



Добре-структуриран XML (Well-Formed XML)

XML синтаксис
Йерархии
Елементи и атрибути
Правила
XML кодиране
XML и JSON

Основни характеристики на документ

Съдържание

- Информация, съдържаща текст, графика, аудио или видео

Структура

- Подялба и последователност на информационните части

Оформление

- Онагледяване на съдържанието и структурата на документ

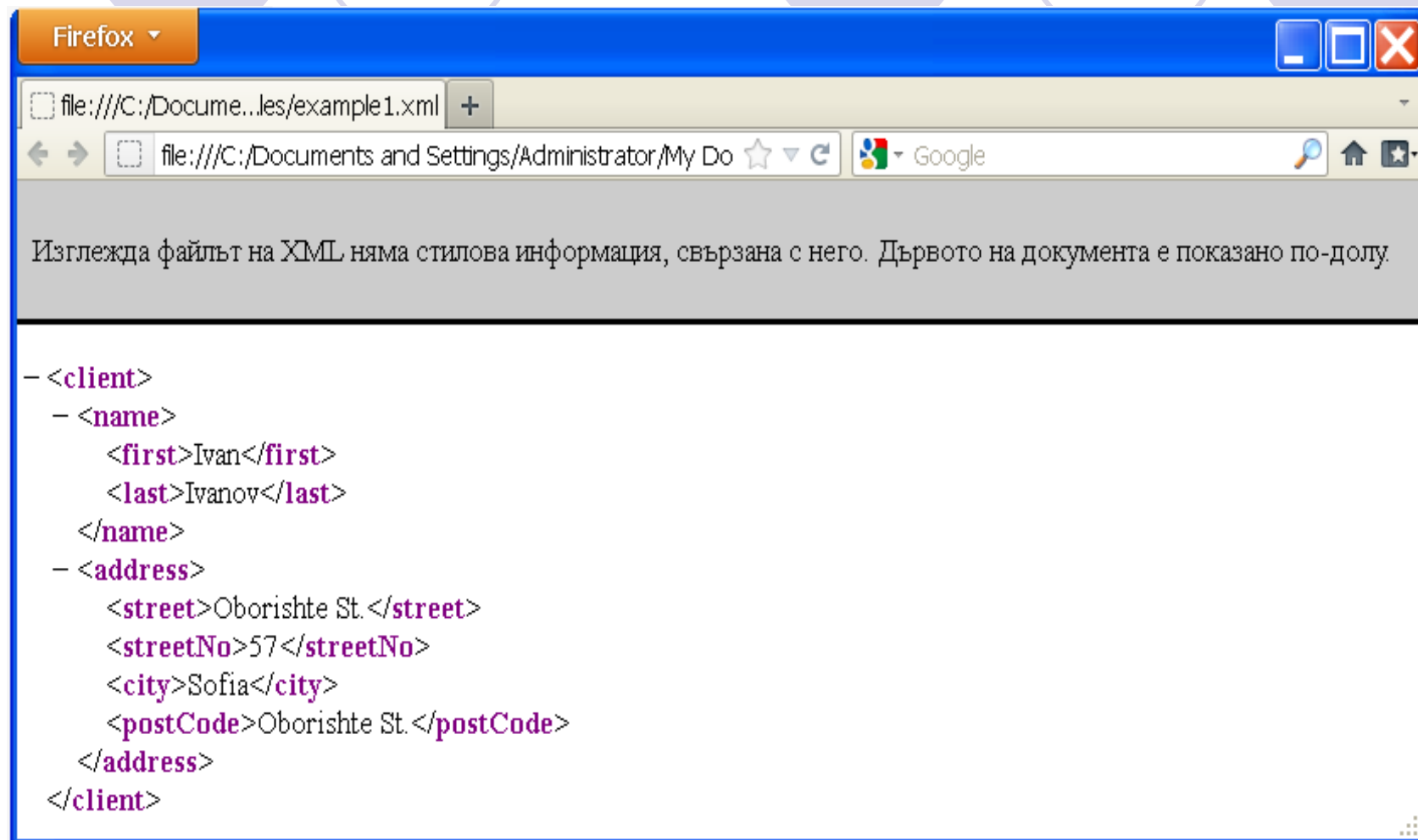
Метаданни

- Описание семантиката на съдържанието

Черти на езиците за маркиране

- **Стилистични** (appearance) – напр. в HTML:
<I><U>
- **Структурни** (layout) - напр. в HTML:
<P>
<H2>
- **Семантични** (meaning) - напр. в HTML:
<TITLE><META NAME=keywords CONTENT =
" " >
- **Функционални** (action) - напр. в HTML:
<BLINK>
Click here

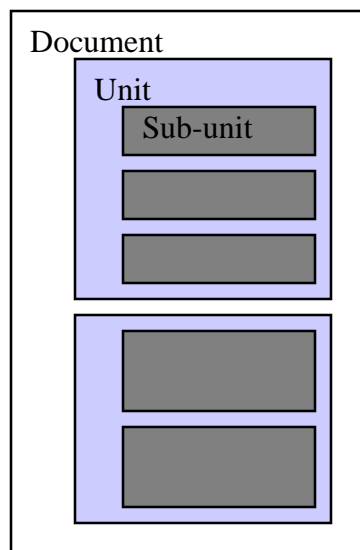
XML в браузер (без стилове)



