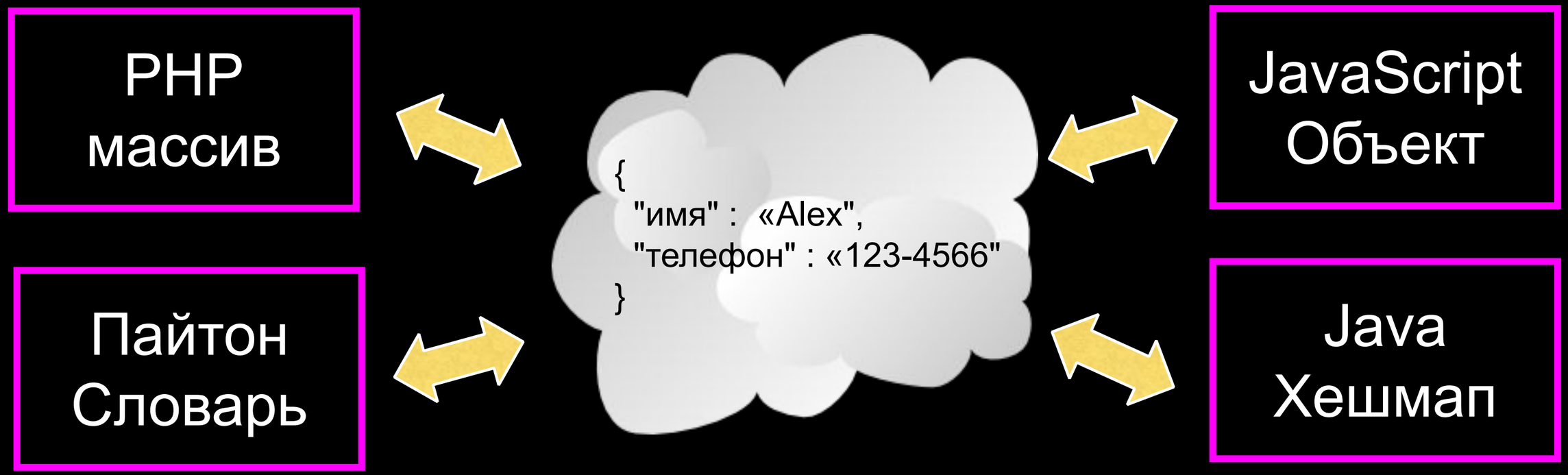
Использование веб-сервисов

Данные в Веб

- Так как технология HTTP (запрос/ответ) хорошо поддерживается, возникает возмжность обмена данными между программами, использующими эти протоколы
- Необходимо было придумать согласованный способ представления данных, передаваемых между приложениями и по сети
- Обычно используются два формата: XML и JSON

Передача данных по «сети»



Также известный как «Проводной протокол передачи данных» — то, что мы отправляем по сети («по проводам»)

Соглашения по формату передачи

Пайтон
Словарь

Сериализовать

```
<person>
  <name>
    Alex
  </name>
  <phone>
    123 4566
  </phone>
</person>
```

Десериализовать

Java
Хешмап

XML

Соглашения по формату передачи

Пайтон
Словарь

Сериализовать

```
{
  "имя" :  “Alex",
  "телефон" :  “123-4566"
}
```

Десериализовать

Java
Хешмап

JSON

XML

Разметка данных для отправки по сети…

https://ru.wikipedia.org/wiki/XML

XML-элементы (Ноды)

- Простой элемент
- Сложный элемент

```
<people>
    <person>
        <name>Alex</name>
        <phone>123 4566</phone>
    </person>
    <person>
        <name>Jon</name>
        <phone>622 7421</phone>
    </person>
</people>
```

Расширяемый язык разметки (XML)

- Основная цель — помочь информационным системам обмениваться структурированными данными
- Появился как подмножество Стандартного обобщенного языка разметки (англ. SGML) и разработан так, чтобы быть понятным человеку

https://ru.wikipedia.org/wiki/XML

Основы XML

- Начальный тег
- Закрывающий тег
- Текстовый контент
- Атрибут
- Самозакрывающийся тег

```
<person>
  <name>Alex</name>
  <phone type=“mob">
    +375293287564
  </phone>
  <email hide="yes" />
</person>
```

