

웹서비스 사용하기

제13장



Python for Everybody
www.py4e.com



XML

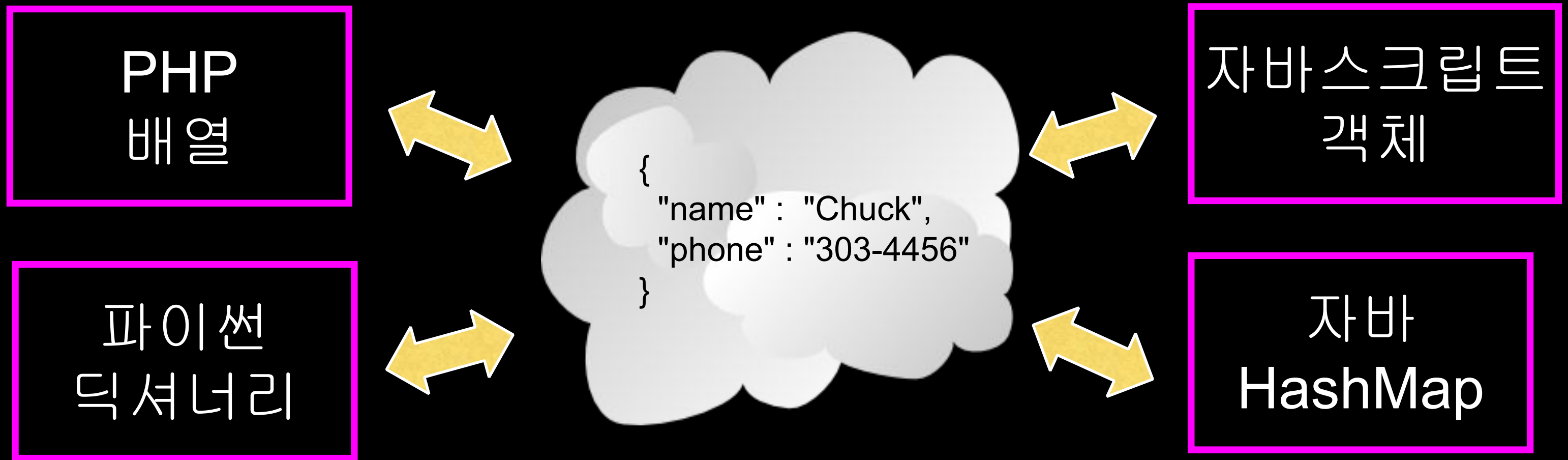
데이터를 마크업하여 네트워크 상에서 전송

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

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

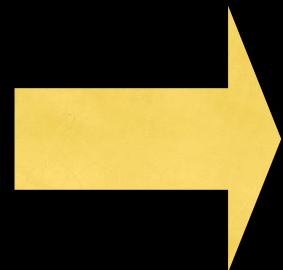
네트워크를 통한 정보 전송



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

“와이어 포맷” 합의하기

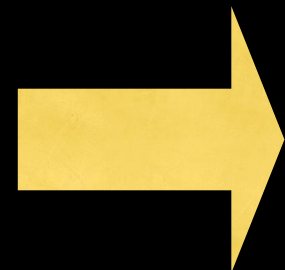
파이썬
딕셔너리



직렬화

```
<person>
  <name>
    Chuck
  </name>
  <phone>
    303 4456
  </phone>
</person>
```

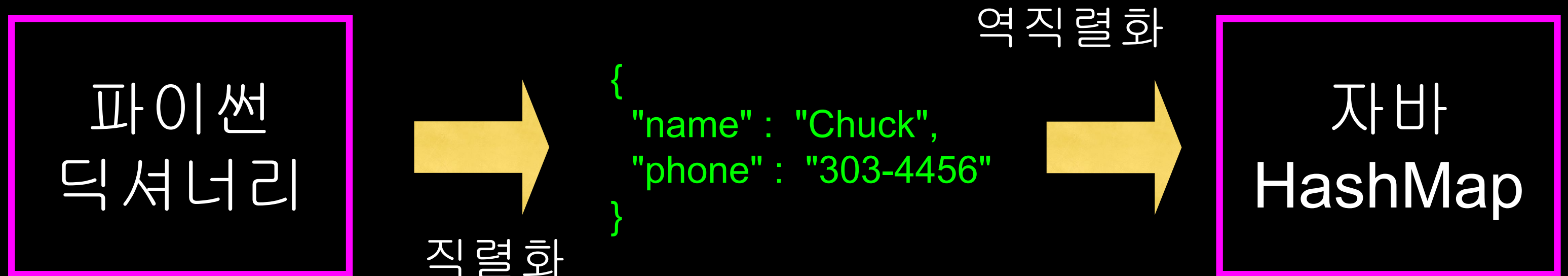
역직렬화



자바
HashMap

XML

“와이어 포맷” 합의하기



JSON

XML 요소들 (또는 노드)

- 단순 요소

```
<people>
  <person>
    <name>Chuck</name>
    <phone>303 4456</phone>
  </person>
```

- 복합 요소

```
  <person>
    <name>Noah</name>
    <phone>622 7421</phone>
  </person>
</people>
```

eXensible 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** - XML의 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