Структура на XML документ

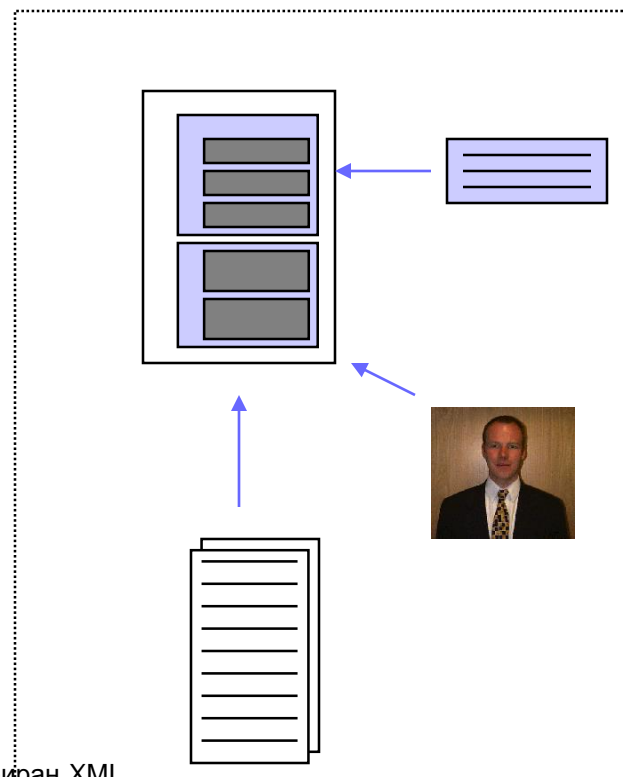
- Логическа структура

- Елементи

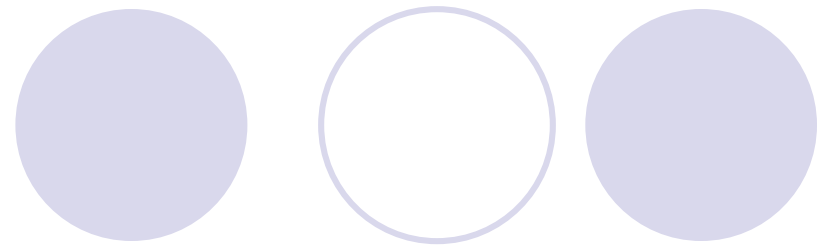


- Физическа структура

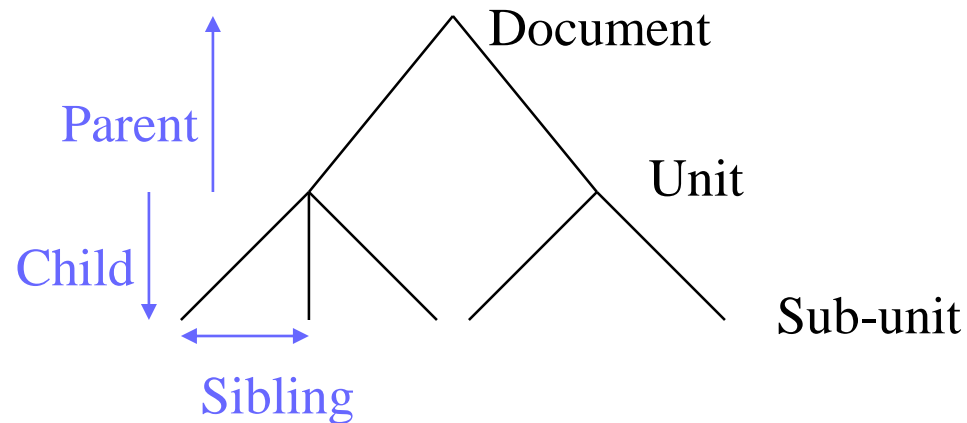
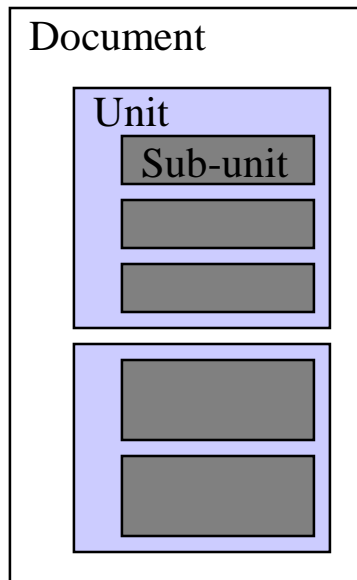
- Единици (Entities)



XML йерархия



Планарно и дървовидно представяне



N.B. All elements must be nested

XML синтаксис

- Маркер (tag):

- Име на елемент, разделено с **<** и **>**.

