Using Web Services

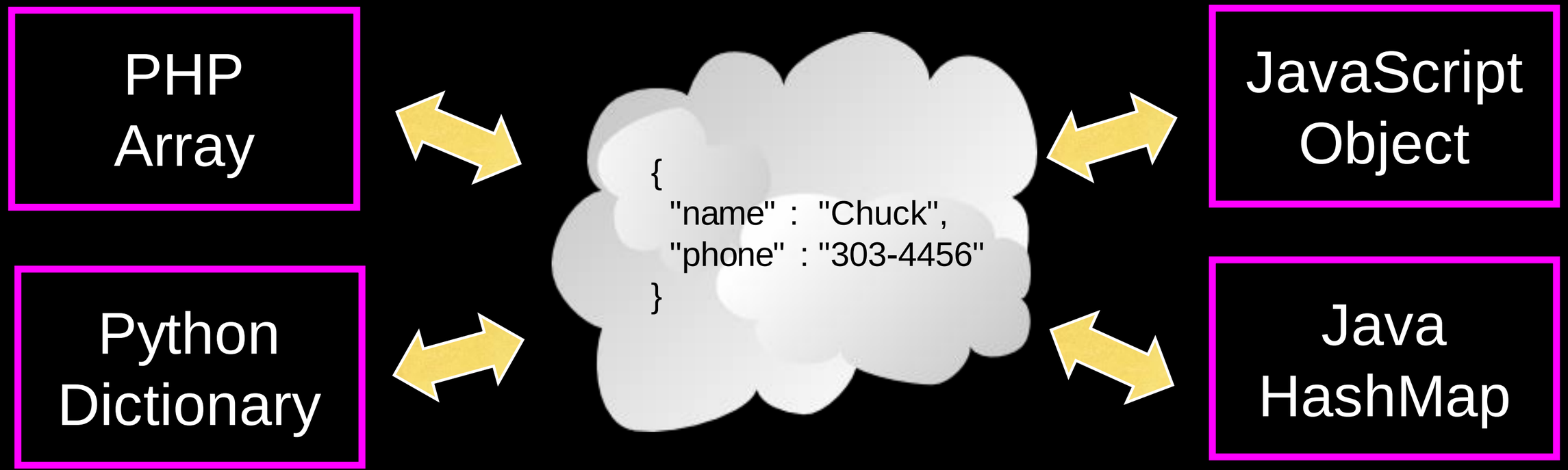
Chapter 13

Python for Everybody
www.py4e.com

Data on the Web

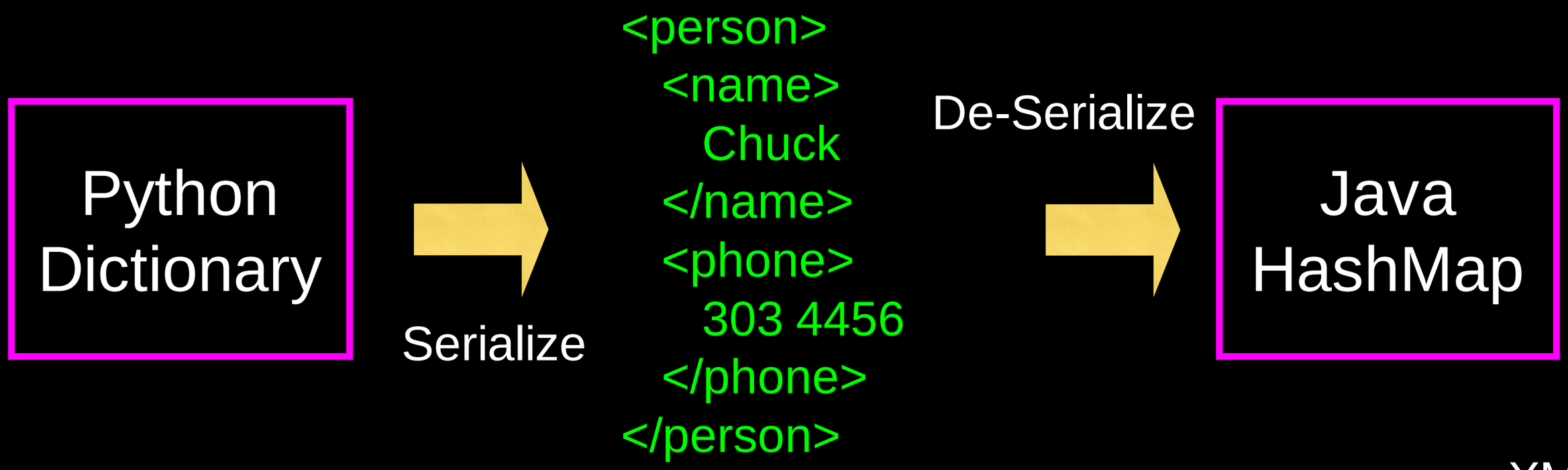
- With the HTTP Request/Response well understood and well supported, there was a natural move toward exchanging data between programs using these protocols
- We needed to come up with an agreed way to represent data going between applications and across networks
- There are two commonly used formats: XML and JSON

Sending Data Across the “Net”

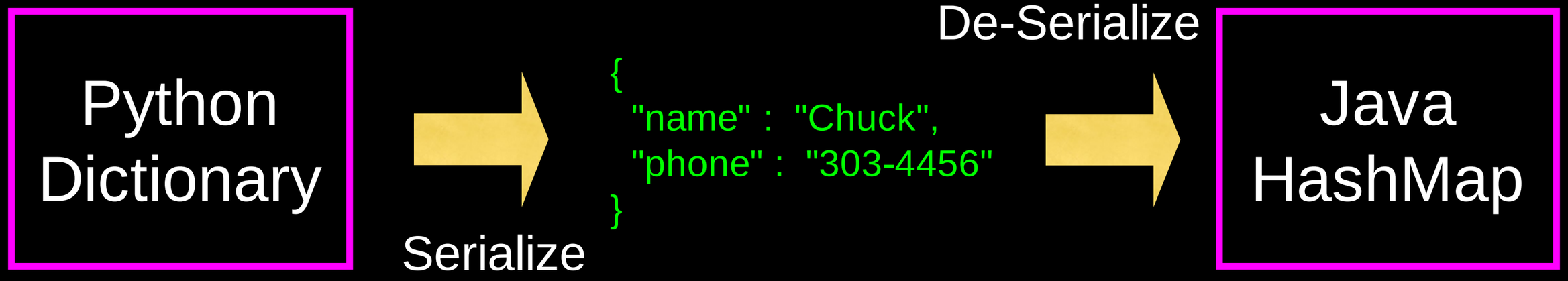


a.k.a. “Wire Protocol” - What we send on the “wire”

Agreeing on a “Wire Format”



Agreeing on a “Wire Format”



JSON

XML

Marking up data to send across the network...

