
adtelligent

{JSON}
&
<XML />

adtelligent

Introduction

- Что такое JSON?
- Синтаксис
- Типы данных
- Использование

adtelligent

Introduction

- Что такое JSON?
- Синтаксис
- Типы данных
- Использование

- Что такое XML?
- Синтаксис
- Типы данных
- Использование

adtelligent

JSON: Overview

- Текстовый формат описания данных
- Programming Language Independent
- Подмножество JavaScript
- Прост для чтения/понимания/генерации
- MIME Type: application/json
- File Extension: .json

JSON: Overview

- Текстовый формат описания данных
- Programming Language Independent
- Подмножество JavaScript
- Прост для чтения/понимания/генерации

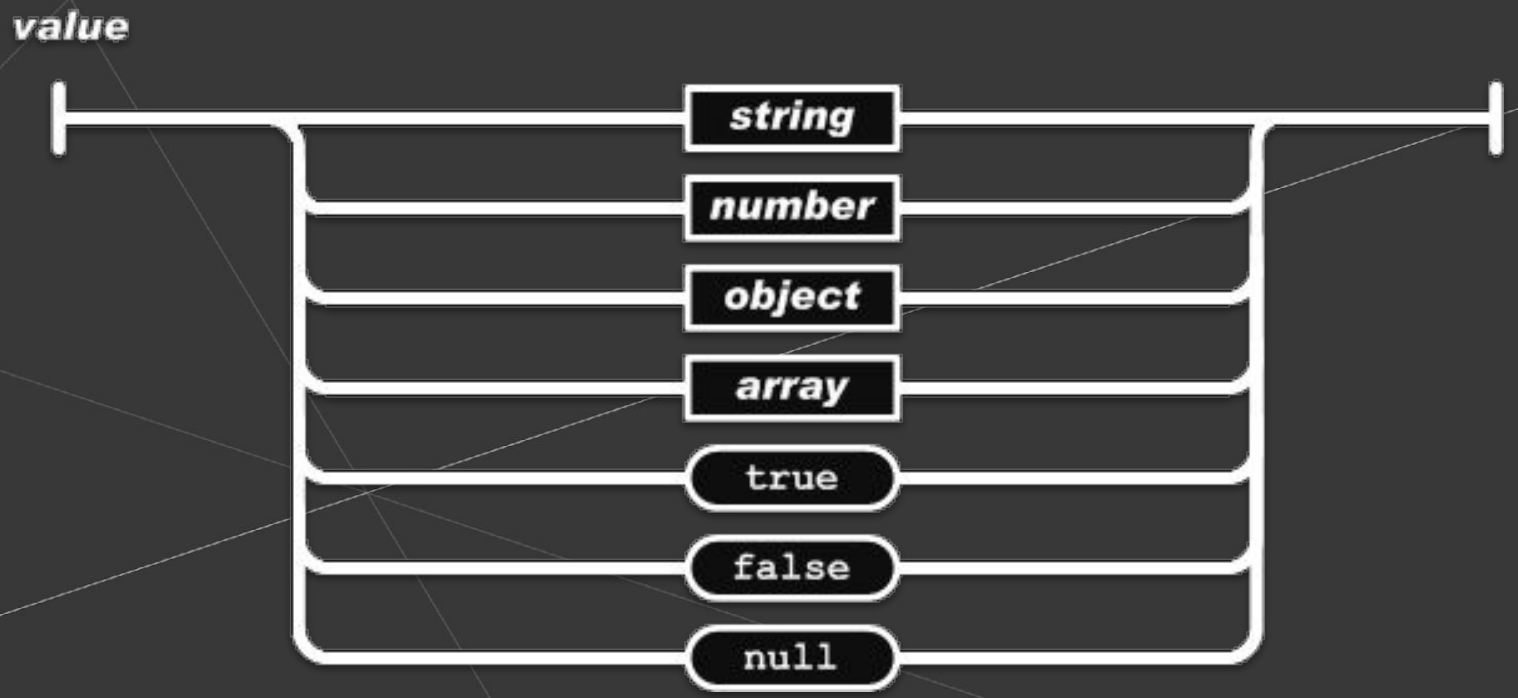
- JSON is NOT:
  - Overly complex
  - Не есть формат описания документа
  - Не является языком разметки
  - Не является языком программирования