- *Start-Tag* **<elementname>**

- *End-Tag* **</elementname>**

- Елемент:

- **<elementname> Content of the element </elementname>**

- Типът на съдържанието е Parsed Character DATA (PCDATA)

- Само един елемент-корен

- Влагане на под-елементи

- **<author>**
 <first>Charles</first>
 <last>Gottlieb</last>
 </author>

XML – документна йерархия 1/2



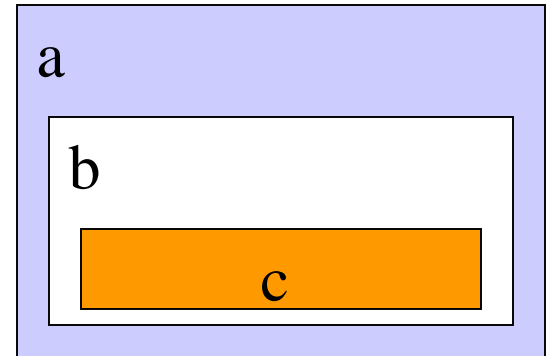
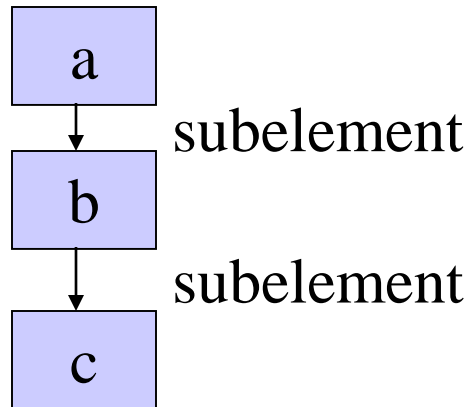
`<a>`