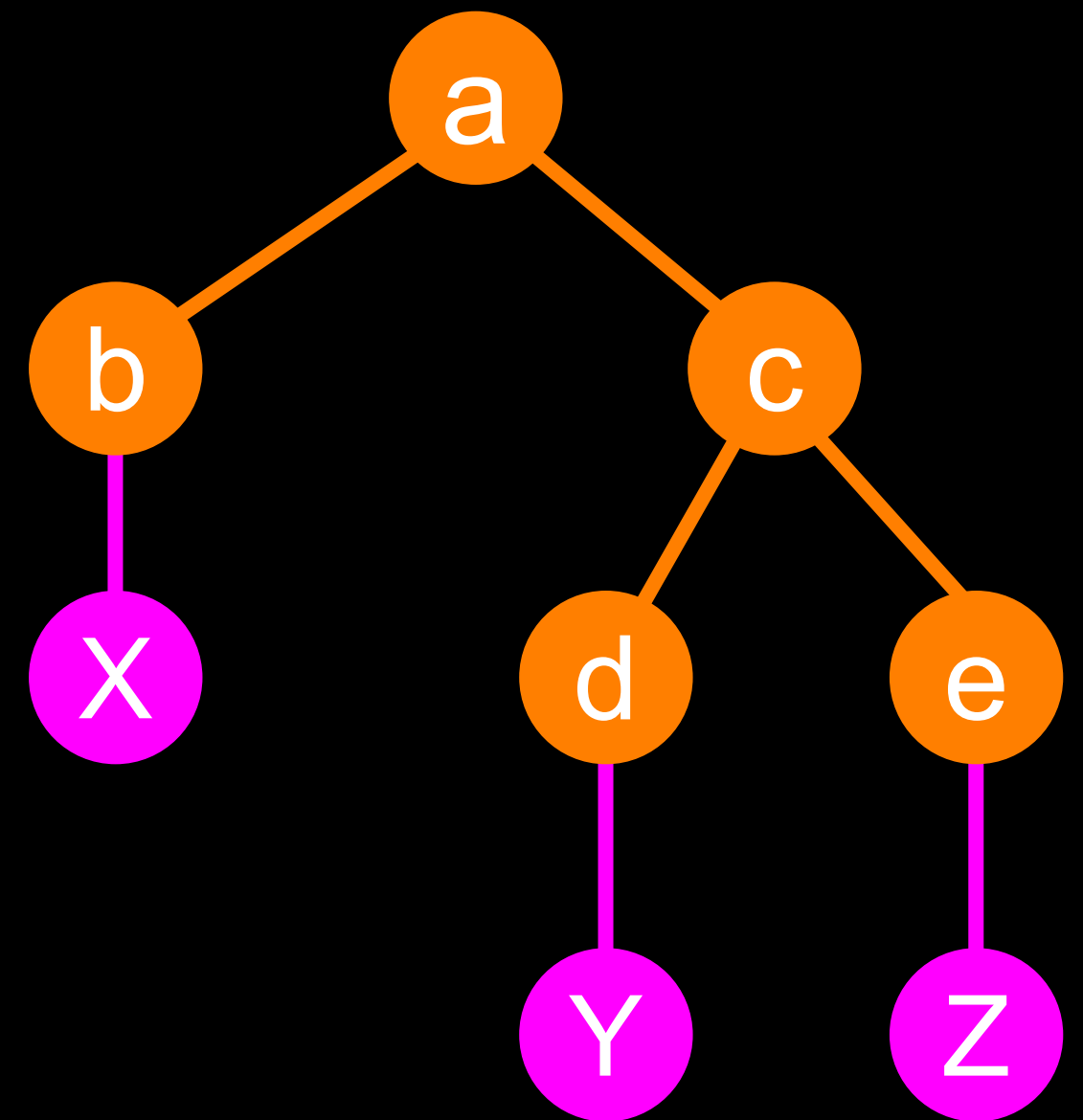
<http://en.wikipedia.org/wiki/Serialization>