JSON: Syntax

Синтаксис

adtelligent

JSON: Syntax

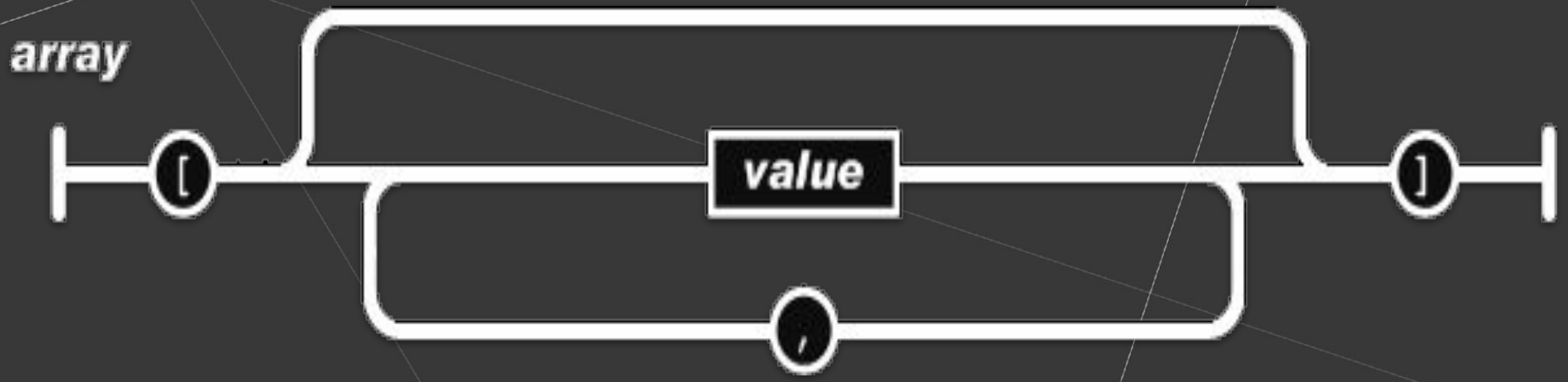



```
{
   “string” : “Some String value”,
   “number”: 5.2173e-12,
   “object” : {
                  “kind” : “object”,
                  “name”: “value”
               },
    “array” : [ 40, 29, “11:12” ],
    “boolean” : true,
    “nullable” : null
}
```

adtelligent

JSON: Array Syntax

- **Массив - упорядоченный набор значений**
- **Начинается с квадратной скобки “[“**
- **И квадратной скобкой заканчивается “]”**
- **Элементы массива разделяются запятой “,”**