``

`<c> </c>`

``

``



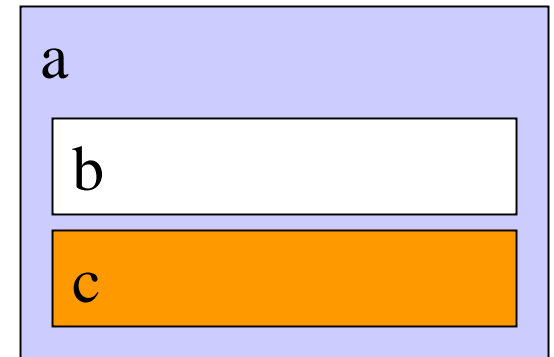
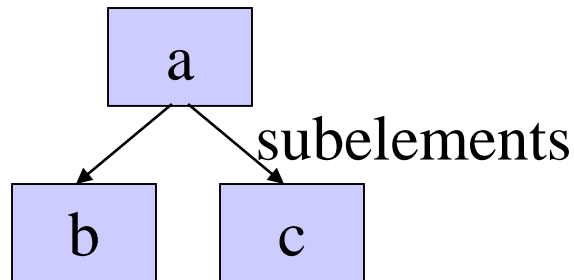
XML – документна йерархия 2/2

<a>

...

<c>...

</c>

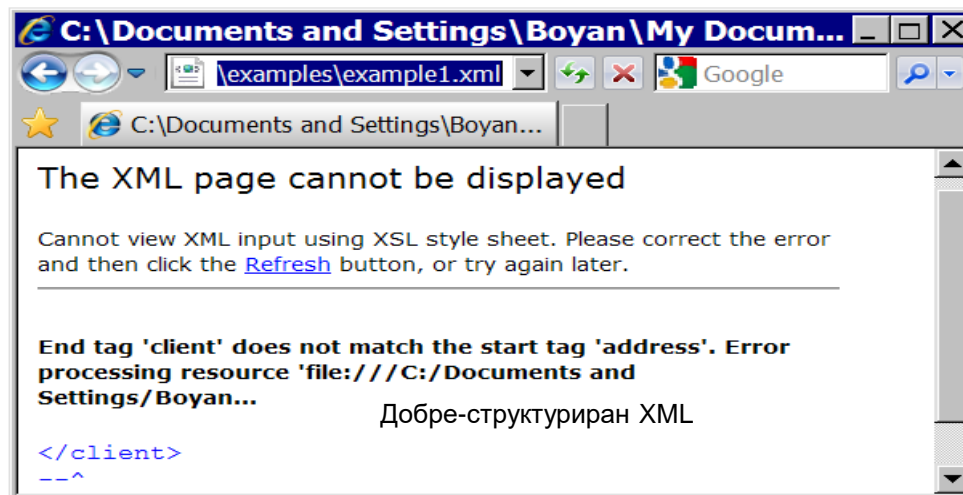
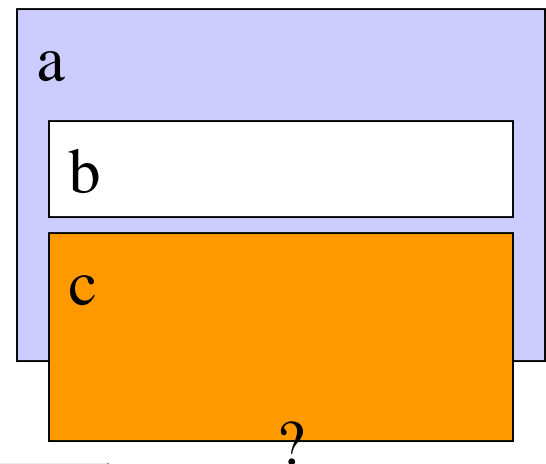
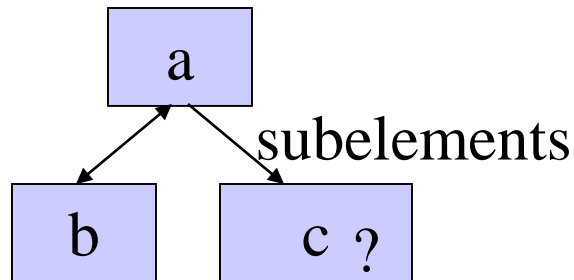