XML 트리

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

요소

문자

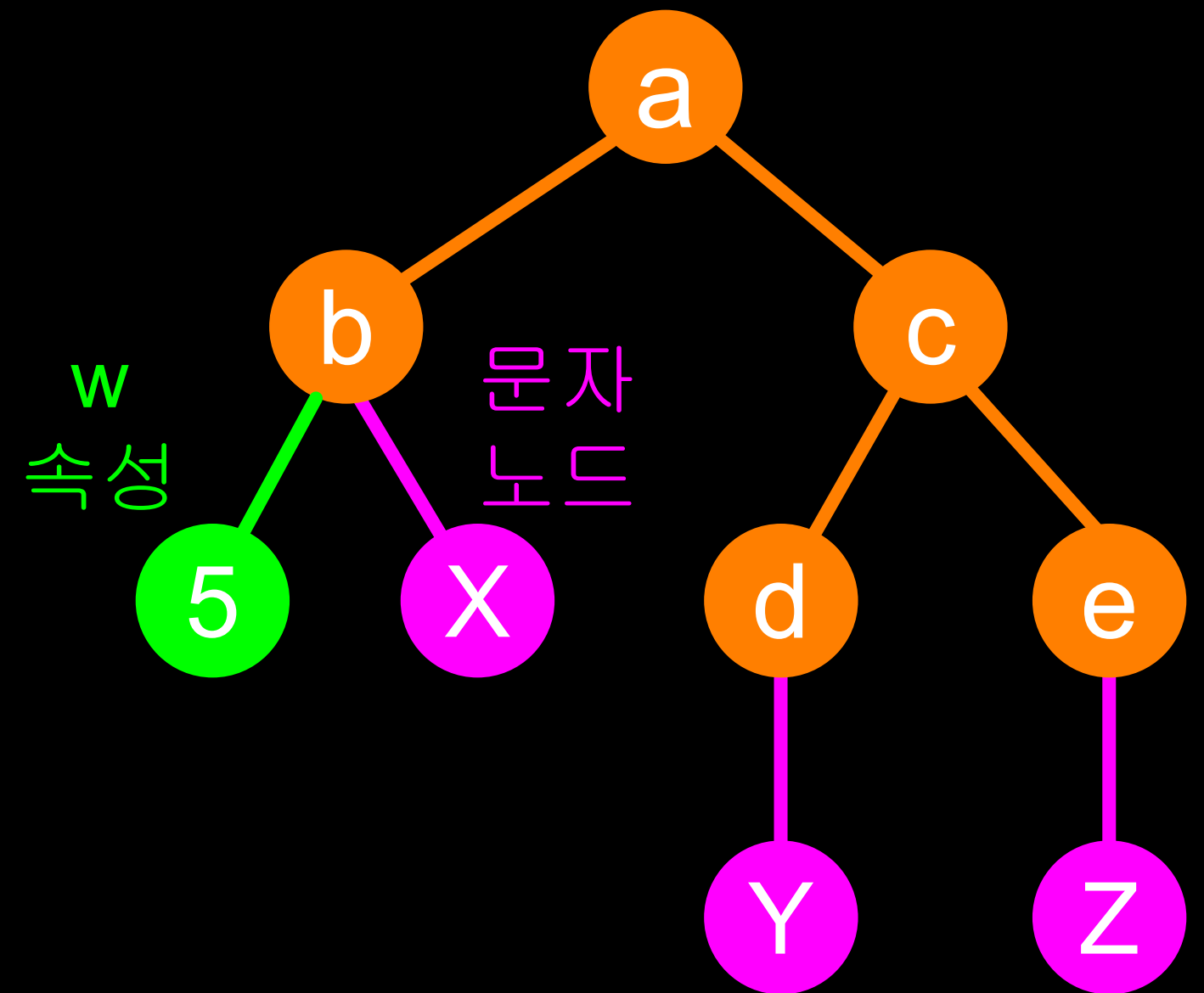


XML 문자와 속성

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

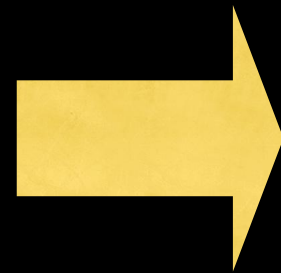
요소

문자



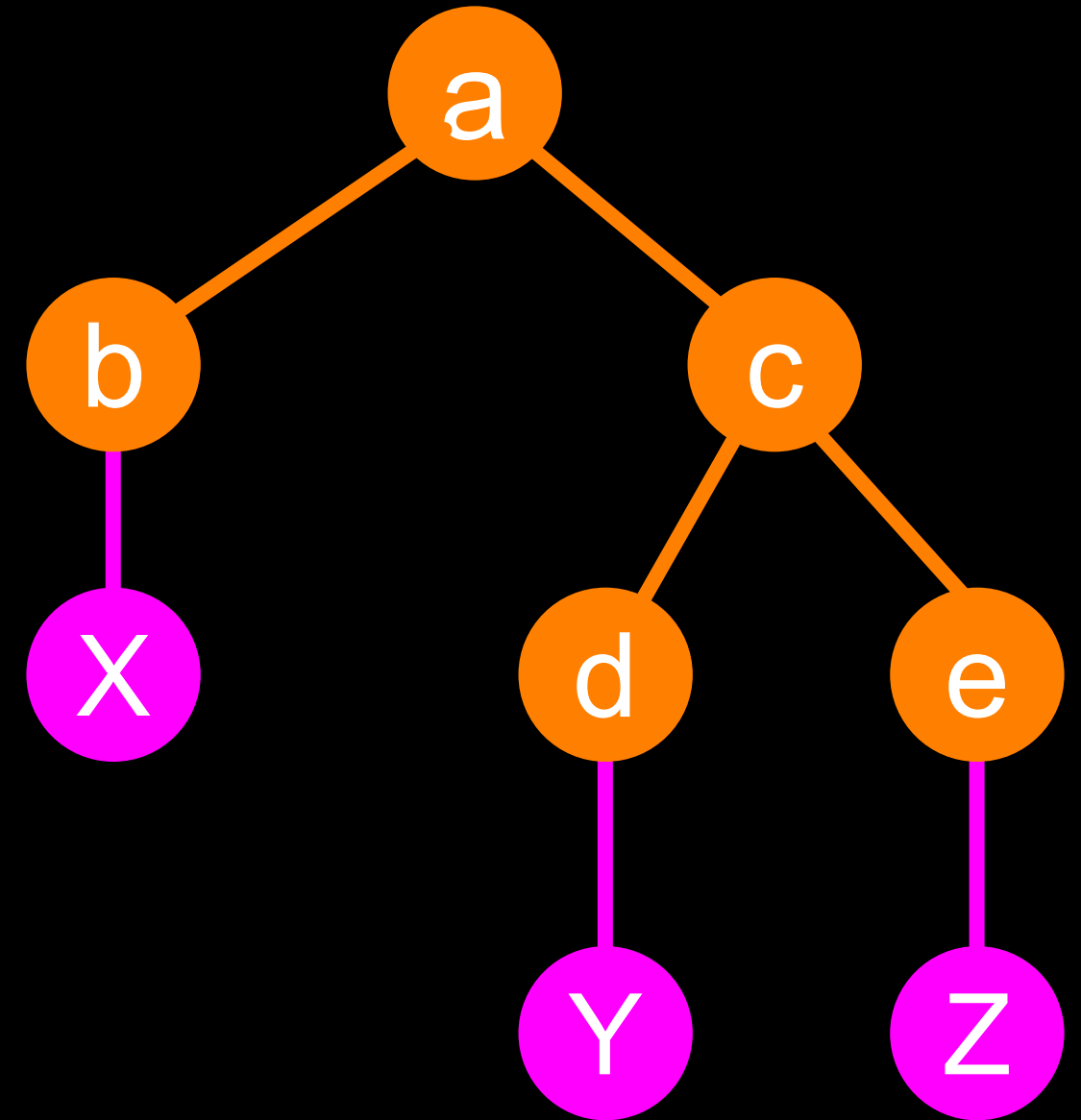
XML 경로

```
<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.wikibooks.org/wiki/XML_Schema

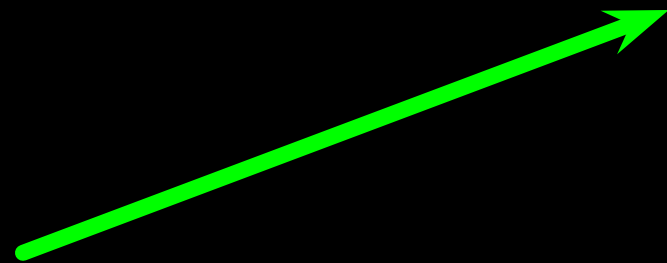
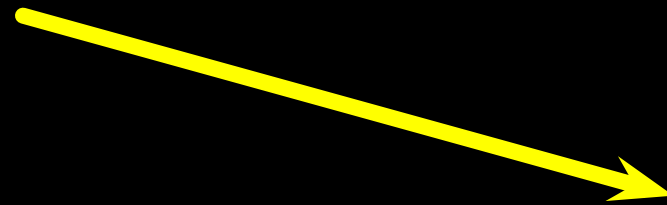
XML 스키마

- 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



검증기

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

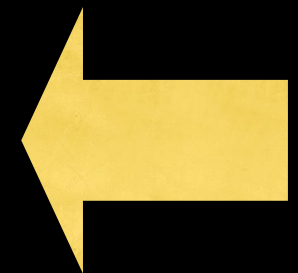


검증기

여러 XML 스키마 언어

- 문서 타입 정의 Document Type Definition (DTD)
 - http://en.wikipedia.org/wiki/Document_Type_Definition
- 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



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/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>
```



```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shiporder">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="orderperson" type="xs:string"/>
      <xs:element name="shipto">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="item" maxOccurs="unbounded">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="title" type="xs:string"/>
            <xs:element name="note" type="xs:string" minOccurs="0"/>
            <xs:element name="quantity" type="xs:positiveInteger"/>
            <xs:element name="price" type="xs:decimal"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="orderid" type="xs:string" use="required"/>
  </xs:complexType>
</xs:element>
</xs:schema>
```