adtelligent

JSON: Array Syntax

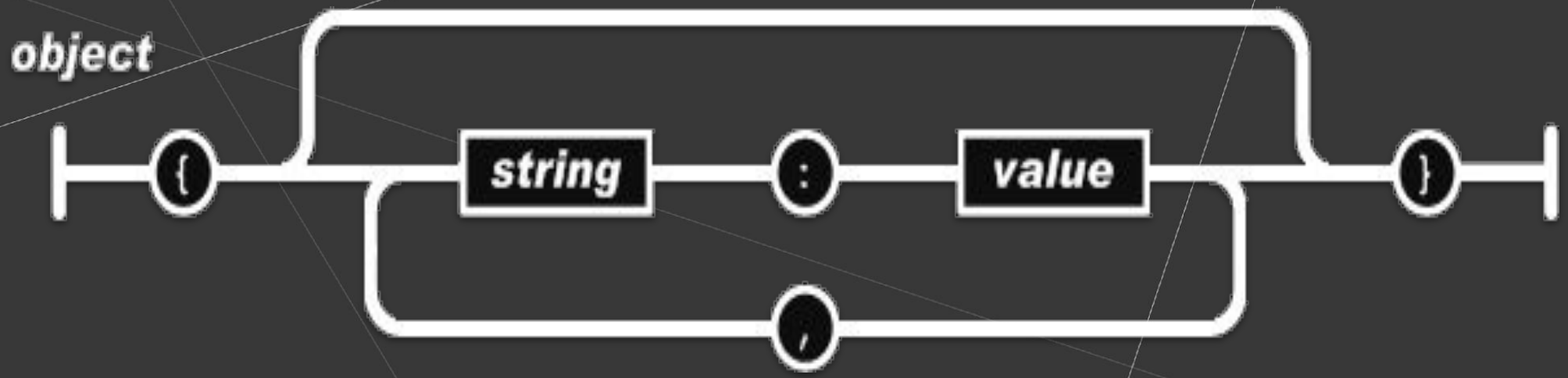
- Массив - упорядоченный набор значений
- Начинается с квадратной скобки “[“
- И квадратной скобкой заканчивается “]”
- Элементы массива разделяются запятой “,”

Пример:

[“яйца”, “хлеб”, “молоко”, “ванишь”, “тб”, “бп”, “кж”, “кс”, “что то вкусненькое”]

JSON: Object Syntax

- Объект - неупорядоченный набор пар ключ/значение
- Фигурные скобки определяют объект
- Ключ - это строка
- После имени ключа идет двоеточие “:”
- Пары ключ/значение разделяются запятой “,”