XML – липса на затварящ маркер

Error: missing C closing element tag

<a>

<c>

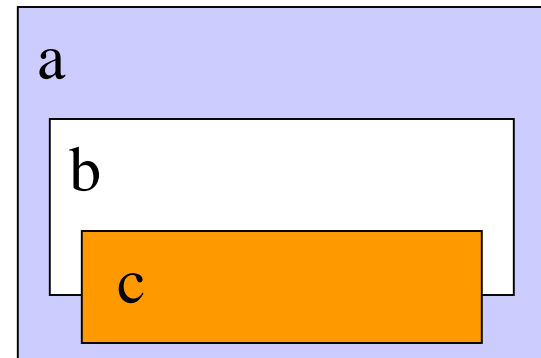
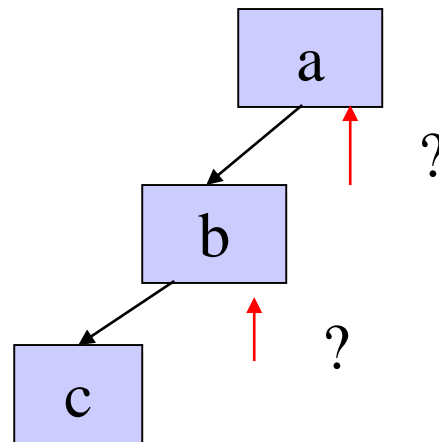


