Использование Web сервисов

Глава 13

Python for Informatics: Exploring Information
www.pythonlearn.com

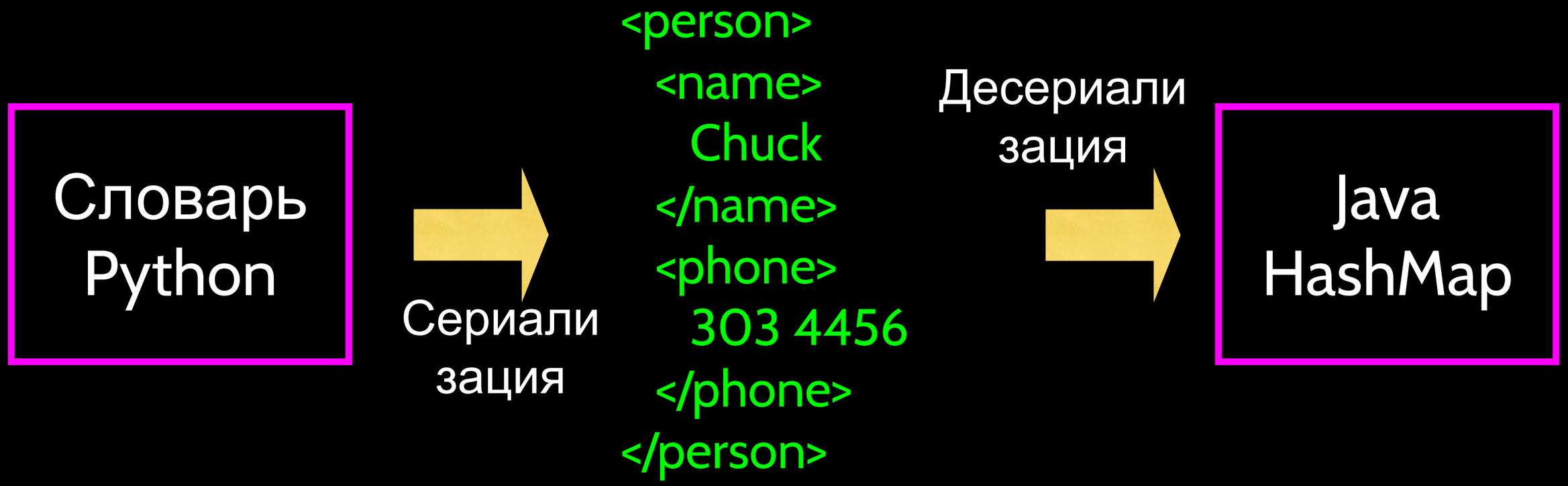
Данные в Web

- С распространением и широкой поддержкой протокола HTTP Запрос/Ответ, появилось естественное стремление обмениваться данными между программами, используя эти протоколы
- Поэтому необходимо было придумать и согласовать способ представления данных, передаваемых по сети между приложениями
- Существует два широко распространённых формата: XML и JSON

Отправка данных через “Сеть”

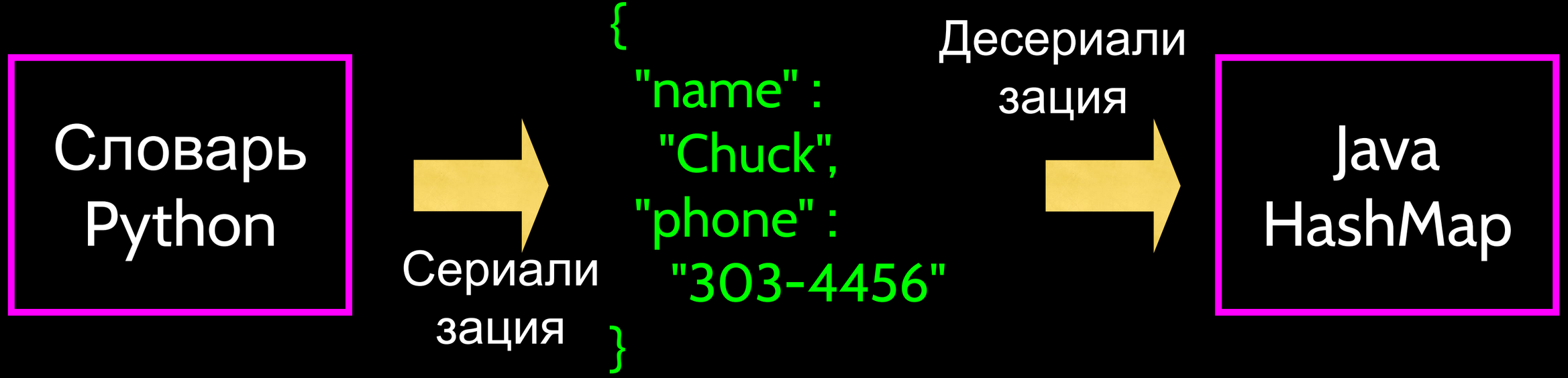
a.k.a. “Wire Protocol” – что посылают “по проводам”

Соглашение о протоколе связи



XML

Соглашение о протоколе связи



JSON

Элементы (или вершины) XML

- Простой элемент
- Составной элемент

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

XML

Разметка данных для передачи через сеть...