adtelligent

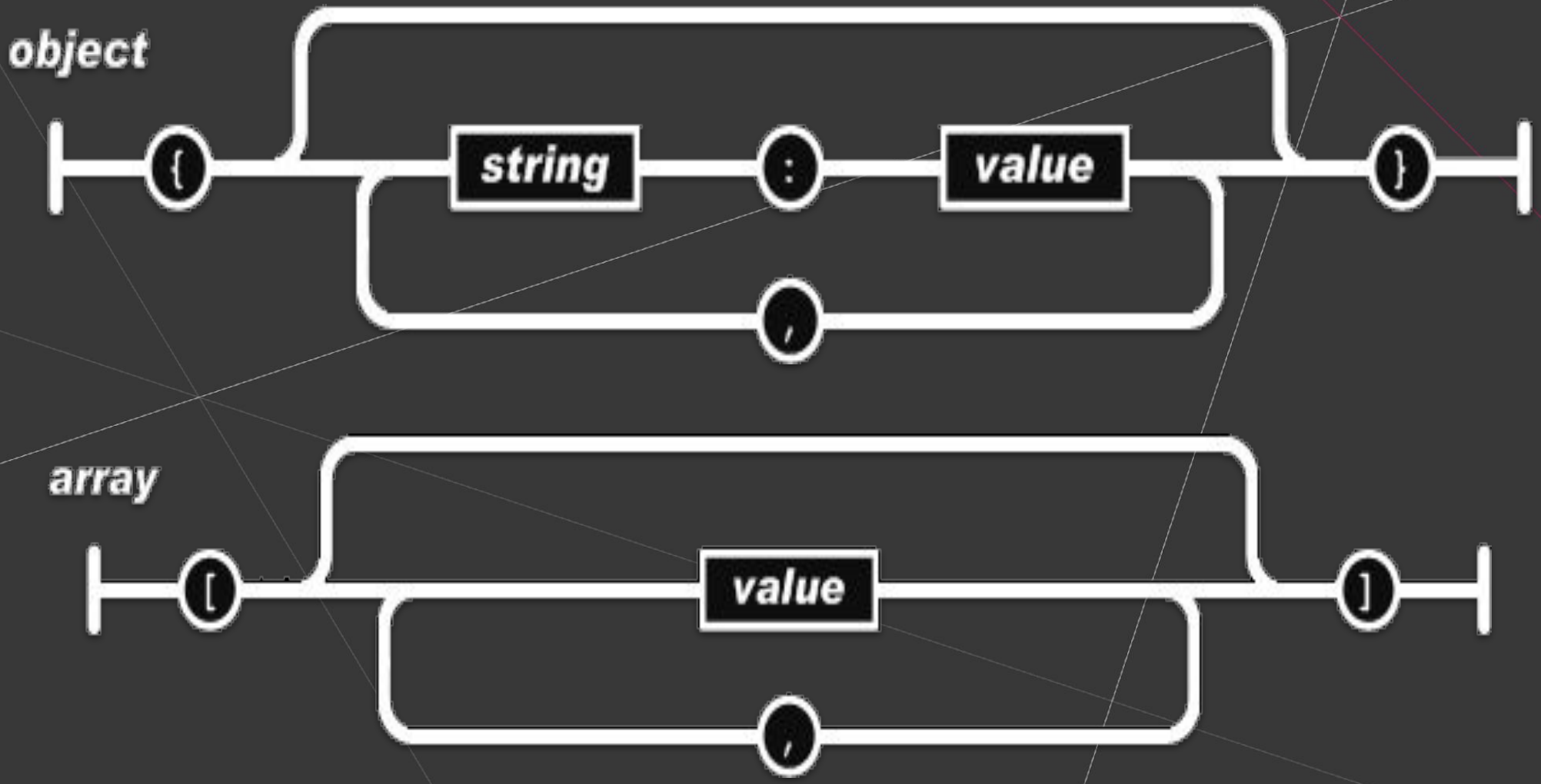
JSON: Object Syntax

- Объект - неупорядоченный набор пар ключ/значение
- Фигурные скобки определяют объект
- Ключ - это строка
- После имени ключа идет двоеточие “:”,
- Пары ключ/значение разделяются запятой “,”

Пример:

```
{
    “alterego”: “Wonder Woman”,
    “species” : “Amazonian”,
    “team”: “Justice League”,
    “aliases” : [ “Princess Diana of Themyscira”, “Diana Princ” ],
    “eyes color” : “celestial light-blue”,
    “height” : 182,
    “weight” : 59
    “hair” : “black”,
    “perks” : [ “immortal”, “healer”, “flying”, “melee”, “limited invulnerability” ]
}
```

JSON: Syntax



adtelligent

JSON: Syntax

- Strings
  - Обрамляются двойными кавычками “
  - Исключительно UNICODE/UTF-8
  - Все что не UNICODE должно быть превращено в UNICODE/UTF-8

JSON: Syntax

- Strings
  - Обрамляются двойными кавычками “
  - Исключительно UNICODE/UTF-8
  - Все что не UNICODE должно быть превращено в UNICODE/UTF-8
  - Двойные кавычки нельзя. Их надо “эскейпить”:

```
{
    “name” : “Jack \”Captain\” Sparrow”
}
```

JSON: Syntax

- Strings
  - Обрамляются двойными кавычками “
  - Исключительно UNICODE/UTF-8
  - Все что не UNICODE должно быть превращено в UNICODE/UTF-8
  - Двойные кавычки нельзя. Их надо “эскейпить”:

```
{
    “name” : “Jack \”Captain\” Sparrow”
}
```

  - Кириллицу тоже нельзя:

```
{
    “reaction” : “\u042B\u044B\u044B\u044B\u044B....”        // Ыыыыы....
}
```

adtelligent

JSON: Syntax

- Strings
  - Обрамляются двойными кавычками “
  - Исключительно UNICODE/UTF-8
  - Все что не UNICODE должно быть превращено в UNICODE/UTF-8
  - Двойные кавычки нельзя. Их надо “эскейпить”:

```
{
    “name” : “Jack \”Captain\” Sparrow”
}
```