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

XML “Elements” (or Nodes)

- Simple Element
- Complex Element

```
<people>
  <person>
    <name>Chuck</name>
    <phone>303 4456</phone>
  </person>
  <person>
    <name>Noah</name>
    <phone>622 7421</phone>
  </person>
</people>
```

eXtensible Markup Language

- Primary purpose is to help information systems share structured data
- It started as a simplified subset of the Standard Generalized Markup Language (SGML), and is designed to be relatively human-legible

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

XML Basics

- Start Tag
- End Tag
- Text Content
- Attribute
- Self Closing Tag

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

White Space

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

Line ends do not matter.
White space is generally discarded on text elements.
We indent only to be readable.

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

XML Terminology

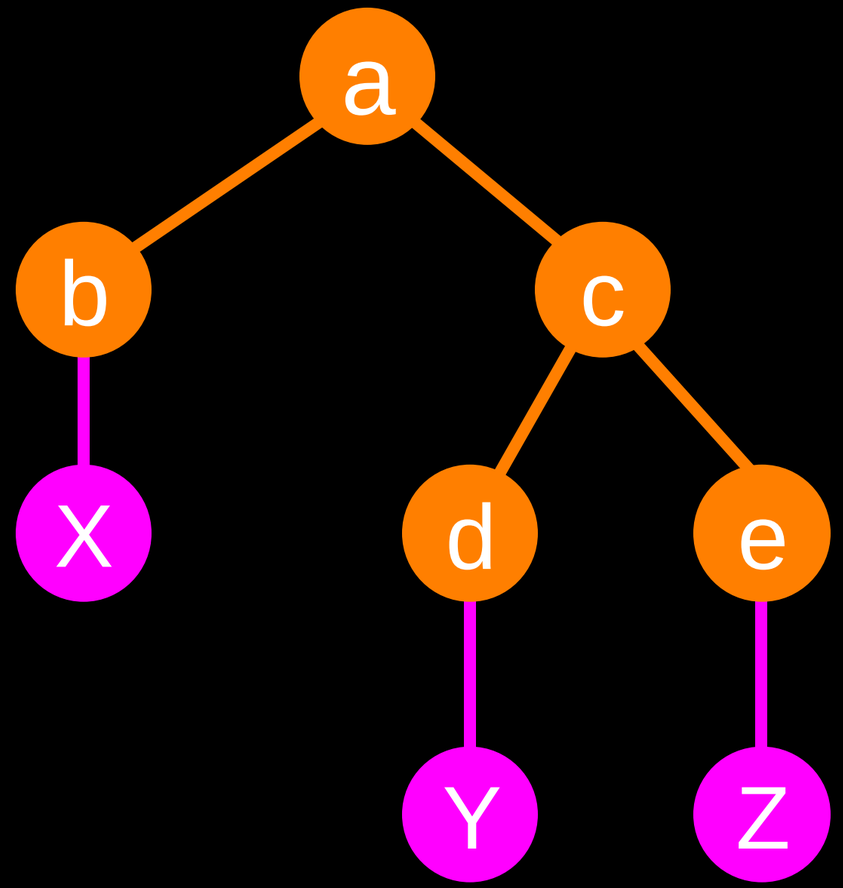
- Tags indicate the beginning and ending of elements
- Attributes - Keyword/value pairs on the opening tag of XML
- Serialize / De-Serialize - Convert data in one program into a common format that can be stored and/or transmitted between systems in a programming language-independent manner

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

XML as a Tree

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

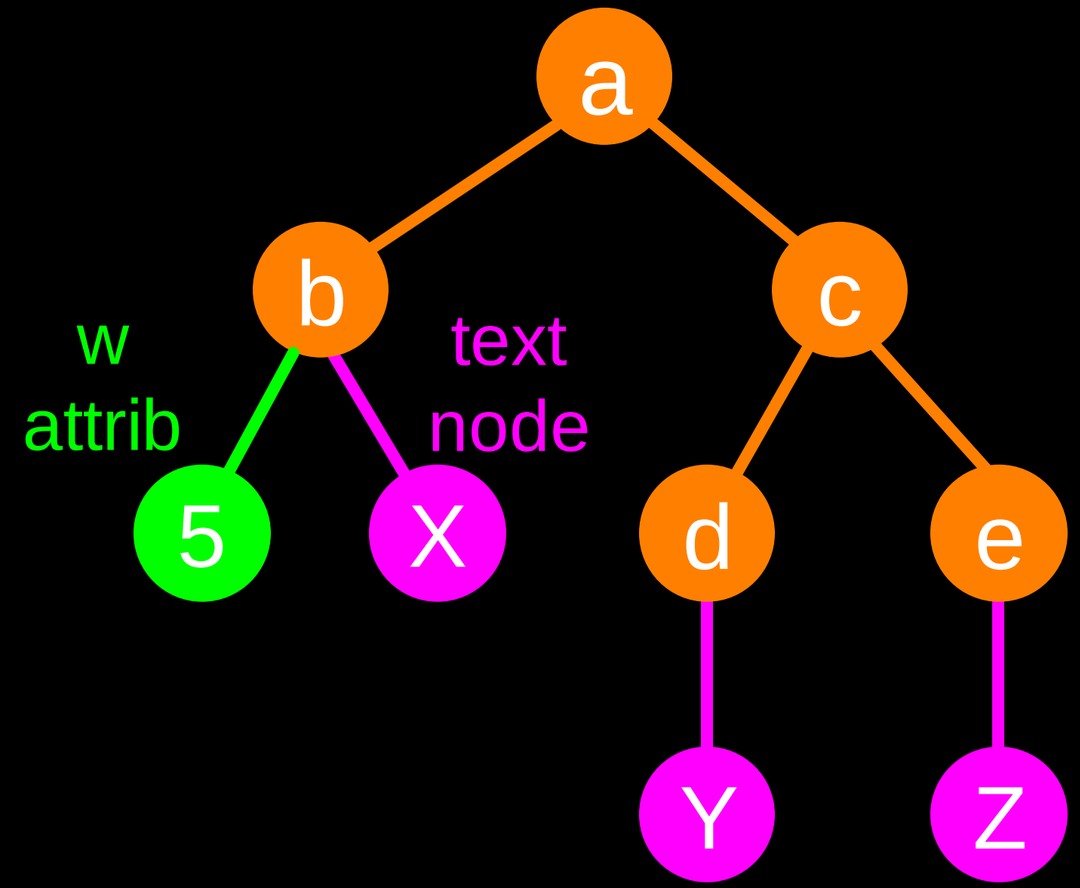
Elements Text



XML Text and Attributes

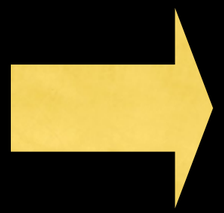
```
<a>
  <b w="5">X</b>
  <c>
    <d>Y</d>
    <e>Z</e>
  </c>
</a>
```