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"))
```

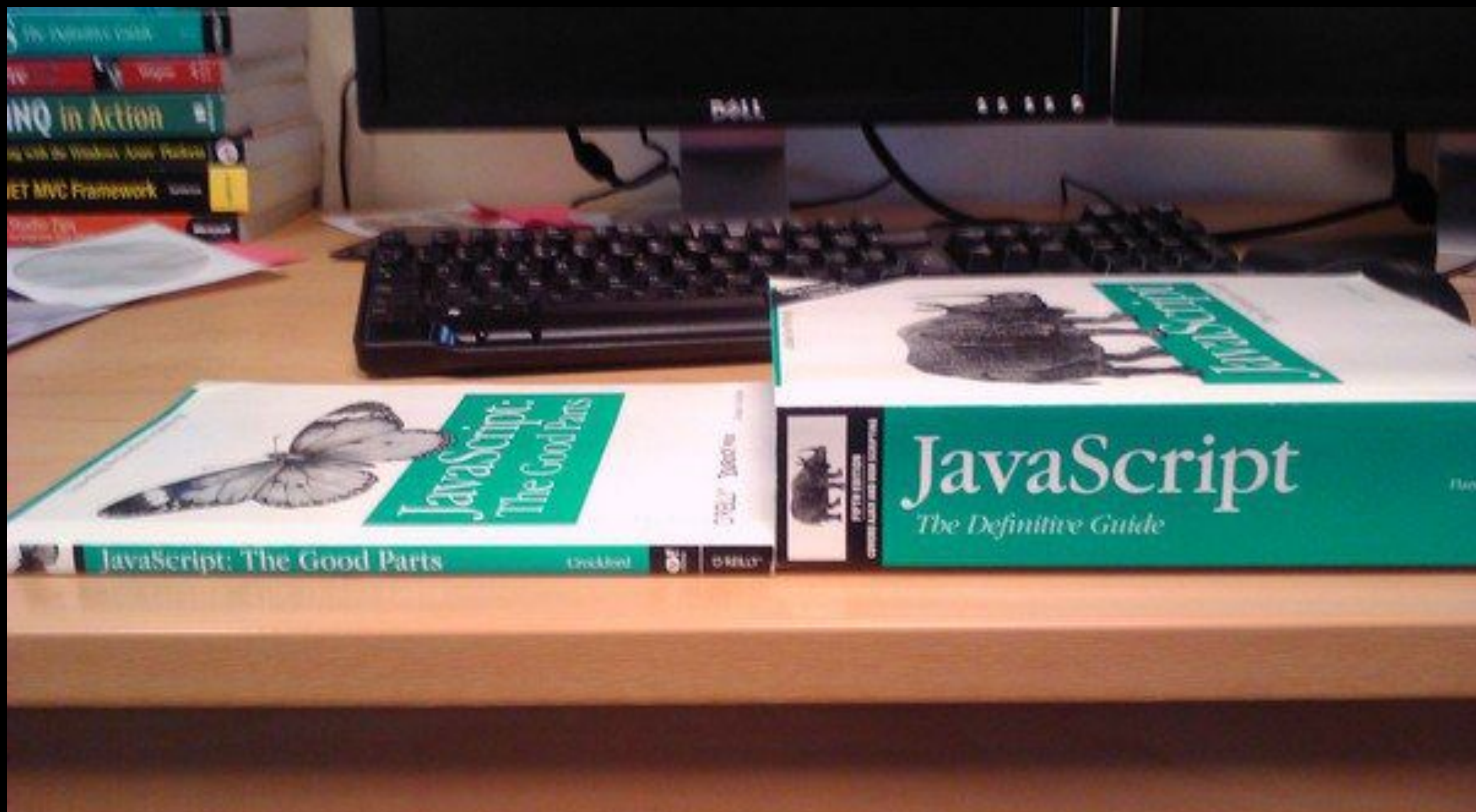
JavaScript Object Notation

JavaScript Object Notation

- Douglas Crockford - JSON을 “발견”
- 자바스크립트의 객체 표현 방식




<http://www.youtube.com/watch?v=kc8BAR7SHJI>



JSON

◀ ▶ ⌂ + 🔍 http://www.json.org/ ↻ 🔍 Google



Introducing JSON

العربية Български 中文 Český Nederlandse Dansk English Esperanto Française Deutsch Ελληνικά עברית Magyar Indonesia Italiano 日本 한국어 فارسی Polski Português Română Русский Српски Slovenščina Español Svenska Türkçe Tiếng Việt

JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the [JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999](#). JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

In JSON, they take on these forms:

An *object* is an unordered set of name/value pairs. An object begins with { (left brace) and ends with } (right

object

{ }

{ *members* }

members

pair

pair , *members*

pair

string : *value*

array

[]

[*elements*]

elements

value

value , *elements*

value

string

number

object

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'])
```