  - Кириллицу тоже нельзя:

```
{
    “reaction” : “\u042B\u044B\u044B\u044B\u044B....”        // Ыыыыы....
}
```

  - Обратный слеш “\\” или “\u005C”
  - Перевод строки “\r”
  - Конец строки “\n”
  - Кавычка “\”” или “\u0022”

adtelligent

JSON: Usage

- Data exchange

```
# curl https://auth-service.adtelligent.com/api/v1/auth
      -H ‘Content-Type: application/json’ -H ‘Accept: application/json”
     --data-binary '{"username":"hbdemo@verta.media","password":"qwerty123"}'
```

Response Body:

```
{
  "success":true,
  "data": {
            "sessionData": {
               "sid":"session_b7a18cb7-8521-483e-bb24-f862ec6f0a8a",
               "uid":271020
             },
            "username":"hbdemo@verta.media",
            "type":"publisher",
            "roles": ["outsream_publisher"]
  }
}
```

adtelligent

JSON: Usage

- JavaScript Applications:
  - JSON.stringify(jsObject);
  - JSON.parse(String);
  - eval(jsonstring);

- Databases

```
> db.users.find( {} )
  {
    "_id" : "5a593553a943c763486d0dd5",
    "created_at" : "2018-01-12T22:23:15.703Z",
    "email" : "test@test.com"
  }
```

- Whatever... тысячи их

adtelligent

JSON: Что плохо

- Текстовый. Не совместим с бинарными данными
- Нет возможности поточной обработки
- Не умеет в комментарии
- Не умеет в букву “Ы” (и в “الله أكبر” тоже)
- Не умеет в ссылки
- Прост, популярен и поддерживается практически всеми ЯП

adtelligent

XML: Overview

- eXtensible Markup Language
- Plaintext Format
- “self-descriptive” (human readable)
- Иерархичен
- Создан для передачи, описания структуры и хранения данных
- Сам по себе XML не делает ничего
- Meta-Language.

XML: Document

- Text-Based :
  - Платформо-независим
  - Читабелен
  - Со всеми вытекающими проблемами

adtelligent

XML: Meta Language

- Есть тысячи XML языков, созданных для разных целей
- У всех общие “xml-правила”
- XHTML, OOXML, Open Document, RSS, SVG, SOAP, XSLT, MathML...
- https://en.wikipedia.org/wiki/List_of_XML_markup_languages

XML: Examples

XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
  <head>
     <title>Title of document</title>
  </head>
  <body>
      some content
  </body>
</html>
```

adtelligent

XML: Examples

SVG

```
<?xml version=”1.0”?>
<!DOCTYPE svg PUBLIC “-//W3C//DTD SVG 1.1//EN”
“http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd”>
>
<svg height="100" width="100" version=”1.1” xmlns=”http://www.w3.org/2000/svg”>
  <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="blue" />
</svg>
```

adtelligent

XML: Examples

VAST DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
  <xs:element name="VAST">
    <xs:annotation>
      <xs:documentation>IAB VAST, Video Ad Serving Template, video xml ad response,
Version 3.0.0, xml schema prepared by Google</xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="Ad" maxOccurs="unbounded" minOccurs="0">
          <xs:annotation>
            <xs:documentation>Top-level element, wraps each ad in the
response</xs:documentation>
...
```

adtelligent

XML: Correctness

- Синтаксически верен
- Валиден : удовлетворяет правилам семантики

adtelligent

XML: Syntax

```
<?xml version=”1.0” encoding=”utf-8” standalone=”yes” ?>
<presentation>
    <slide number=”28”>
        <name> XML : Syntax </name>
        <notes>
            <note lang=”en”>
                Take a look at example xml document representing current slide
            </note>
            <note lang=”ru”>
                А на краяне мужыки сейчас водку пьют.
                Я вообще хотел стать орнитологом, какой xml?
            </note>
        </notes>
    </slide>
</presentation>
```

adtelligent

XML: Syntax

<?xml version=”1.0” encoding=”utf-8” standalone=”yes” ?>

XML Declaration.
Была когда то опциональна (в 1.0). Обязательна в 1.1

version: “1.0” или “1.1”

encoding: по умолчанию “utf-8”.