Elements Text

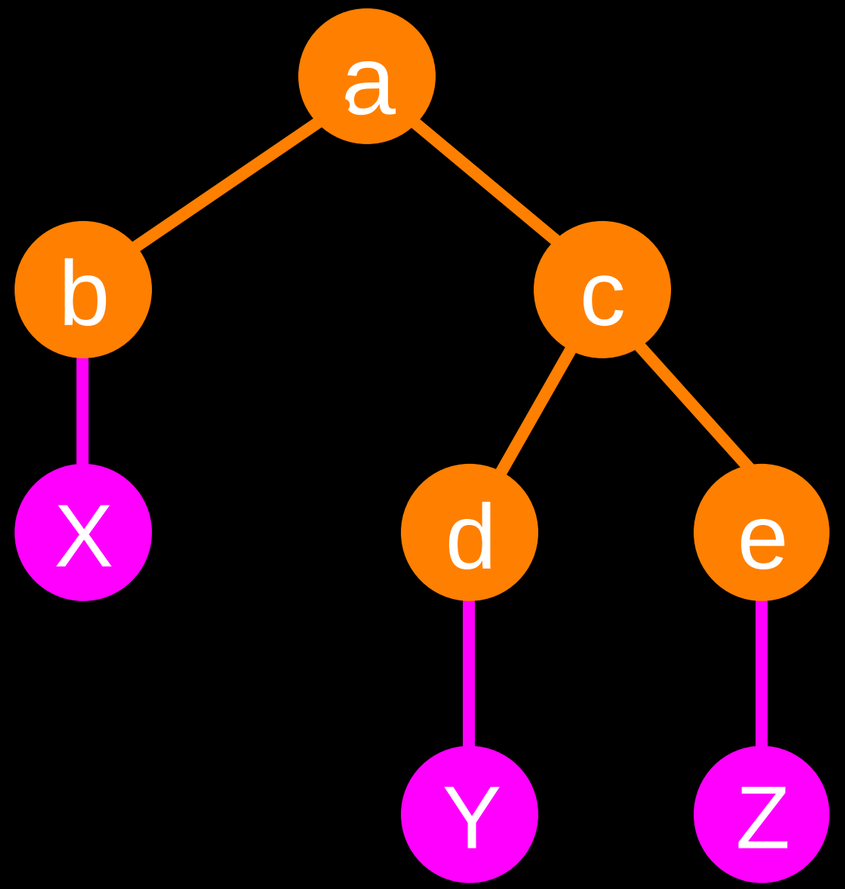


XML as Paths

```
<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 Schema

Describing a “contract” as to what is acceptable XML

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

http://en.wikibooks.org/wiki/XML_Schema

XML Schema

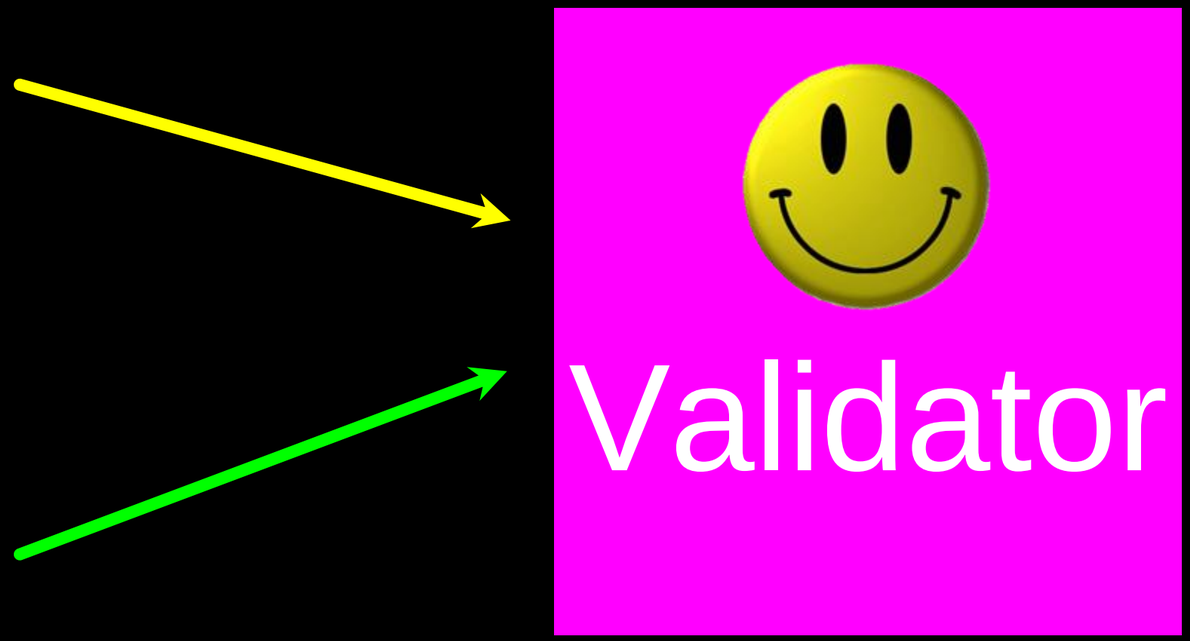
- Description of the legal format of an XML document
- Expressed in terms of constraints on the structure and content of documents
- Often used to specify a “contract” between systems - “My system will only accept XML that conforms to this particular Schema.”
- If a particular piece of XML meets the specification of the Schema - it is said to “validate”

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

XML Validation

XML
Document

XML Schema
Contract



XML Validation

XML Document

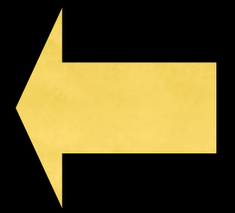
```
<person>
  <lastname>Severance</lastname>
  <age>17</age>
  <dateborn>2001-04-17</dateborn>
</person>
```

XML Schema Contract

```
<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>
```

Many XML Schema Languages

- Document Type Definition (DTD)
  - http://en.wikipedia.org/wiki/Document_Type_Definition
- Standard Generalized Markup Language (ISO 8879:1986 SGML)
  - http://en.wikipedia.org/wiki/SGML
- XML Schema from W3C - (XSD)
  - http://en.wikipedia.org/wiki/XML_Schema_(W3C)



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

XSD XML Schema (W3C spec)

- We will focus on the World Wide Web Consortium (W3C) version
- It is often called “W3C Schema” because “Schema” is considered generic
- More commonly it is called XSD because the file names end in .xsd

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

http://en.wikipedia.org/wiki/XML_Schema_(W3C)

XSD Structure

- 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 Constraints

```
<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>
```

http://www.w3schools.com/Schema/schema_complex_indicators.asp

XSD Data Types

```
<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"/>
```

It is common to represent time in UTC/GMT, given that servers are often scattered around the world

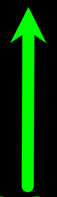
```
<customer>John Smith</customer>
<start>2002-09-24</start>
<startdate>2002-05-30T09:30:10Z</startdate>
<prize>999.50</prize>
<weeks>30</weeks>
```

http://www.w3schools.com/Schema/schema_dtypes_numeric.asp

ISO 8601 Date/Time Format

2002-05-30T09:30:10Z

Year-month-day



Time of day

Timezone - typically specified in UTC / GMT rather than local time zone

http://en.wikipedia.org/wiki/ISO_8601

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