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

eXtensible Markup Language (Расширяемый язык разметки)

- Основная цель – помочь информационным системам делиться структурированными данными

- Создавался как упрощённое подмножество стандартного обобщённого языка разметки (Standard Generalized Markup Language), и проектировался, чтобы быть относительно легко понятным для человека

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

Основы XML

- Тег начала
- Тег конца
- Текстовое содержание
- Атрибут
- Самозакрывающийся тег

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

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

Пробелы

Не важно, где кончается строка.
Отступы нужны только для
читаемости.

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

Пример XML...

```
<recipe name="bread" prep_time="5 mins" cook_time="3 hours">
  <title>Basic bread</title>
  <ingredient amount="8" unit="dL">Flour</ingredient>
  <ingredient amount="10" unit="grams">Yeast</ingredient>
  <ingredient amount="4" unit="dL" state="warm">Water</ingredient>
  <ingredient amount="1" unit="teaspoon">Salt</ingredient>
  <instructions>
    <step>Mix all ingredients together.</step>
    <step>Knead thoroughly.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Knead again.</step>
    <step>Place in a bread baking tin.</step>
    <step>Cover with a cloth, and leave for one hour in warm room.</step>
    <step>Bake in the oven at 180(degrees)C for 30 minutes.</step>
  </instructions>
</recipe>
```

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

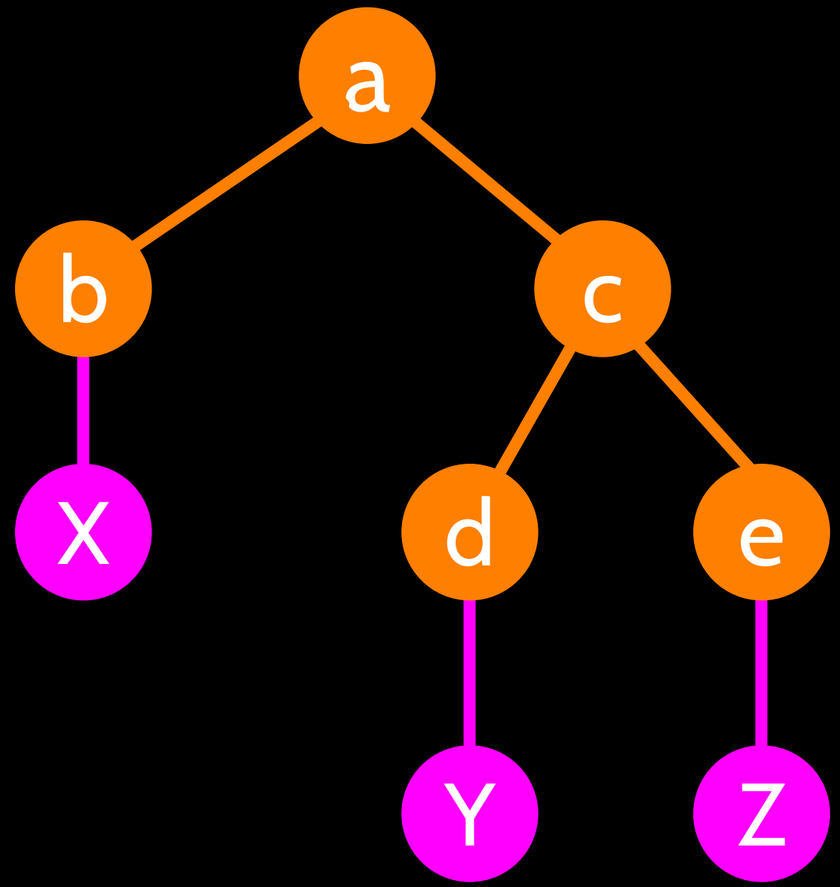
Терминология XML

- Теги показывают начало и конец элементов
- Атрибуты – пары ключ/значение в открывающемся теге XML
- Сериализация / Десериализация – преобразование данных конкретной программы в общепринятый формат, который можно хранить и/или передавать между системами независимо от используемого языка программирования