Пробелы

```
<person>
  <name>Alex</name>
  <phone type=“mob">
    +375 29 4534785
   </phone>
   <email hide="yes" />
</person>
```

Концы строк не имеют значения.
Пробелы в текстовых элементах обычно отбрасываются.
Мы делаем отступ только для удобства чтения

```
<person>
  <name>Alex</name>
  <phone type=“mob">+375 29 4534785</phone>
  <email hide="yes" />
</person>
```

XML-терминология

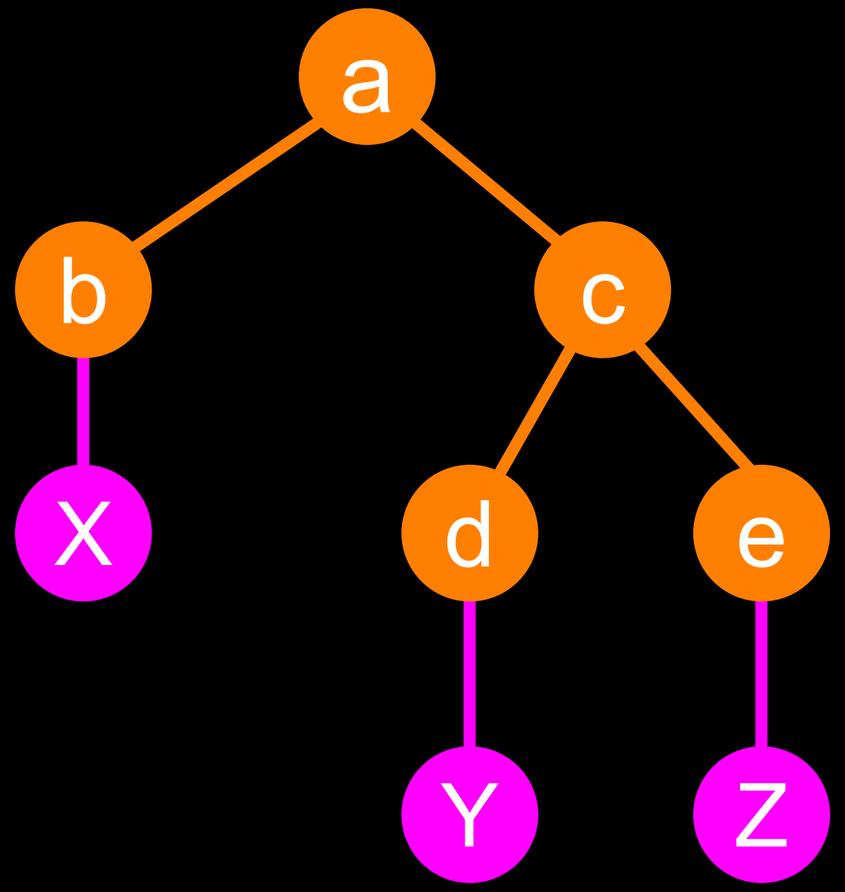
- Теги обозначают начало и конец элемента
- Атрибуты — Пары Ключевое слово/Значение в открывающем теге XML
- Сериализовать / Десериализовать — преобразовать данные программы в общий формат, который может храниться и/или передаваться между системами независимо от языка программирования