```xml
<?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
<?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>
```

http://www.w3schools.com/Schema/schema_example.asp

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

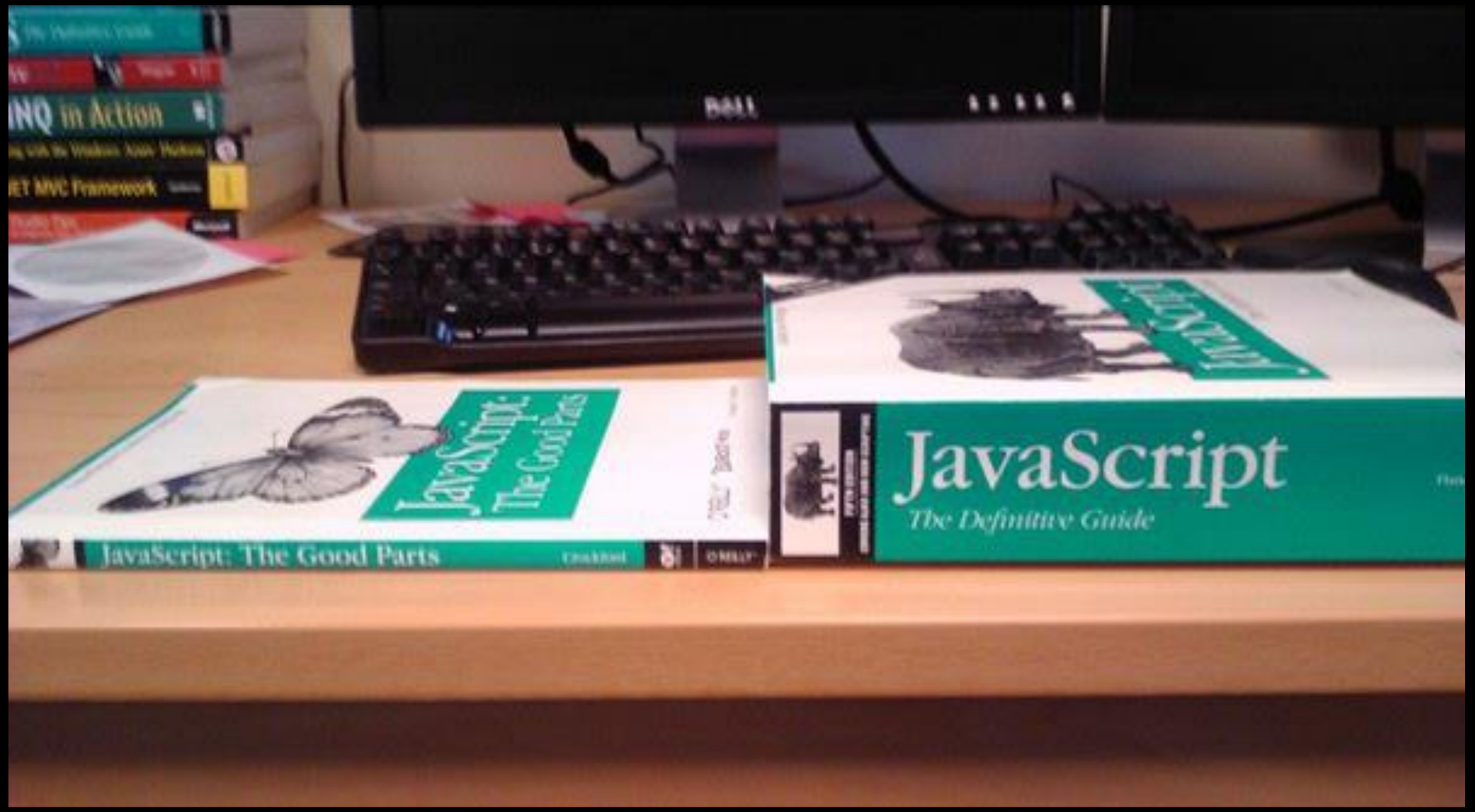
```
import xml.etree.ElementTree as ET
input = '''<stuff>
    <users>
        <user x="2">
            <id>001</id>
            <name>Chuck</name>
        </user>
        <user x="7">
            <id>009</id>
            <name>Brent</name>
        </user>
    </users>
</stuff>'''

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"))
```

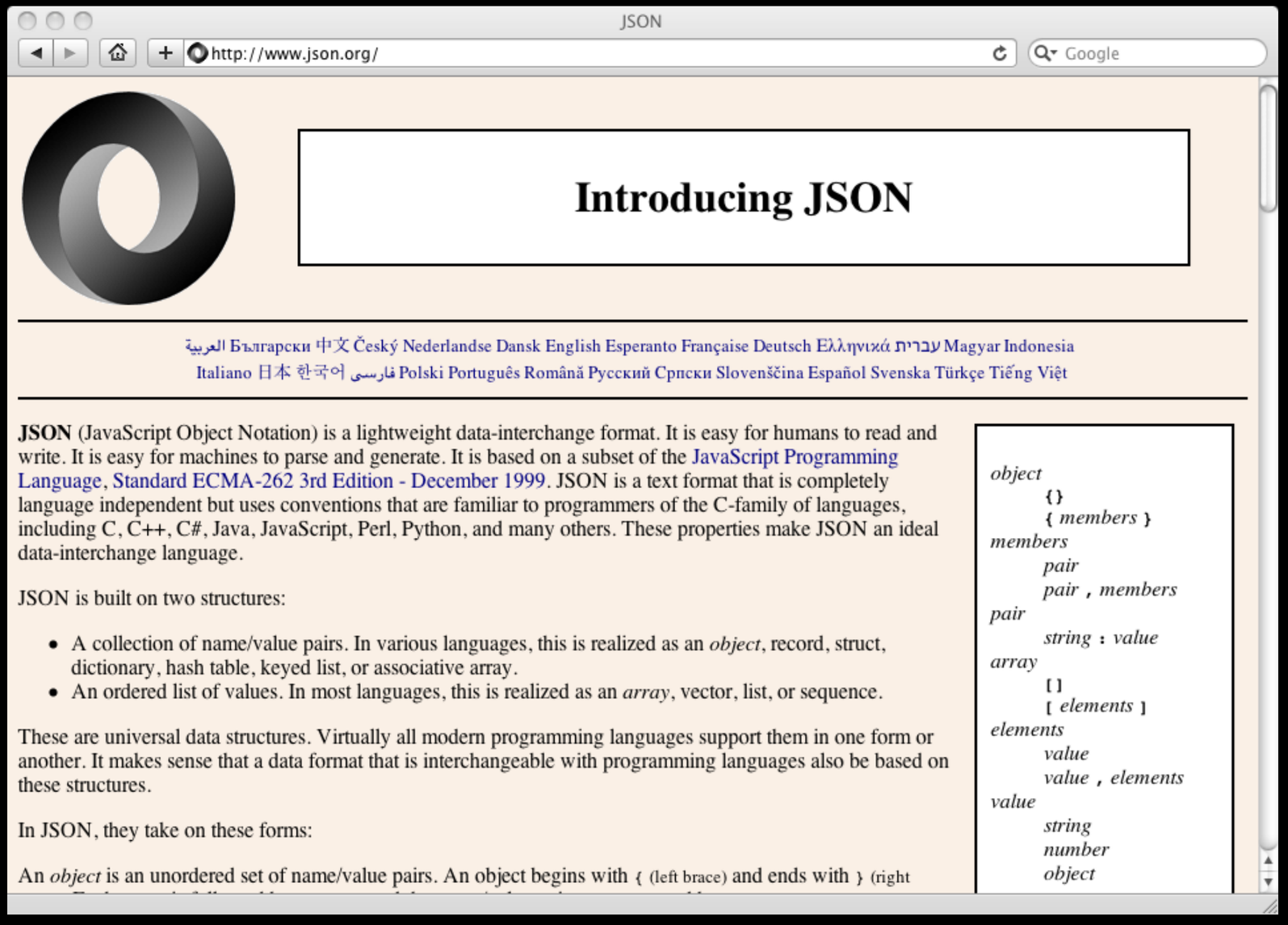
JavaScript Object Notation

JavaScript Object Notation

- Douglas Crockford - “Discovered” JSON
- Object literal notation in JavaScript

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


NQ in Action
ET MVC Framework
JavaScript: The Good Parts
JavaScript
The Definitive Guide


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 represents data as nested “lists” and “dictionaries”

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'])
```