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

XML как дерево

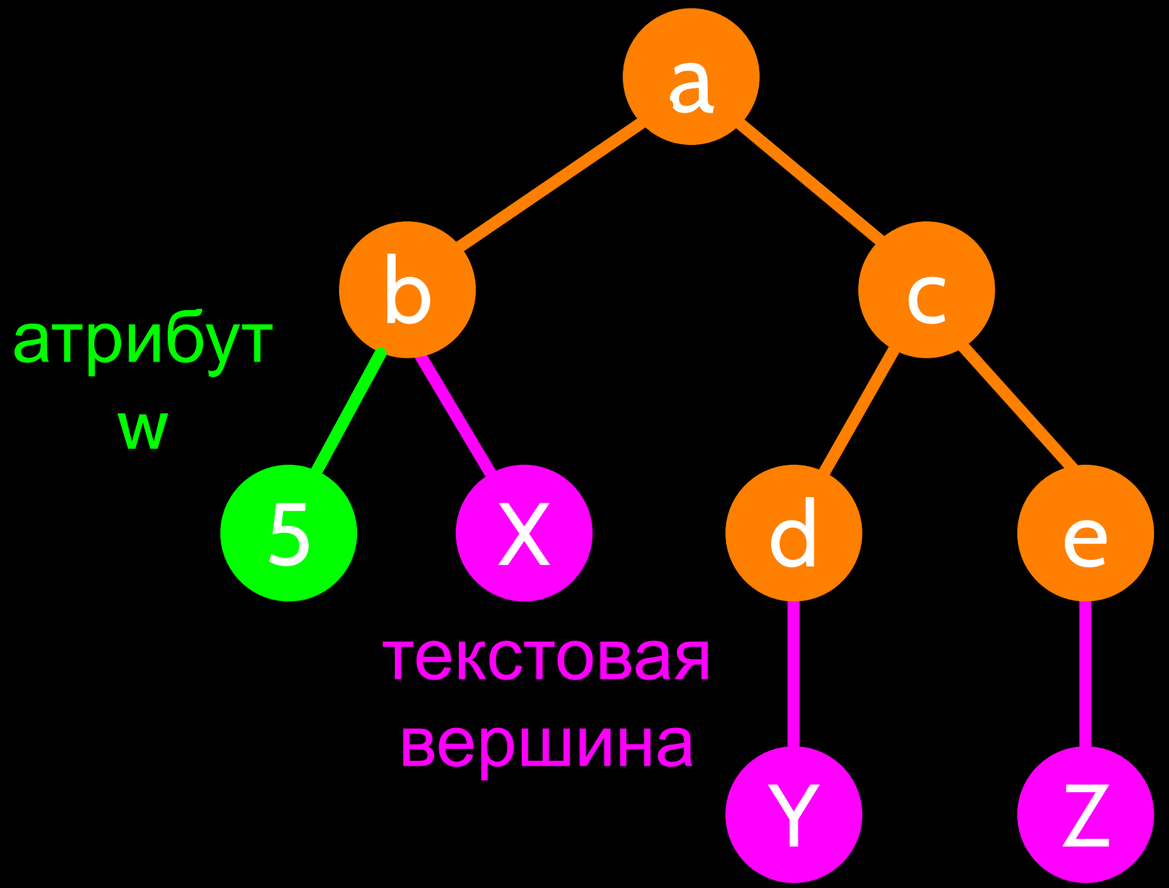
```
<a>
  <b>X</b>
   <c>
     <d>Y</d>
     <e>Z</e>
   </c>
</a>
```