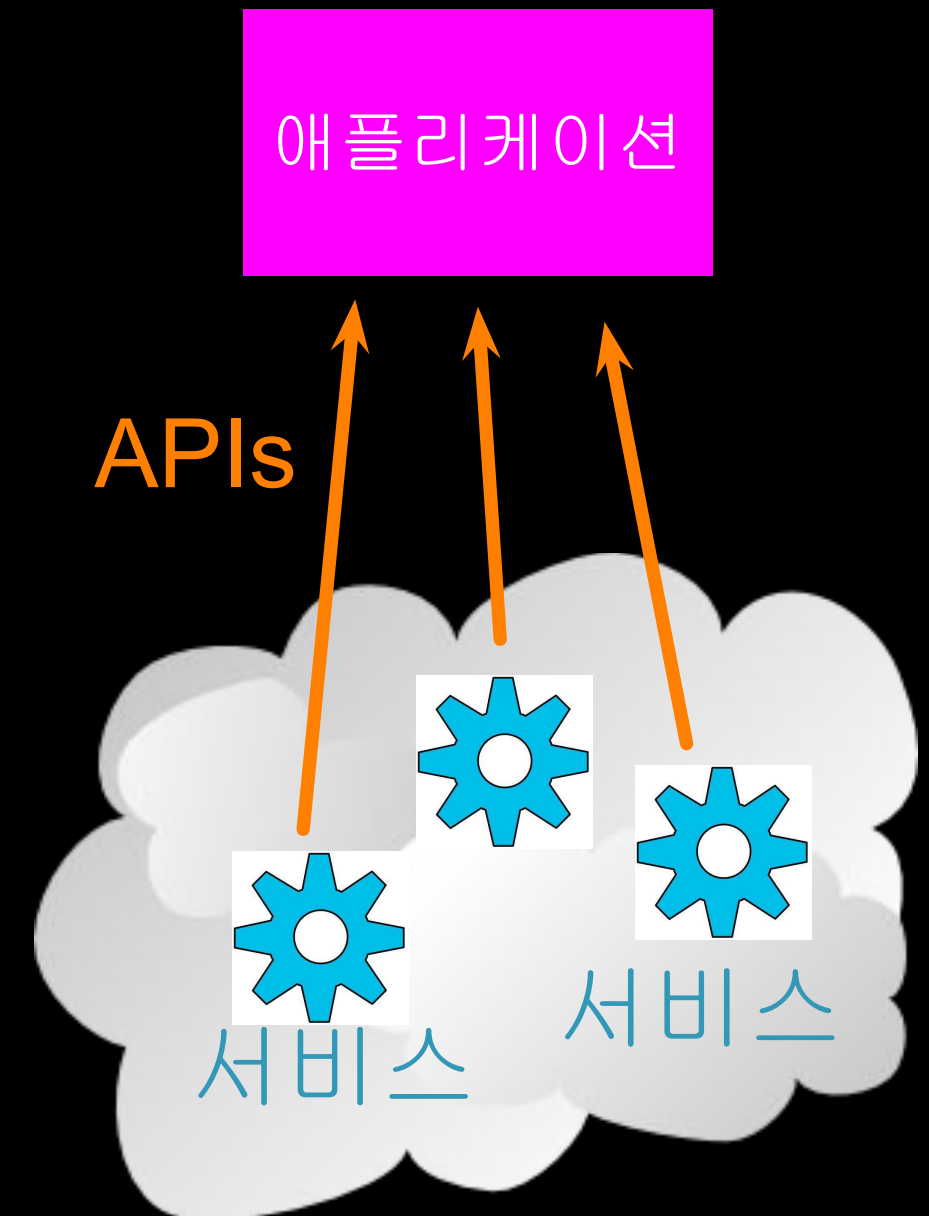
JSON 은 데이터를
중첩된 “리스트”와
“딕셔너리” 로 표현

서비스 지향적 접근

http://en.wikipedia.org/wiki/Service-oriented_architecture

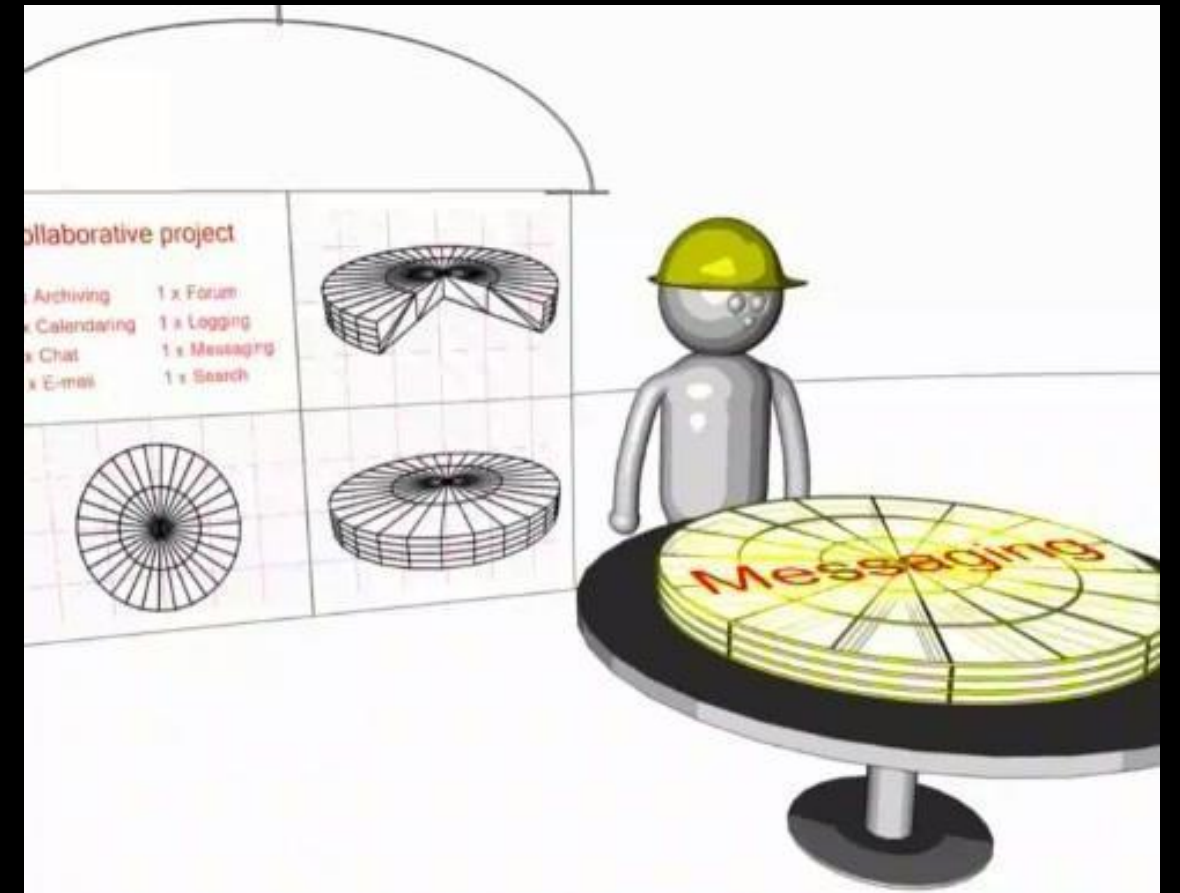
서비스 지향적 접근

- 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

웹 서비스

http://en.wikipedia.org/wiki/Web_services


응용 프로그램 인터페이스(API)

API는 인터페이스를 지정하고 그 인터페이스의 객체의 행동을 제어한다는 점에서 매우 추상적. API에 명시된 기능을 제공하는 소프트웨어를 API의 “실행”이라고 하며, API는 대체로 애플리케이션을 구성하게 되는 언어로 정의됨.

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


Getting Started | Google Map x

← → ↺ <https://developers.google.com/maps/documentation/geocoding/start> ☆ ⚙ ⋮

 Google Maps APIs

Home Documentation Pricing and Plans

🔍 Search

☰ All Products 

Web Services > Geocoding API

GET A KEY VIEW PRICING AND PLANS

GUIDES SUPPORT

SEND FEEDBACK

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

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

<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

API 보안과 비율 제한

- API를 실행하기 위한 계산 자원은 “무료”가 아님
- API를 통해 제공된 데이터는 대체로 매우 가치가 높음
- 데이터의 제공자는 하루 요청량을 제한하여서 API의 “키”를 요구하거나, 사용료를 부과하기도 함
- 발전을 거치면서 여러 규칙들이 바뀌기도 함

Usage Limits

The Google Geocoding API has the following limits in place:

- 2,500 requests per day.

[Google Maps API for Business](#) customers have higher limits:

- 100,000 requests per day.

These limits are enforced to prevent abuse and/or repurposing of the Geocoding API, and may be changed in the future without notice. Additionally, we enforce a request rate limit to prevent abuse of the service. If you exceed the 24-hour limit or otherwise abuse the service, the Geocoding API may stop working for you temporarily. If you continue to exceed this limit, your access to the Geocoding API may be blocked.

The Geocoding API may only be used in conjunction with a Google map; geocoding results without displaying them on a map is prohibited. For complete details on allowed usage, consult the [Maps API Terms of Service License Restrictions](#).