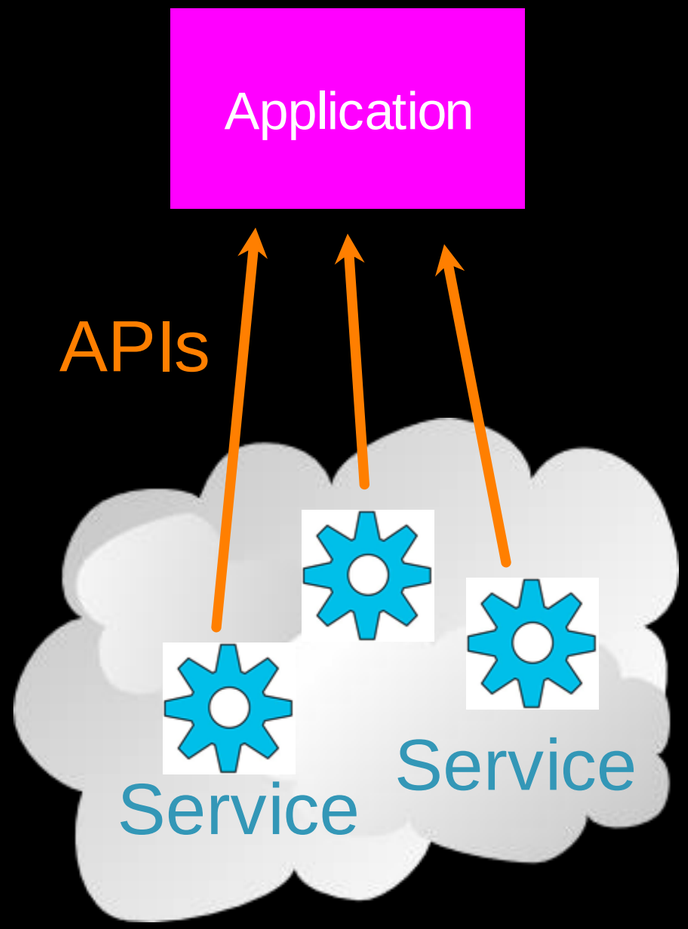
JSON represents data as nested “lists” and “dictionaries”

Service Oriented Approach

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

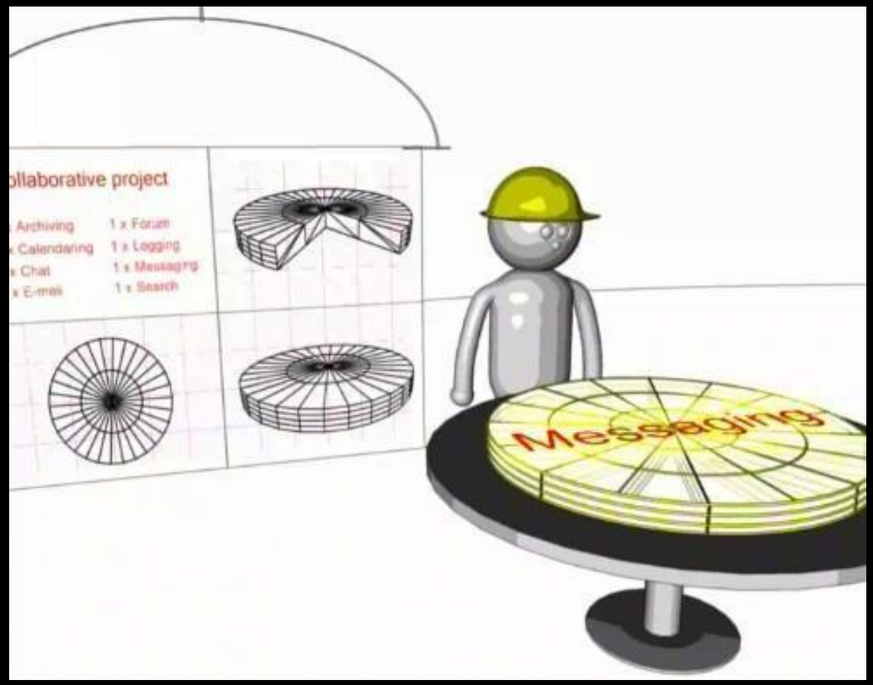
Service Oriented Approach

- Most non-trivial web applications use services
- They use services from other applications
  - Credit Card Charge
  - Hotel Reservation systems
- Services publish the “rules” applications must follow to make use of the service (API)



Multiple Systems

- Initially - two systems cooperate and split the problem
- As the data/service becomes useful - multiple applications want to use the information / application