Элементы Текст

Текст и атрибуты XML

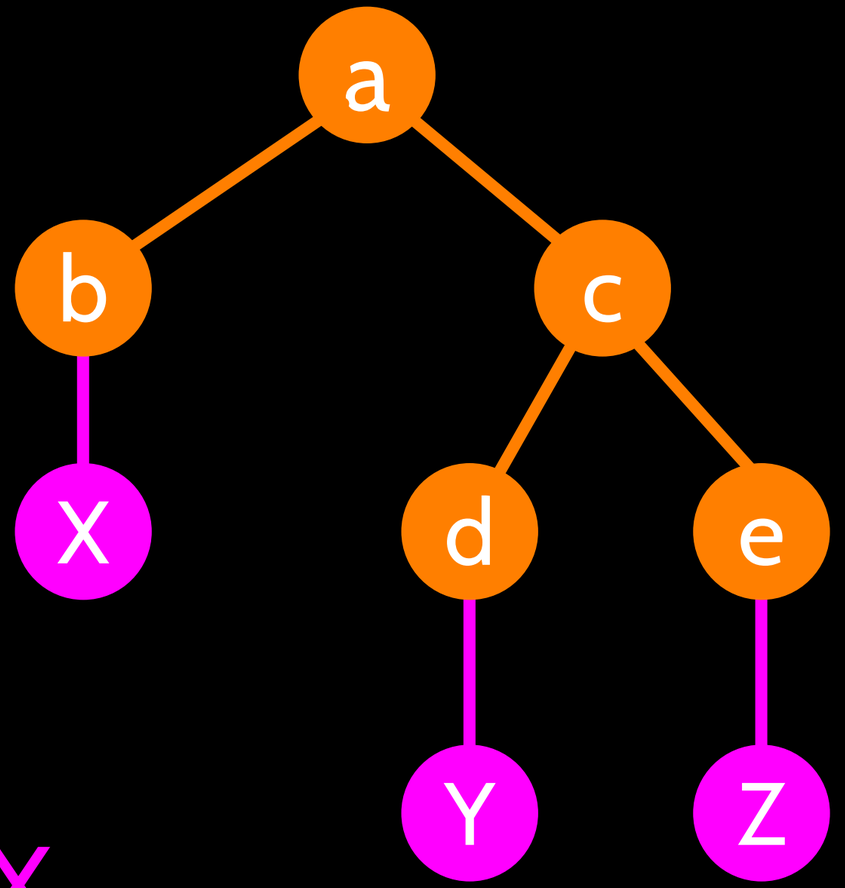
```
<a>
 <b w="5">X</b>
 <c>
  <d>Y</d>
  <e>Z</e>
 </c>
</a>
```

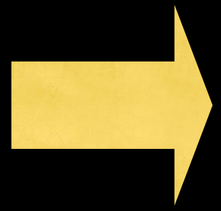


Элементы Текст

XML как пути

```
<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 Schema

Описание “соглашения” о том, что является допустимым XML.

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

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

XML Schema

- Описание допустимого формата документа XML
- Выражается в ограничениях на структуру и содержание документа
- Часто используется для указания “соглашения” между системами - “Моя система будет принимать только XML соответствующие определённой схеме.”
- Если определённый XML удовлетворяет требованиям схемы, он называется “валидным”

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

Валидация XML

Документ XML

Соглашение XML Schema

Валидатор

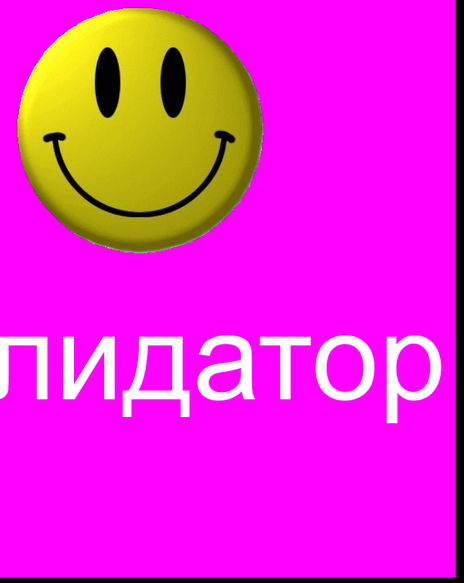
Валидация XML

Документ
XML

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

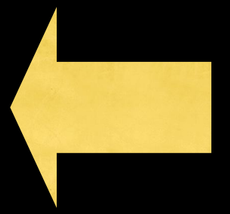
Соглашение
XML Schema

```
<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 Schema

- Document Type Definition (DTD)
  - https://ru.wikipedia.org/wiki/DTD
- Standard Generalized Markup Language (ISO 8879:1986 SGML)
  - https://ru.wikipedia.org/wiki/SGML
- XML Schema from W3C - (XSD)
  - https://ru.wikipedia.org/wiki/XML_Schema



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

XSD XML Schema (спецификация W3C)

- Мы сфокусируемся на версии World Wide Web Consortium (W3C)
- Её часто называют “W3C Schema”
- Более широко она известна как XSD, поскольку файлы имеют расширение .xsd

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

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

Структура XSD

- xs:element - элемент
- xs:sequence - последовательнсть
- xs:complexType – составной элемент

```
<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>
```

Ограничения XSD

```
<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

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"/>
```

Обычно время представляют в формате UTC/GMT, поскольку сервера 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