XML- припокриване на елементи

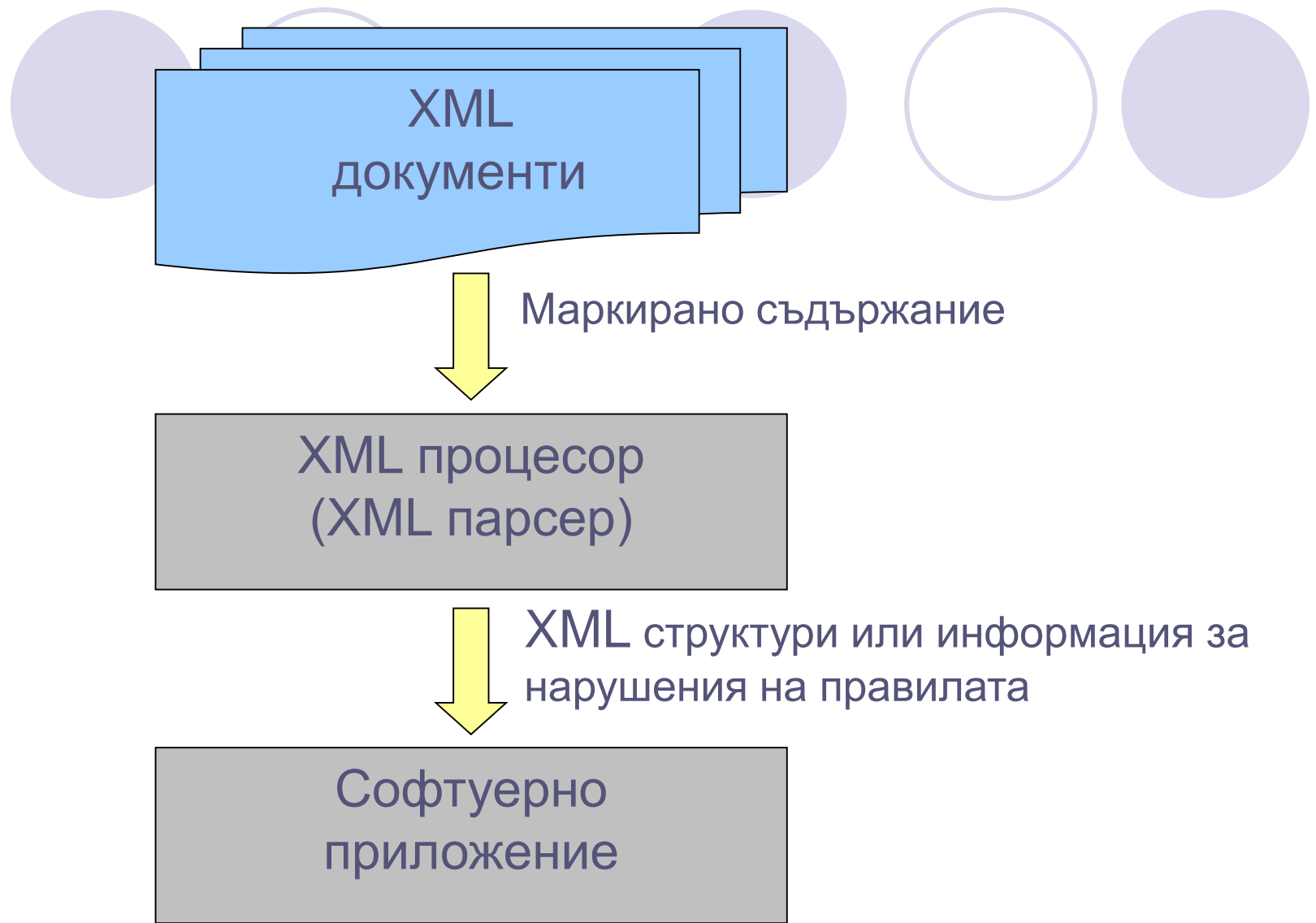
<a>

<c>

</c>

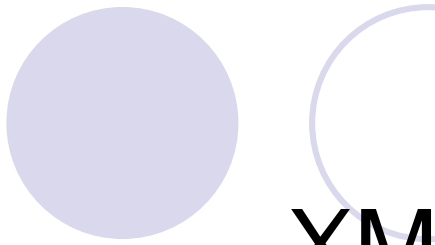


Error: broken elements -
with an *overlap* between the b and c element



Важно:

XML и всички XML-базирани езици са формални и имат формални граматики.



```
1. <person>
2.   <name>
3.     <first>Ganyo</first>
4.     <last>Balkanski</last>
5.   </name>
6.   <title>Bai</title>
7.   <alias>New European</alias>
8.   <birth>
9.     <place>any place</place>
10.    <time>any time</time>
11.  </birth>
12.  <character>
13.    <physical>
14.      slim, tall, cheery, attractive, too    clean
15.    </physical>
16.    <personal>
17.      dedicated, honest, philanthropist, leader, idealist
18.    </personal>
19.  </character>
20.  <address>anywhere in Bulgaria</address>
21.  <professions>
22.    <profession>image maker</profession>
23.    <profession>politician</profession>
24.  </professions>
25. </person>
```