https://ru.wikipedia.org/wiki/Сериализация

XML в виде дерева

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

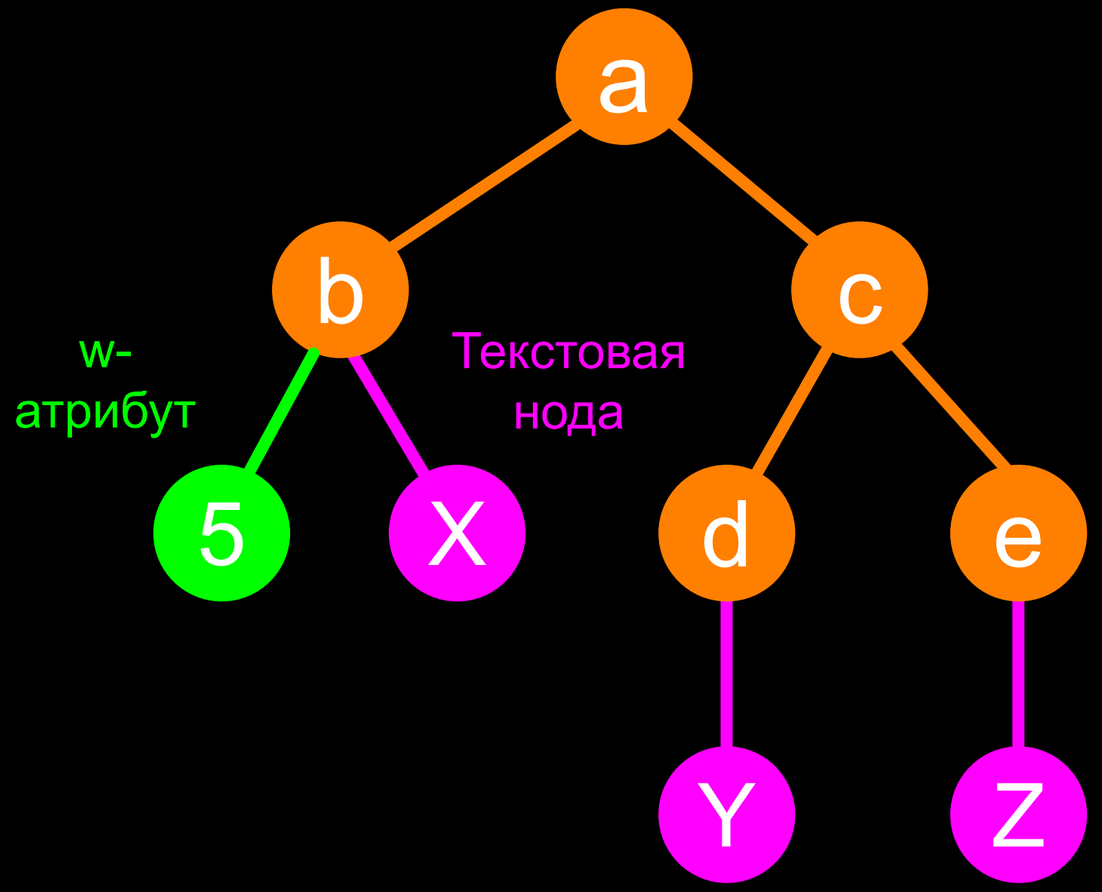
Элементы Текст



Текст и атрибуты в XML

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

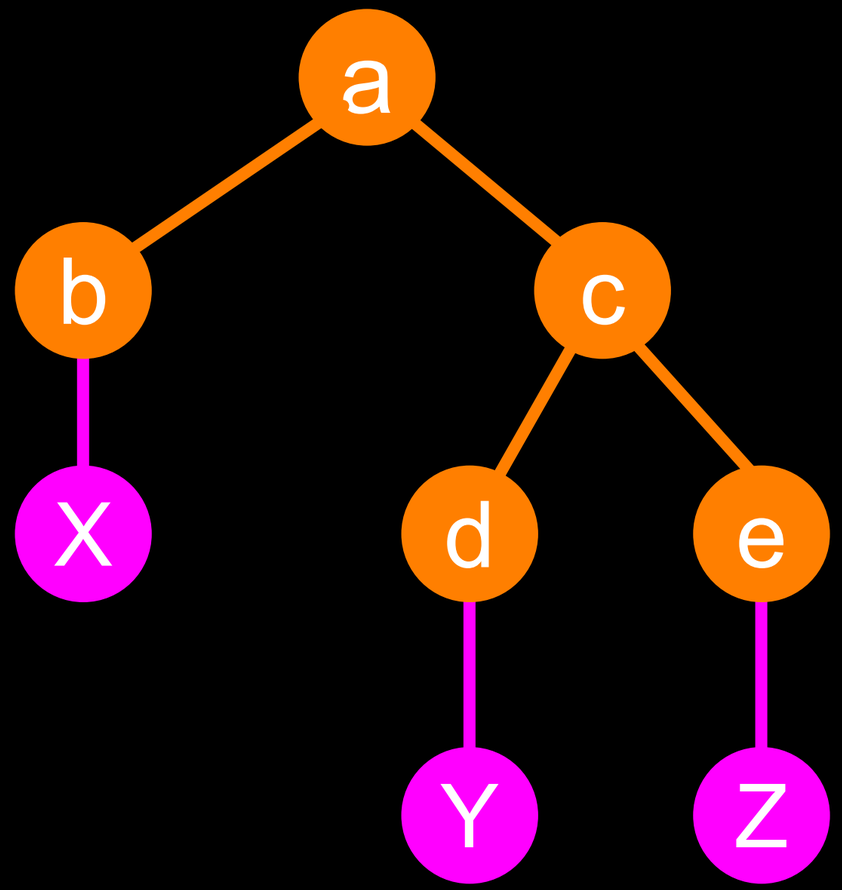
Элементы Текст



XML в виде путей

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

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



Элементы Текст

XML-схема

Описание правил, которым должен подчиняться документ

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

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

XML-схема

- Язык описания структуры XML-документа
- Налагает ограничения на структуру и содержимое документа
- Часто используется для установления «контракта» между системами: «Моя система будет принимать XML, который соответствует этой конкретной Схеме»
- Если конкретный фрагмент XML соответствует Схеме, он считается «подтвержденным»

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

Валидация XML

XML-документ

Договор об
XML-схеме

Валидация XML

XML-документ

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

Договор об XML-схеме

```
<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>
```

Множество языков XML-схемы

- Определение типа документа (англ. DTD)

  - https://ru.wikipedia.org/wiki/DTD

- Стандартный обобщенный язык разметки (ISO 8879:1986 SGML)

  - https://ru.wikipedia.org/wiki/SGML

- XML-схема (W3C) - (.XSD - расширение файла)

  - https://ru.wikipedia.org/wiki/XML_Schema_(W3C)

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

XSD XML-схема (W3C)

- Мы сосредоточимся на версии Консорциума Всемирной паутины (англ.W3C)
- Ее часто называют «W3C-схема», так как «Схема» считается универсальной
- Частенько ее называют XSD, так как файл имеет расширение .xsd

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

https://ru.wikipedia.org/wiki/XML_Schema_(W3C)

Структура XSD

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

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

```
<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-ограничения

```
<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>Ivan Petrov</full_name>
  <child_name>Ivan</child_name>
  <child_name>Vanya</child_name>
  <child_name>Vano</child_name>
  <child_name>Ivanko</child_name>
</person>
```

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

Типы данных XSD

```
<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"/>
```

Время обычно отображается в формате UTC/GMT, учитывая, что серверы часто разбросаны по всему миру