standalone: определяет наличие ссылок на внешние документы или определения

XML: Syntax

Element vs Tag vs Attribute

- Element:

```
<note lang=”ru”>
    Здесь надо что то рассказать чтобы было нескучно
</note>
```

- Start tag: <note>
- End tag </note>
- Content: “Здесь надо что то рассказать чтобы было нескучно”
- Стартовый тег может содержать атрибуты:
  - lang=”ru”

XML: Syntax

Element Rules

- Только один root элемент
- За каждым открывающим тег должен следовать закрывающий
- Контент опционален :
  - <meaning-of-life></meaning-of-life> <!-- same as : -->
  - <meaning-of-life />
- Имена тегов - регистро-зависимы
  - <big-black-suv></big-black-SUV> <!-- ERROR -->

- Порядок важен:
  - <negation>
    <depression>
    <humility>
    </depression> <!-- problem here -->
    </humility>
    </negation>

adtelligent

XML: Syntax

Attribute Rules

- Только в открывающем теге
- Значения должны быть в кавычках
  - <actress name=”Anjelica Boyd” />
  - <actress name=’Anjelica “Krystal” Boyd’ />
  - <actress name=”Anjelica "Krystal" Boyd” />

XML: Syntax

Tag Naming Conventions

- Имена могут содержать буковы, цифры и немного других символов
- Не могут начинаться с цифры или знака пунктуации
- Не могут начинаться со слова “xml”.
- Не могут содержать пробелы

adtelligent

XML: Syntax

- Комментарии
  <!-- This is some comment for XML -->

- Неоднозначность разметки
  < <
  > >
  & &
  ‘ '
  “ "

  <!-- Джонни Смит <nagibat0r@gmail.com> -->
  <email-address>John Smith<nagibat0r@gmail.com></email-address>

XML: Syntax

- Комментарии
  <!-- This is some comment for XML -->

- Неоднозначность разметки
  < <
  > >
  & &
  ‘ '
  “ "

  <!-- Джонни Смит <nagibat0r@gmail.com> -->
  <email-address>John Smith<nagibat0r@gmail.com></email-address>

- CDATA-секции. Начинаются с <![CDATA[, закрываются]]>
  <email-address>
  <![CDATA[John Smith<nagibat0r@gmail.com>]]>
  </email-address>

adtelligent

XML: Valid Document

Валидность XML документа

adtelligent

XML: Valid Document

Валидность XML документа

- **Синтаксически корректный**
- **Следует неким семантическим правилам**

- **Обычно XML документ связан с внешним файлом, описывающим семантические правила для документа (.dtd или .xsd)**
- **Семантические правила? WTF?**
  - **Допустимые имена тегов**
  - **Порядок элементов**
  - **Обязательные атрибуты**
  - **Количество child-элементов**
  - **Значения по умолчанию для атрибутов**
  - **типы данных**
  - **...**

adtelligent

XML: VAST XSD Example

```
<?xml version="1.0" encoding="utf-8" ?>
<xs:schema  ...
...
<xs:complexType name="TrackingEvents_type" id="vtype.ad.creative.trackingevents">
        <xs:sequence><xs:element name="Tracking"  minOccurs="0">
            <xs:complexType>
                <xs:simpleContent>
                    <xs:extension base="xs:anyURI">
                        <xs:attribute name="event"  use="required">
                            <xs:simpleType>
                                <xs:restriction base="xs:token">
                <xs:enumeration value="mute" />
                <xs:enumeration value="unmute" />
                <xs:enumeration value="pause" />
                <xs:enumeration value="resume" />
                <xs:enumeration value="rewind" />
                <xs:enumeration value="skip" />
                <xs:enumeration value="playerExpand" />
```