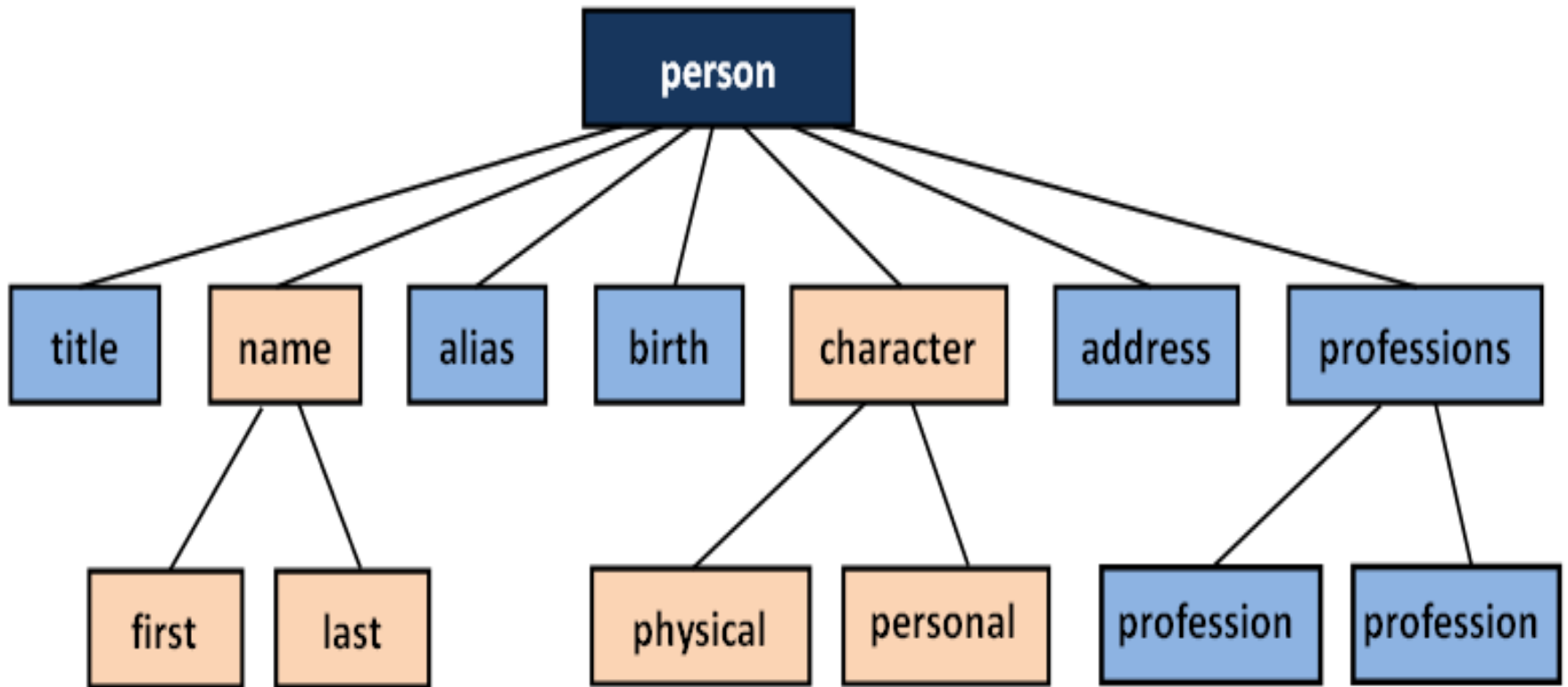
XML

Добре-структуриран XML

XML –
актуален
пример

XML дърво

N.B.: Only one root element!



Видове XML елементи 1/3

- Елементи с текстово съдържание - съдържанието представлява текст, ограден от начален и краен таг на XML елемент.
- По исторически причини, наследени от SGML, такъв тип елементно съдържание се нарича ***parsed character data***, или съкратено ***PCDATA***.
- Празните пространства (whitespace) се задават в текстовото съдържание посредством интервал (space), нов ред (чрез клавиша Enter) или табулация (Tab)

Видове XML елементи 2/3

- Елементи със структурно съдържание представляват такива XML елементи,
- в които текстовото съдържание е структурирано под формата на вложени елементи,
- организирани в дървовидна йерархия.