```
<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 Формат Дата/Время

2002-05-30T09:30:10Z

Год-месяц-день

Время

Часовой пояс обычно указывается в формате UTC / GMT, а не в формате местного часового пояса

https://ru.wikipedia.org/wiki/ISO_8601

https://ru.wikipedia.org/wiki/Всемирное_координированное_время

```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
<?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

XMLDemo_1.py

```python
import xml.etree.ElementTree as ET

data = '''
<student>
  <name>Alex</name>
  <phone type="mobile">
    +375293452290
  </phone>
  <email hide="yes" />
</student>'''

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

XMLDemo_2.py

```python
import xml.etree.ElementTree as ET

input = '''
<people>
  <users>
    <user level="admin">
      <id>001</id>
      <name>Mike</name>
    </user>
    <user level="guest">
      <id>009</id>
      <name>Vlad</name>
    </user>
  </users>
</people>'''
```

```python
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('level’))
```

XMLDemo_3.py

Задача. Найти все блюда, содержащие кальций. Данные о блюдах представлены в XML-формате

```
<nutrition>
<food>
<name>Avocado
Dip</name>
   <mfr>Sunnydale</mfr>
   <serving
units="g">29</serving>
   <calories total="110"
fat="100"/>

<total-fat>11</total-fat
>

<saturated-fat>3</satura
ted-fat>

<cholesterol>5</choleste
rol>
   <sodium>210</sodium>
   <carb>2</carb>
<vitamins>
      <a>0</a>
      <c>0</c>
   </vitamins>
   <minerals>
      <ca>0</ca>
      <fe>0</fe>
   </minerals>
</food>
```

Задача. Найти все блюда, содержащие кальций. Данные о блюдах представлены в XML-формате

XMLDemo_3.py

```
import
xml.etree.ElementTree as
ET
file = open("food.xml",
"r")
content = file.read()
print(content)
print("Блюда, содержащие кальций:")
for item in lst:
    ca = item.find('minerals/ca').text
    if item.find('minerals/ca').text ≠
'0':
        print("Название: ",
item.find("name").text, end="")
        print("Калорийность: ",
item.find("calories").get("total"))
```

Текстовый формат обмена данными (JSON)

Литеральная нотация объектов в JavaScript

- Формат JSON был разработан Дугласом Крокфордом
- Литеральная нотация объектов в JavaScript

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

Введение в JSON

العربية Български 中文 Český Dansk Nederlands English Esperanto Français Deutsch Ελληνικά עברית Magyar Indonesia
Italiano 日本 한국어 فارسی Polski Português Română Русский Српско-хрватски Slovenščina Español Svenska Türkçe Tiếng Việt

ECMA-404 The JSON Data Interchange Standard.