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

5:15

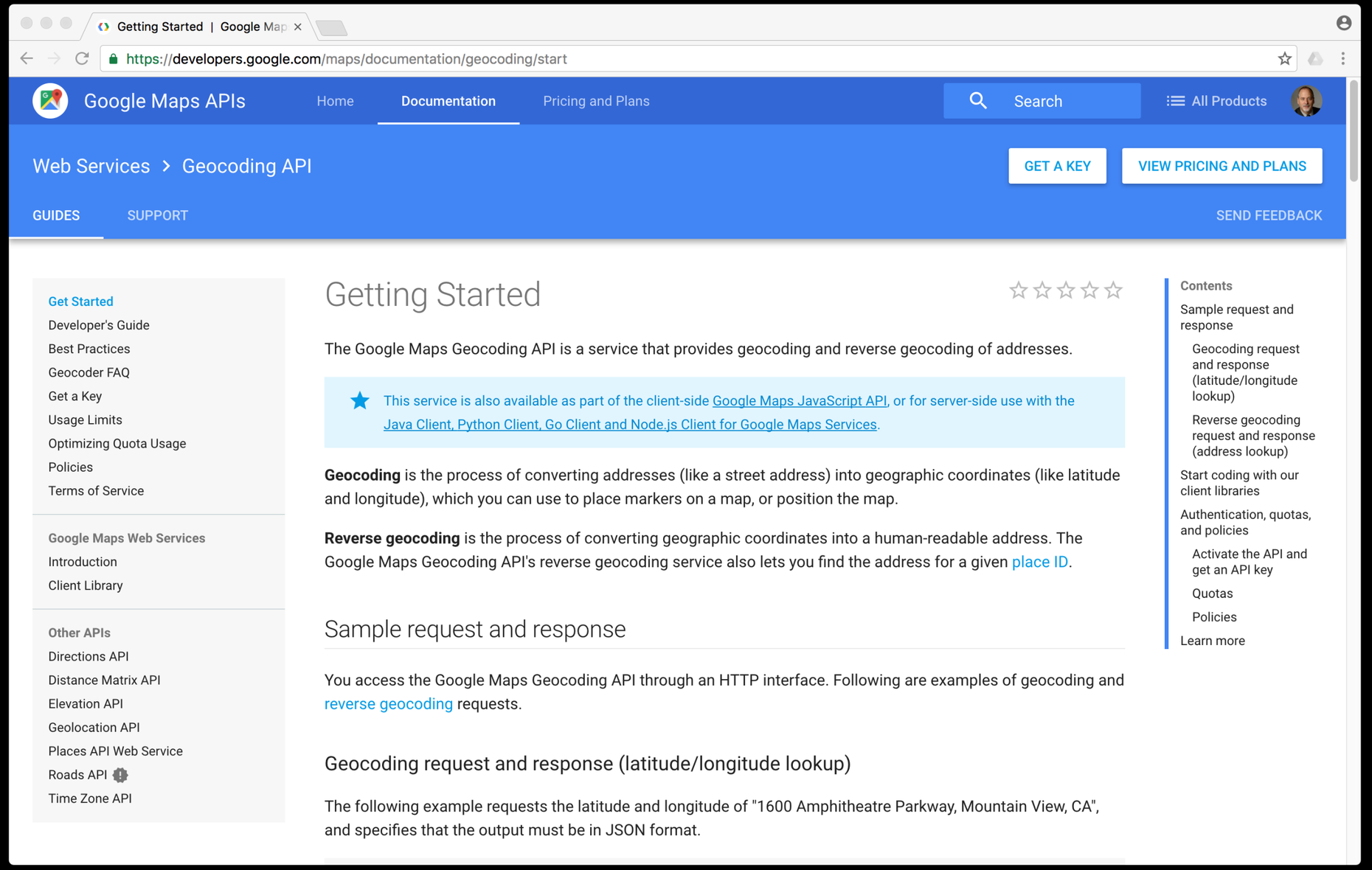
Web Services

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

Application Program Interface

The API itself is largely abstract in that it specifies an interface and controls the behavior of the objects specified in that interface. The software that provides the functionality described by an API is said to be an “implementation” of the API. An API is typically defined in terms of the programming language used to build an application.

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

API Security and Rate Limiting

- The compute resources to run these APIs are not “free”
- The data provided by these APIs is usually valuable
- The data providers might limit the number of requests per day, demand an API “key”, or even charge for usage
- They might change the rules as things progress...

Usage Limits

The Google Geocoding API has the following limits in place:

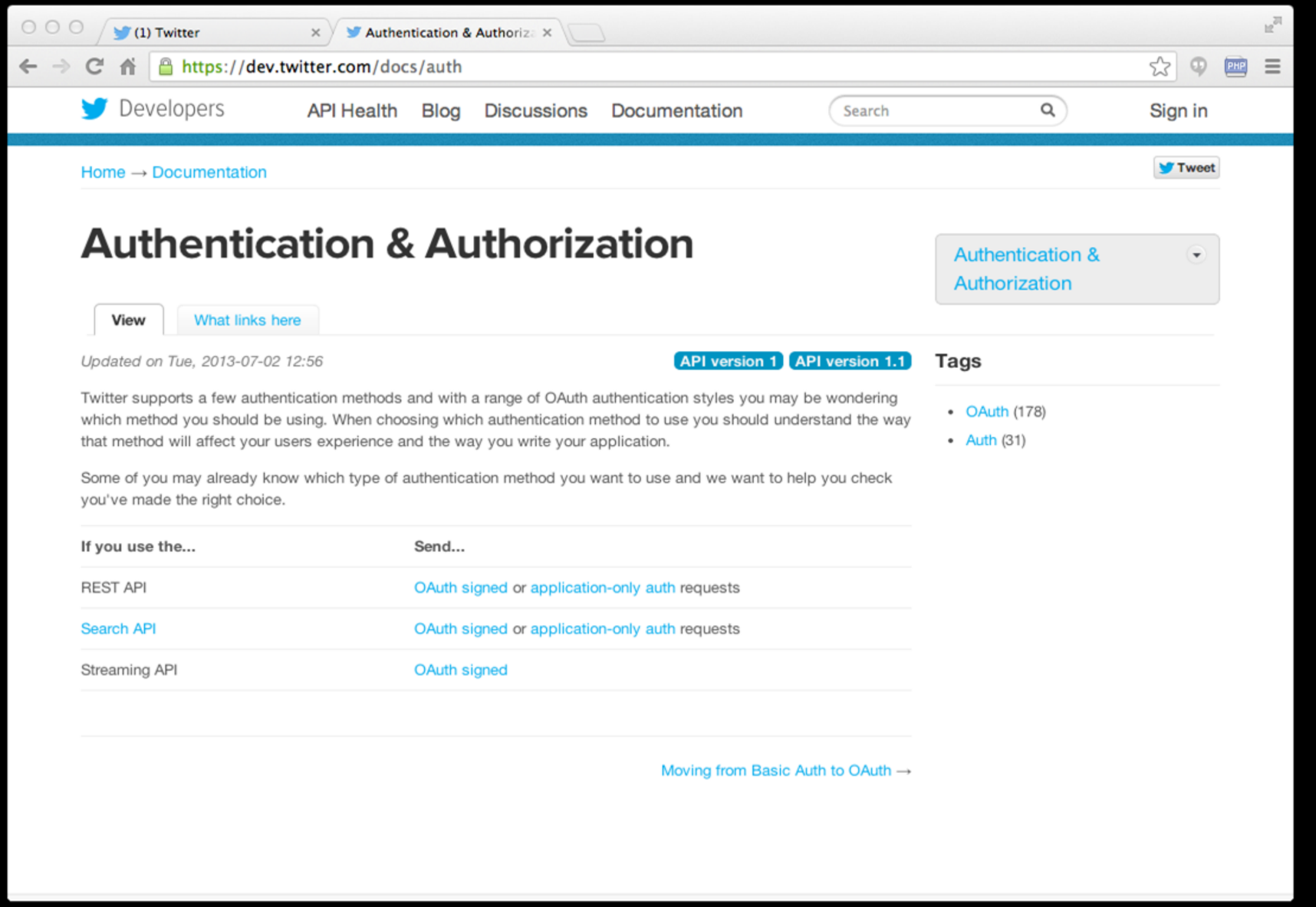
- 2,500 requests per day.

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.


(1) Twitter
Authentication & Authoriz
https://dev.twitter.com/docs/auth
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

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.