Twitter

Authentication & Authoriz

https://dev.twitter.com/docs/auth

PHP

Developers

API Health

Blog

Discussions

Documentation

Search

Sign in

Home → Documentation

Tweet

Authentication & Authorization

Authentication & Authorization

View

What links here

Updated on Tue, 2013-07-02 12:56

API version 1 API version 1.1

Twitter supports a few authentication methods and with a range of OAuth authentication styles you may be wondering which method you should be using. When choosing which authentication method to use you should understand the way that method will affect your users experience and the way you write your application.

Some of you may already know which type of authentication method you want to use and we want to help you check you've made the right choice.

If you use the...	Send...
REST API	OAuth signed or application-only auth requests
Search API	OAuth signed or application-only auth requests
Streaming API	OAuth signed

[Moving from Basic Auth to OAuth →](#)

Tags

- [OAuth](#) (178)
- [Auth](#) (31)

Tweets | Twitter Developer

← → ↺ ⌂ <https://dev.twitter.com/docs/platform-objects/tweets> ☆ 🗨️ PHP ☰

Developers API Health Blog Discussions Documentation Search Sign in

Home → Documentation → Platform Objects Tweet

Tweets


View

What links here

Updated on Tue, 2013-08-13 17:29

API version 1 API version 1.1

Tweets are the basic atomic building block of all things Twitter. [Users tweet](#) Tweets, also known more generically as "status updates." Tweets can be [embedded](#), [replied to](#), [favorited](#), [unfavorited](#) and [deleted](#).



Brian Sutorius

@bsuto

Follow

The "http://" at the beginning of URLs is a command to the browser. It stands for "head to this place:" followed by two laser-gun noises.

4:29 PM - 21 Feb 2012

4,218 RETWEETS 1,768 FAVORITES


Field Guide

Consumers of Tweets should tolerate the addition of new fields and variance in ordering of fields with ease. Not all fields appear in all contexts. It is generally safe to consider a nulled field, an empty set, and the absence of a field as the same thing. Please note that Tweets found in Search results vary somewhat in structure from this document.

Field	Type	Description
annotations	Object	Unused. Future/beta home for status annotations.

Natural habitat

Tweets can be found [alone](#), within [user objects](#), but most often within [timelines](#).



Related API Resources

- [GET favorites](#)

Twitter

REST API v1.1 Resources

← → ↺ ⌂

https://dev.twitter.com/docs/api/1.1

☆ 🔔 PHP ☰

Developers

API Health

Blog

Discussions

Documentation

Search

Sign in

[Home](#)

REST API v1.1 Resources

Jump to

Timelines

Timelines are collections of Tweets, ordered with the most recent first.

Resource	Description
GET statuses/mentions_timeline	Returns the 20 most recent mentions (tweets containing a users's @screen_name) for the authenticating user. The timeline returned is the equivalent of the one seen when you view your mentions on twitter.com. This method can only return up to 800 tweets. See Working with Timelines for...
GET statuses/user_timeline	Returns a collection of the most recent Tweets posted by the user indicated by the screen_name or user_id parameters. User timelines belonging to protected users may only be requested when the authenticated user either "owns" the timeline or is an approved follower of the owner. The timeline...
GET statuses/home_timeline	Returns a collection of the most recent Tweets and retweets posted by the authenticating user and the users they follow. The home timeline is central to how most users interact with the Twitter service. Up to 800 Tweets are obtainable on the home timeline. It is more volatile for users that follow...
GET statuses/retweets_of_me	Returns the most recent tweets authored by the authenticating user that have been retweeted by others. This timeline is a subset of the user's GET statuses/user_timeline. See Working with Timelines for instructions on traversing timelines.

Tweets

Tweets are the atomic building blocks of Twitter, 140-character status updates with additional associated metadata. People tweet for a variety of reasons about a multitude of topics.

Resource	Description
----------	-------------

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

twitter2.py

```
TWITTER_URL = 'https://api.twitter.com/1.1/friends/list.json'
```

```
while True:
    print('')
    acct = input('Enter Twitter Account:')
    if (len(acct) < 1): break
    url = twurl.augment(TWITTER_URL,
                        {'screen_name': acct, 'count': '5'})
    print('Retrieving', url)
    connection = urllib.request.urlopen(url)
    data = connection.read().decode()
    headers = dict(connection.getheaders())
    print('Remaining', headers['x-rate-limit-remaining'])
    js = json.loads(data)
    print(json.dumps(js, indent=4))

    for u in js['users']:
        print(u['screen_name'])
        s = u['status']['text']
        print('    ', s[:50])
```


Enter Twitter Account:drchuck
Retrieving https://api.twitter.com/1.1/friends ...
Remaining 14

twitter2.py

```
{
  "users": [
    {
      "status": {
        "text": "@jazzychad I just bought one .__.",
        "created_at": "Fri Sep 20 08:36:34 +0000 2013",
      },
      "location": "San Francisco, California",
      "screen_name": "leahculver",
      "name": "Leah Culver",
    },
    {
      "status": {
        "text": "RT @WSJ: Big employers like Google ...",
        "created_at": "Sat Sep 28 19:36:37 +0000 2013",
      },
      "location": "Victoria Canada",
      "screen_name": "_valeriei",
      "name": "Valerie Irvine",
    },
  ],
}
```