JSON (JavaScript Object Notation) - простой формат обмена данными, удобный для чтения и написания как человеком, так и компьютером. Он основан на подмножестве языка программирования JavaScript, определенного в стандарте ECMA-262 3rd Edition - December 1999. JSON - текстовый формат, полностью независимый от языка реализации, но он использует соглашения, знакомые программистам C-подобных языков, таких как C, C++, C#, Java, JavaScript, Perl, Python и многих других. Эти свойства делают JSON идеальным языком обмена данными.

JSON основан на двух структурах данных:

> Коллекция пар ключ/значение. В разных языках, эта концепция реализована как *объект*, запись, структура, словарь, хэш, именованный список или ассоциативный массив.
> Упорядоченный список значений. В большинстве языков это реализовано как *массив*, вектор, список или последовательность.

Это универсальные структуры данных. Почти все современные языки программирования поддерживают их в какой-либо форме. Логично предположить, что формат данных, независимый от языка программирования, должен быть основан на этих структурах.

В нотации JSON это выглядит так:

```
json
    element

value
    object
    array
    string
    number
    "true"
    "false"
    "null"

object
    '{' ws '}'
    '{' members '}'
```

```
import json
data = '''{
  "name" : "Alex",
  "phone" : {
    "type" : "mob",
    "number" : "+376 29
3287765"
   },
   "email" : {
     "hide" : "yes"
   }
}'''

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

JSONDemo_1.py

Формат JSON представляет данные в виде вложенных «списков» и «словарей»

```
  { "id" : "001",
    "x" : "2",
    "name" : "Alex"
  } ,
  { "id" : "009",
    "x" : "7",
    "name" : "Vlad"
  }
]'''

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

JSONDemo_2.py

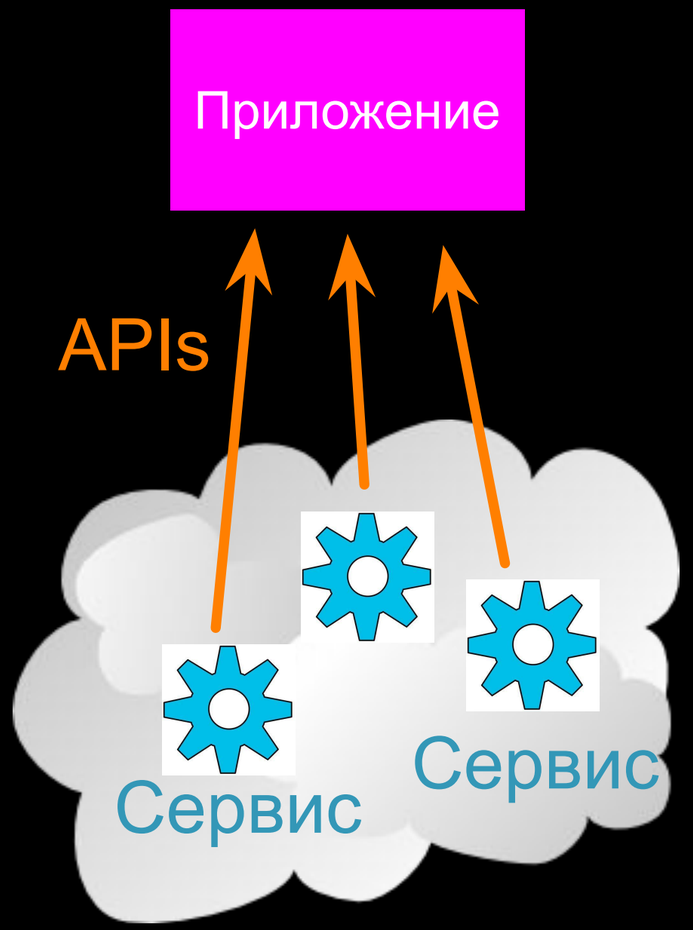
Формат JSON представляет данные в виде вложенных «списков» и «словарей»