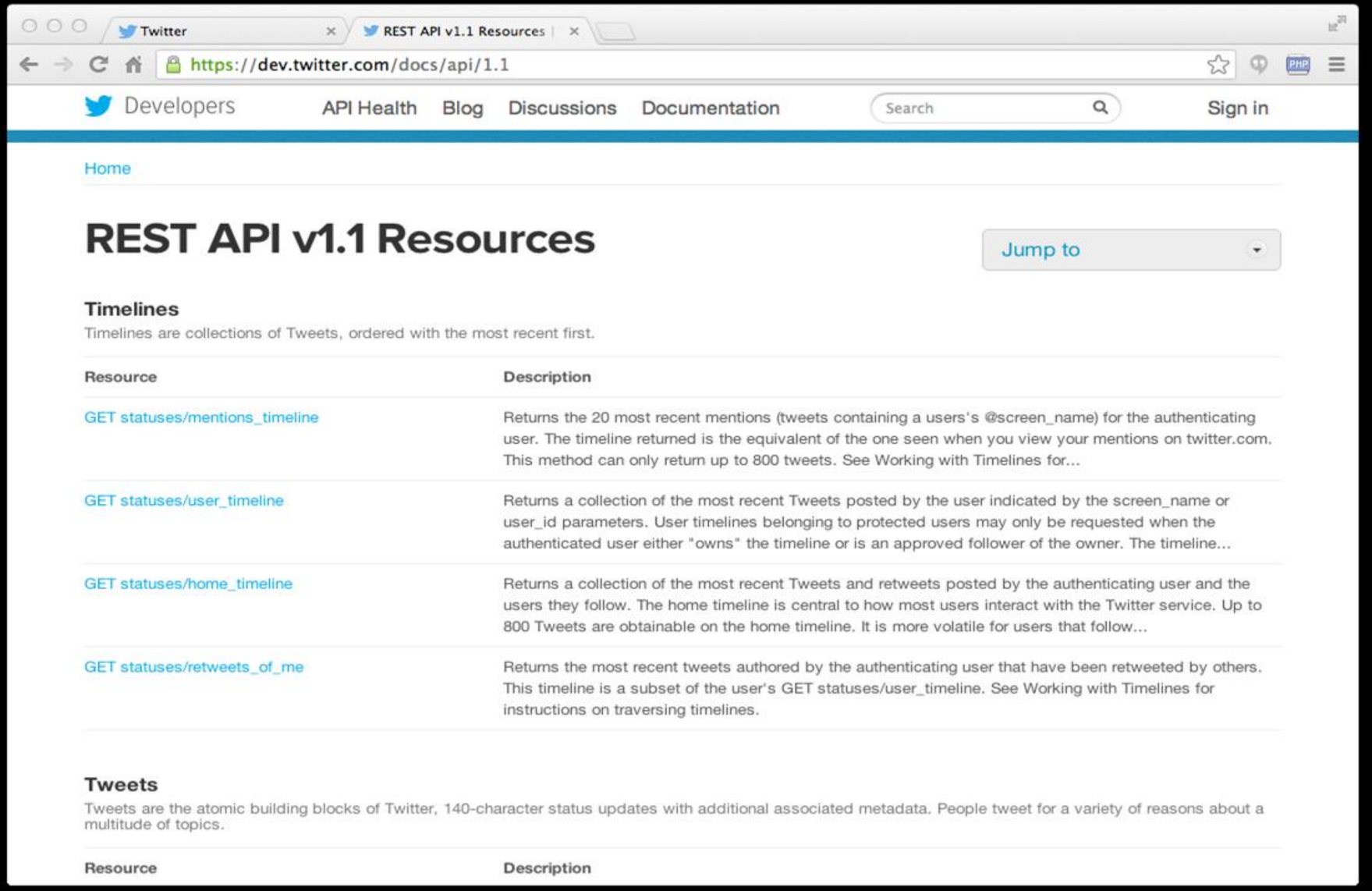
Field	Type	Description
annotations	Object	*Unused.* Future/beta home for status annotations.

Natural habitat

Tweets can be found alone, within user objects, but most often within timelines.