adtelligent

XML: VAST XML Example

```
<?xml version="1.0" encoding="utf-8" ?>
<VAST version="2.0">
  <Ad id="static"> <InLine>
     <AdSystem>Static VAST Template</AdSystem>
     <AdTitle>Static VAST Tag</AdTitle>
     <Impression>http://example.com/pixel.gif</Impression>
     <Creatives>
      <Creative>
       <Linear>
        <Duration>00:00:08</Duration>
        <TrackingEvents>
         <Tracking event="start">http://example.com/pixel.gif</Tracking>
         <Tracking event="firstQuartile">http://example.com/pixel.gif</Tracking>
         <Tracking event="midpoint">http://example.com/pixel.gif</Tracking>
         <Tracking event="thirdQuartile">http://example.com/pixel.gif</Tracking>
         <Tracking event="complete">http://example.com/pixel.gif</Tracking>
         <Tracking event="pause">http://example.com/pixel.gif</Tracking>
         <Tracking event="mute">http://example.com/pixel.gif</Tracking>
         <Tracking event="fullscreen">http://example.com/pixel.gif</Tracking>
```

adtelligent

XML vs JSON

```
<error code="500">
   <message>Something bad happened</message>
</error>
```

```
{
      “error” : {
            “code” : 500,
            “message”: “Something bad happened”
      }
}
```

adtelligent

XML vs JSON

```
<error code="500">
   <message>Something bad happened</message>
   <message xml:lang=”de”>Etwas schlimmes ist passiert</message>
</error>
```

```
{
      “error” : {
            “id” : 500,
            “messages”: [
                  { “lang”: “en”, “message”: “Something bad happened” },
                  { “lang”: “de”, “message”: “Etwas schlimmes ist passiert” }
            ]
      }
}
```

adtelligent

XML vs JSON

```
<error code="500">
   <message>Something bad happened</message>
   <message xml:lang=”de”>Etwas schlimmes ist passiert</message>
</error>
```

```
{
      “error” : {
            “id” : 500,
            “message”: “Something bad happened”,
            “message_de”: “Etwas schlimmes ist passiert”
      }
}
```

adtelligent

XML vs JSON

```
<TrackingEvents>
  <Tracking event="start">http://example.com/pixel.gif</Tracking>
  <Tracking event="firstQuartile">http://example.com/pixel.gif</Tracking>
  <Tracking event="midpoint">http://example.com/pixel.gif</Tracking>
  <Tracking event="thirdQuartile">http://example.com/pixel.gif</Tracking>
  <Tracking event="complete">http://example.com/pixel.gif</Tracking>
  <Tracking event="pause">http://example.com/pixel.gif</Tracking>
  <Tracking event="mute">http://example.com/pixel.gif</Tracking>
  <Tracking event="fullscreen">http://example.com/pixel.gif</Tracking>
</TrackingEvents>
```

```
{
    “TrackingEvents”: [
        { “name” : “start”, “uri”: “http://example.com/pixel.gif” },
        { “name” : “mute”, “uri”: “http://example.com/pixel.gif” },
        { “name” : “Anjelica Boyd”, “uri”: “http://metart.com/model/krystal” }
    ]
}
```

adtelligent

Tools & Links

- FireBug and Chrome Developer Tools (any browser)
- https://codebeautify.org/json-tools
- https://codebeautify.org/xml-tools
- JetBrains Toolkit

- https://www.ibm.com/developerworks/ru/xml/
- http://json-schema.org/
- https://spacetelescope.github.io/understanding-json-schema/

adtelligent

Thank You!

// Nickolay Kharchevin
// https://github.com/mbIkola/json-xml/
// skype: nickolay.kharchevin

adtelligent