.com/...
Retrieved 1669 characters
lat 42.2808256 lng -83.7430378
Ann Arbor, MI, USA
Enter location:

```

geojson.py



# Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance ([www.dr-chuck.com](http://www.dr-chuck.com)) of the University of Michigan School of Information and [open.umich.edu](http://open.umich.edu) and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

...

Initial Development: Charles Severance, University of Michigan School of Information

Contributor:

- Seung-June Lee ([plusjune@gmail.com](mailto:plusjune@gmail.com))
- Connect Foundation

Translator:

- Jo Ha Nul
- Jeungmin Oh ([tangza@gmail.com](mailto:tangza@gmail.com))