Related API Resources

- GET favorites

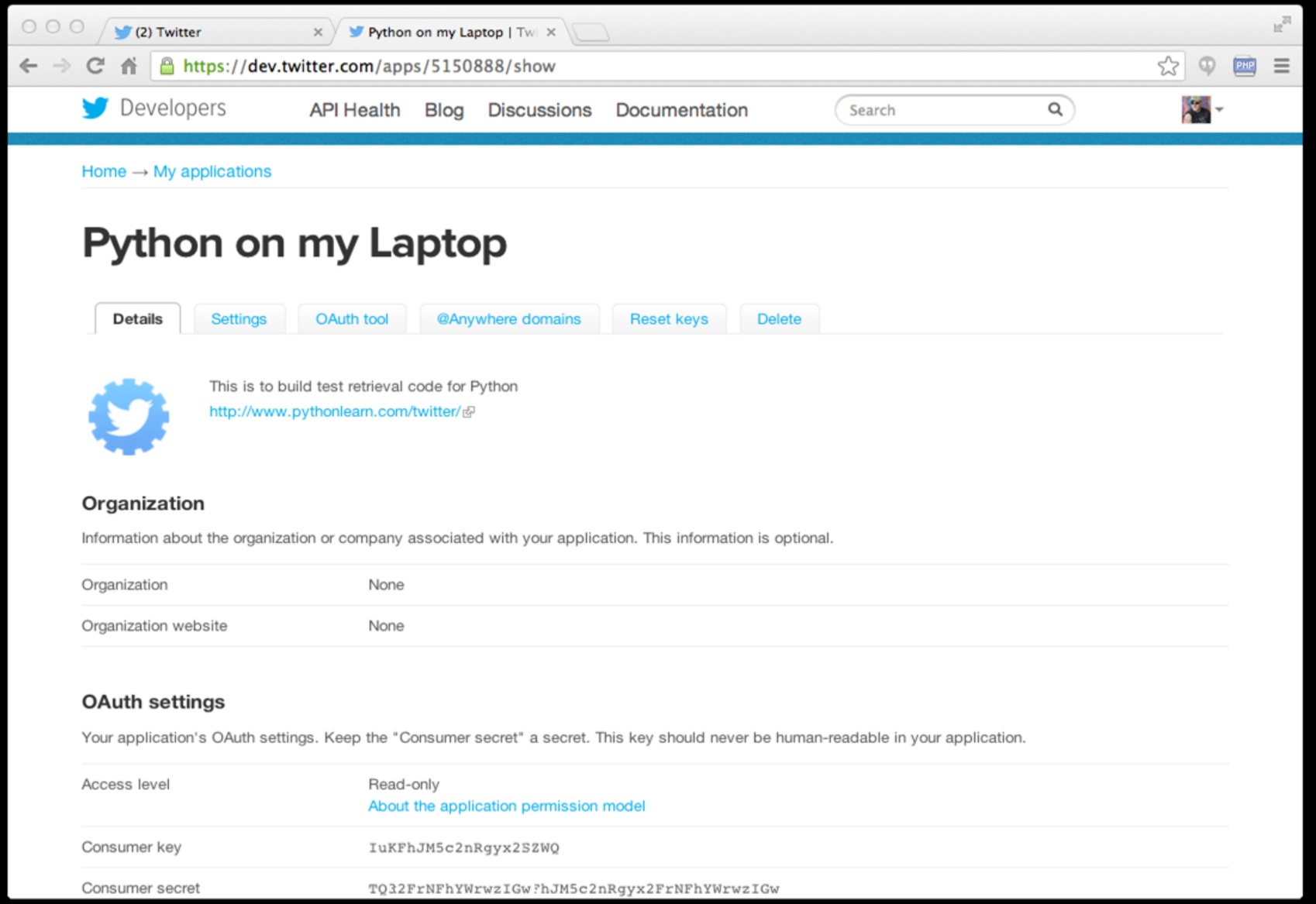

Twitter
REST API v1.1 Resources
https://dev.twitter.com/docs/api/1.1
Developers
API Health
Blog
Discussions
Documentation
Search
Sign in
Home
REST API v1.1 Resources
Jump to
Timelines
Timelines are collections of Tweets, ordered with the most recent first.
Resource
Description
GET statuses/mentions_timeline
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

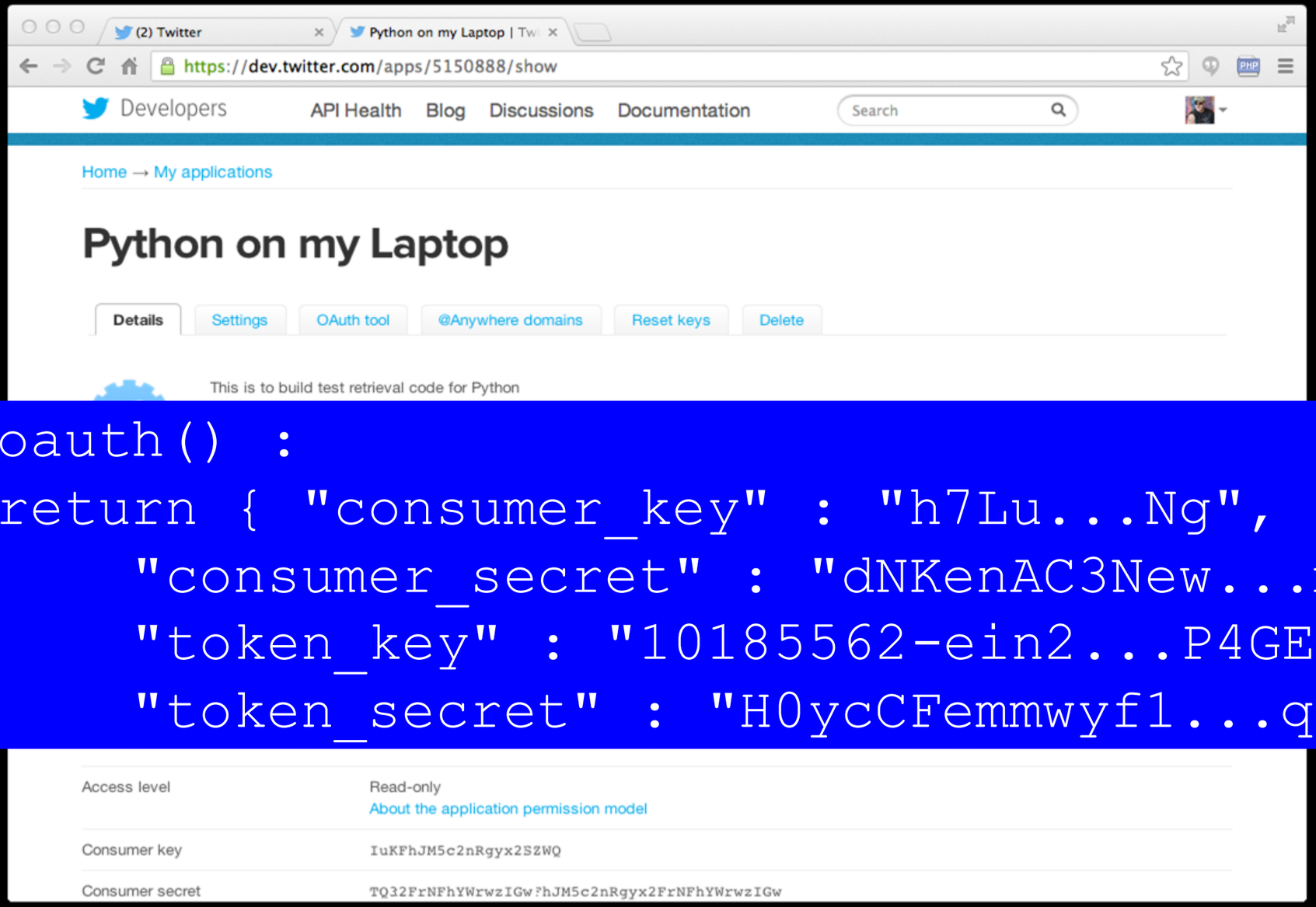
twitter2.py

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

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])
```

twitter2.py

```
Enter Twitter Account:drchuck
Retrieving https://api.twitter.com/1.1/friends ...
Remaining 14
{
    "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&amp;amp;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,
```


(2) Twitter
Python on my Laptop | Tw
https://dev.twitter.com/apps/5150888/show
Developers
API Health
Blog
Discussions
Documentation
Search
Home → My applications
Python on my Laptop
Details
Settings
OAuth tool
@Anywhere domains
Reset keys
Delete
This is to build test retrieval code for Python
http://www.pythonlearn.com/twitter/
Organization
Information about the organization or company associated with your application. This information is optional.
Organization
None
Organization website
None
OAuth settings
Your application's OAuth settings. Keep the "Consumer secret" a secret. This key should never be human-readable in your application.
Access level
Read-only
About the application permission model
Consumer key
IuKFhJM5c2nRgyx2SZWQ
Consumer secret
TQ32FrNFhYWrwzIGw?hJM5c2nRgyx2FrNFhYWrwzIGw



hidden.py

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

(1) Twitter | OAuth | Twitter Developers

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

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

Send secure authorized requests to the Twitter API

Twitter uses OAuth to provide authorized access to its API.

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.

Related Questions

- Do Twitter's OAuth 1.0A access tokens expire?
- Will an application have to request user authorization just to make public API calls?

Tags

- OAuth (178)
- Auth (31)

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&oauth_nonce=09239679&oauth_timestamp=1380395644&oauth_signature=rLK...BoD&oauth_consumer_key=h7Lu...GNg&oauth_signature_method=HMAC-SHA1

Summary

- Service Oriented Architecture - allows an application to be broken into parts and distributed across a network
- An Application Program Interface (API) is a contract for interaction
- Web Services provide infrastructure for applications cooperating (an API) over a network - SOAP and REST are two styles of web services
- XML and JSON are serialization formats

Acknowledgements / Contributions

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

… Insert new Contributors here

…