Leahculver

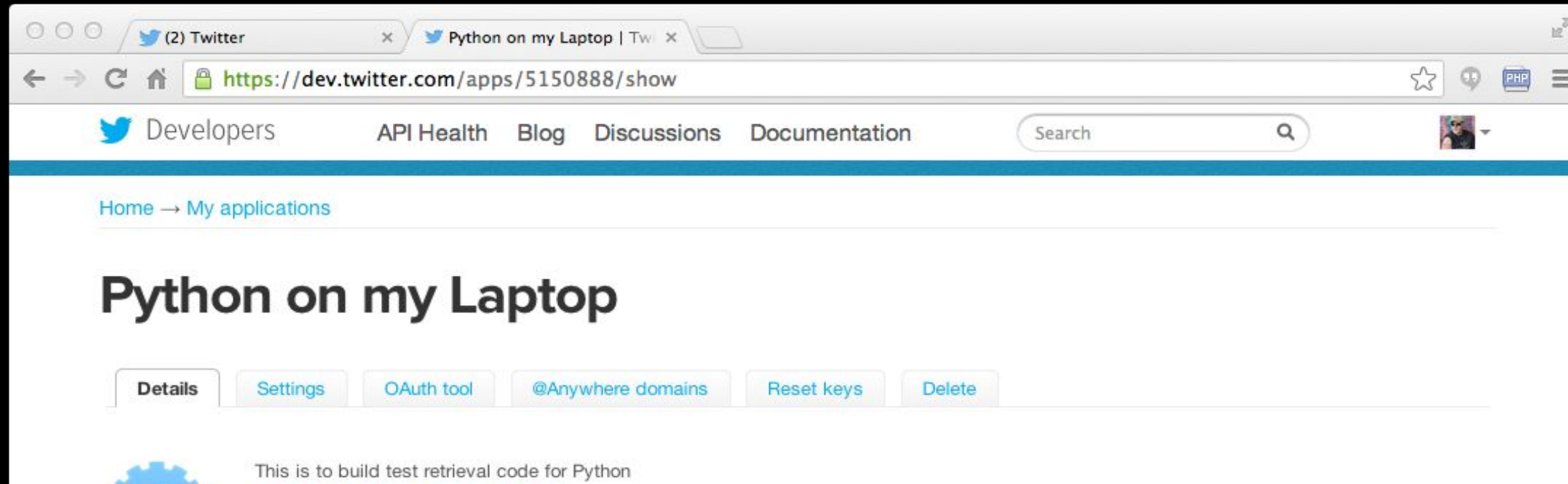
@jazzychad I just bought one .__. _

Valeriei

RT @WSJ: Big employers like Google, AT&T are h
Ericbollens

RT @lukew: sneak peek: my LONG take on the good &a
halherzog

Learning Objects is 10. We had a cake with the LO,



```
def oauth() : hidden.py  
    return { "consumer_key" : "h7Lu...Ng",  
            "consumer_secret" : "dNKenAC3New...mmn7Q",  
            "token_key" : "10185562-ein2...P4GEQQOSGI",  
            "token_secret" : "H0ycCFemmwyf1...qoIpBo" }
```

Access level	Read-only About the application permission model
Consumer key	IuKFhJM5c2nRgyx2S2WQ
Consumer secret	TQ32FrNFhYWrwzIGw?hJM5c2nRgyx2FrNFhYWrwzIGw

(1) Twitter

OAuth | Twitter Developers

← → ↻ 🏠

https://dev.twitter.com/docs/auth/oauth

★ 🗨️ PHP ☰

🐦 Developers

API Health

Blog

Discussions

Documentation

Search

Sign in

Home → Documentation

🐦 Tweet

OAuth

OAuth

View

What links here

Updated on Mon, 2013-03-11 12:22


API version 1

API version 1.1

Related Questions

Send secure authorized requests to the Twitter API

Twitter uses [OAuth](#) to provide authorized access to its API.



Tags

OAuth (178)

Auth (31)

Features

- **Secure** - Users are not required to share their passwords with 3rd party applications, increasing account security.
- **Standard** - A wealth of client libraries and example code are compatible with Twitter's OAuth implementation.

About the Authentication Model

twurl.py

```
import urllib
import oauth
import hidden
```

```
def augment(url, parameters) :
    secrets = hidden.oauth()
    consumer = oauth.OAuthConsumer(secrets['consumer_key'], secrets['consumer_secret'])
    token = oauth.OAuthToken(secrets['token_key'], secrets['token_secret'])
    oauth_request = oauth.OAuthRequest.from_consumer_and_token(consumer,
        token=token, http_method='GET', http_url=url, parameters=parameters)
    oauth_request.sign_request(oauth.OAuthSignatureMethod_HMAC_SHA1(), consumer, token)
    return oauth_request.to_url()
```

```
https://api.twitter.com/1.1/statuses/user_timeline.json?count=2
&oauth_version=1.0&oauth_token=101...SGI&screen_name=drchuck&oa
uth_nonce=09239679&oauth_timestamp=1380395644&oauth_signature=r
LK...BoD&oauth_consumer_key=h7Lu...GNg&oauth_signature_method=H
MAC-SHA1
```

요약

- 서비스 지향 아키텍처 - 애플리케이션이 부분적으로 나뉘어 네트워크 상에 퍼질 수 있게 함
- 응용 프로그램 인터페이스(API) 상호 작용에 대한 계약/약속
- 웹서비스는 애플리케이션끼리 네트워크 상에서 협력할 기반을 제공함 - SOAP와 REST는 웹서비스의 두 가지 형태
- XML과 JSON은 직렬화 형식



Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance (www.dr-chuck.com) of the University of Michigan School of Information and 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)
- Connect Foundation

Translator:

- Jo Ha Nul
- Jeungmin Oh (tangza@gmail.com)