Сервис-ориентированный подход

https://ru.wikipedia.org/wiki/Сервис-ориентированная_архитектура

Сервис-ориентированный подход

- Большинство приложений используют сервисы
- Используются сервисы из других приложений:
  - Сервис оплаты по кредитной карте;
  - Системы бронирования отелей.
- Сервисы публикуют «правила», которым приложения должны следовать, чтобы использовать данный сервис (API)



Программный интерфейс приложения (API)

API определяет интерфейс и поведение объектов в этом интерфейсе, но при этом позволяет абстрагироваться от того, как именно эта функциональность будет реализована. Программное обеспечение, которое предоставляет функциональность, описываемую в API, называется «реализация» API. Обычно API определяется в терминах языка программирования, используемого при создании приложения

https://ru.wikipedia.org/wiki/API

Getting Started | Google Map

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

API-безопасность и ограничение на количество обращений

- Вычислительные ресурсы для запуска этих API являются платными
- Обычно данные, предоставляемые этими API, имеют ценность
- Поставщики данных могут ограничивать количество запросов в день, запрашивать API-ключ, а также взимать плату за использование
- Время от времени правила могут меняться

Usage Limits Ограничения по использованию

The Google Geocoding API has the following limits in place:

- 2,500 requests per day. 2,500 запросов в день

Google Maps API for Business customers have higher limits:

- 100,000 requests per day. Не более 100,000 запросов в день для бизнес-клиентов

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

Задача. Узнать погоду в городе и запросит прогноз на ближайшие дни, используя API сервиса OpenMeatherMap.org

```
import requests
# id, полученный при регистрации на OpenWeatherMap.org.
appid = "908f7745f7066f579da091f54d6bac78"


# Проверка наличия в базе информации о нужном населенном пункте
def get_city_id(s_city_name):
    res =
requests.get("http://api.openweathermap.org/data/2.5/find",
                         params={'q': s_city_name, 'type': 'like',
'units': 'metric', 'lang': 'ru', 'APPID': appid})
    data = res.json()
    cities = ["{} ({})".format(d['name'], d['sys']['country'])
for d in data['list']]
    print("city:", cities)
    city_id = data['list'][0]['id']
    print('city_id=', city_id)
    return city_id
```

Задача. Узнать погоду в городе и запросит прогноз на ближайшие дни, используя API сервиса OpenWeatherMap.org

```
# Запрос текущей погоды
def request_current_weather(city_id):
    res =
requests.get("http://api.openweathermap.org/data/2.5
/weather",
                         params={'id': city_id,
'units': 'metric', 'lang': 'ru', 'APPID': appid})
    data = res.json()
    print("Погоденые условия:",
data['weather'][0]['description'])
    print("Температуры:", data['main']['temp'])
    print("Min:", data['main']['temp_min'])
    print("Max:", data['main']['temp_max'])
    print("Влажность:", data['main']['humidity'])
    print("Все данные о погоде на сегодня")
    print("data:", data)
```

Задача. Узнать погоду в городе и запросит прогноз на ближайшие дни, используя API сервиса OpenMeatherMap.org

```
# Запрос прогноза погоды на ближайшее время
def request_forecast(city_id):
    res =
requests.get("http://api.openweathermap.org/data/2.5
/forecast",
                         params={'id': city_id,
'units': 'metric', 'lang': 'ru', 'APPID': appid})
    data = res.json()
    print('city:', data['city']['name'],
data['city']['country'])
    for i in data['list']:
        print((i['dt_txt'])[:16],
'{0:+3.0f}'.format(i['main']['temp']),
              '{0:2.0f}'.format(i['wind']['speed'])
+ " м/с",
              i['weather'][0]['description'])
```

Задача. Узнать погоду в городе и запросит прогноз на ближайшие дни, используя API сервиса OpenMeatherMap.org

```
s_city_name = "Барселона"
city_id = get_city_id(s_city_name)
print("Погода на сегодня в",
s_city_name)
request_current_weather(city_id)
print("Прогноз на ближайшие дни в ",
s_city_name)
request_forecast(city_id)
```