Формат дат и времени
ISO 8601

2002-05-30T09:30:10Z

Год-месяц-день

Время суток

Часовой пояс – обычно указывается в UTC / GMT

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

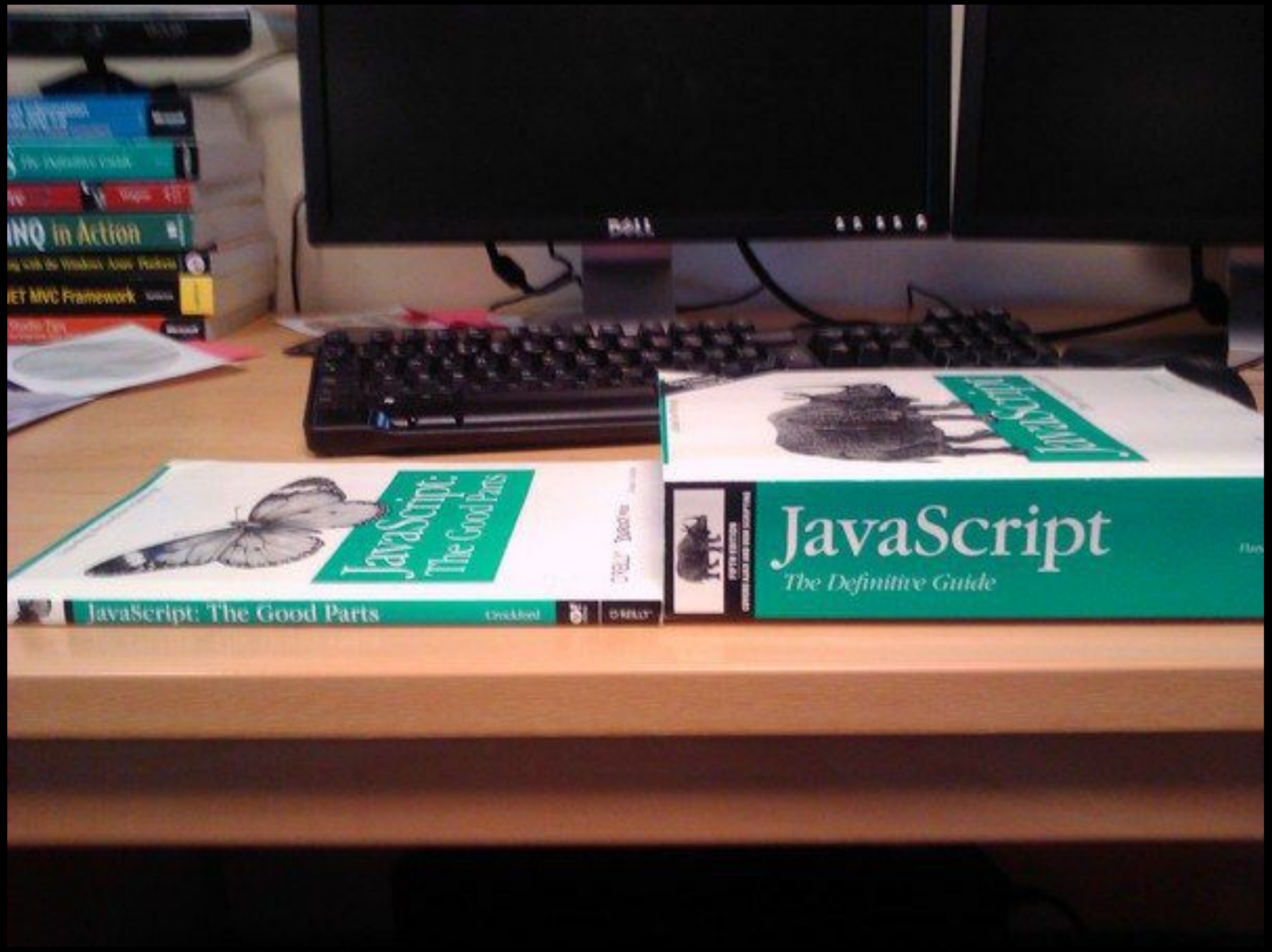
```
import xml.etree.ElementTree as ET
input = '''<stuff>
    <users>
        <user x="2">
            <id>001</id>
            <name>Chuck</name>
        </user>
        <user x="7">
            <id>009</id>
            <name>Brent</name>
        </user>
    </users>
</stuff>'''

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

- Дуглас Крокфорд - “открыл” JSON
- Нотация литералов объектов в JavaScript

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


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

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 представляет данные в виде вложенных “списков” и “словарей”

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 представляет данные в виде вложенных “списков” и “словарей”

Acknowledgements / Contributions

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

… Insert new Contributors here

...