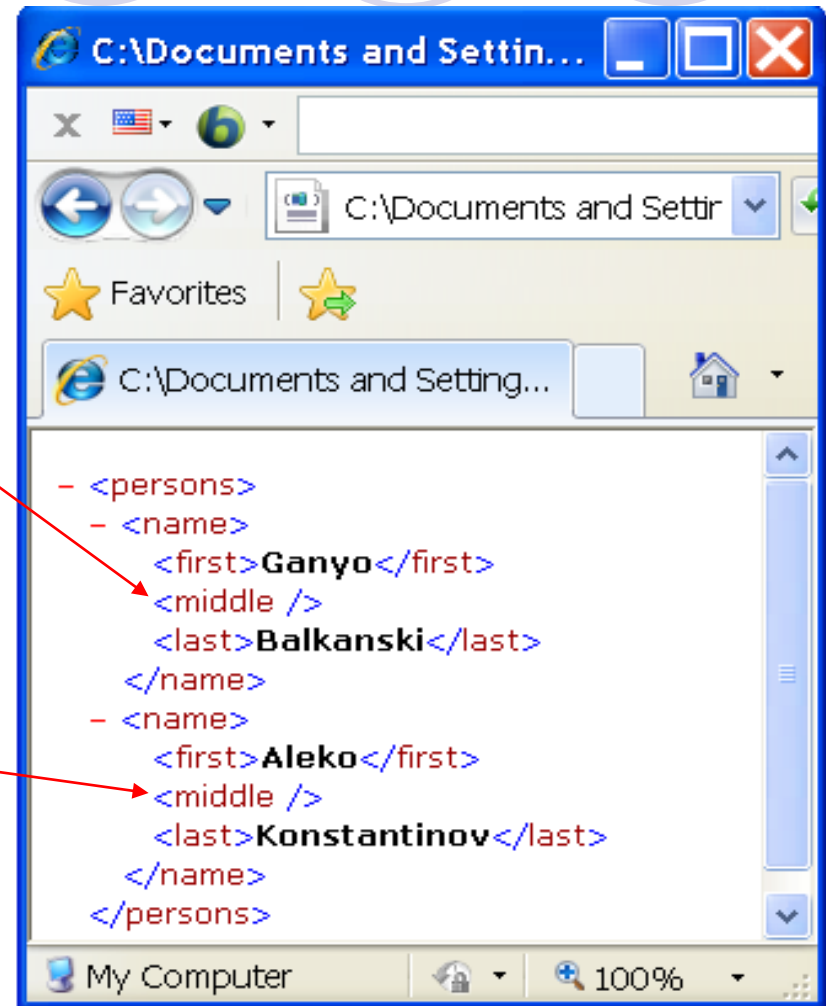
Видове XML елементи 3/3

Понякога елементите нямат нито PCDATA, нито структурно съдържание. Такива елементи се наричат **празни** и могат да се означат по два начина в XML документите:

- ❑ С начален и краен таг както всички останали елементи – такъв е първият елемент `<middle>` в примера по-долу;
- ❑ Посредством само-затварящ се (self-closing) таг, какъвто е вторият елемент `<middle>` в примера на следващата страница:

Пример за само-затварящ се (self-closing) таг

```
1. <persons>
2.   <name >
3.     <first>Ganyo</first>
4.     <middle></middle>
5.     <last>Balkanski</last>
6.   </name>
7.   <name >
8.     <first>Aleko</first>
9.     <middle />
10.    <last>Konstantinov</last>
11.  </name>
12.</persons>
```



Структура на XML документа

Пролог

```
<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<?xml:stylesheet type="text/css" href="s.css"?>
<!DOCTYPE test SYSTEM "test.dtd">
```

- Декларации на документ, стилове и тип на документа

Екземпляр

- Елементна йерархия

```
<author>
    <first>Ganyo</first>
    <last>Balkanski</last>
</author>
```

Допълнения

- Коментари, CDATA секции и инструкции за обработка

```
<!-- comment -->
<?myApp status="draft"?>
<![CDATA[ Press <<<ENTER>>> ]]>
```

Синтактични правила за елементи 1/2

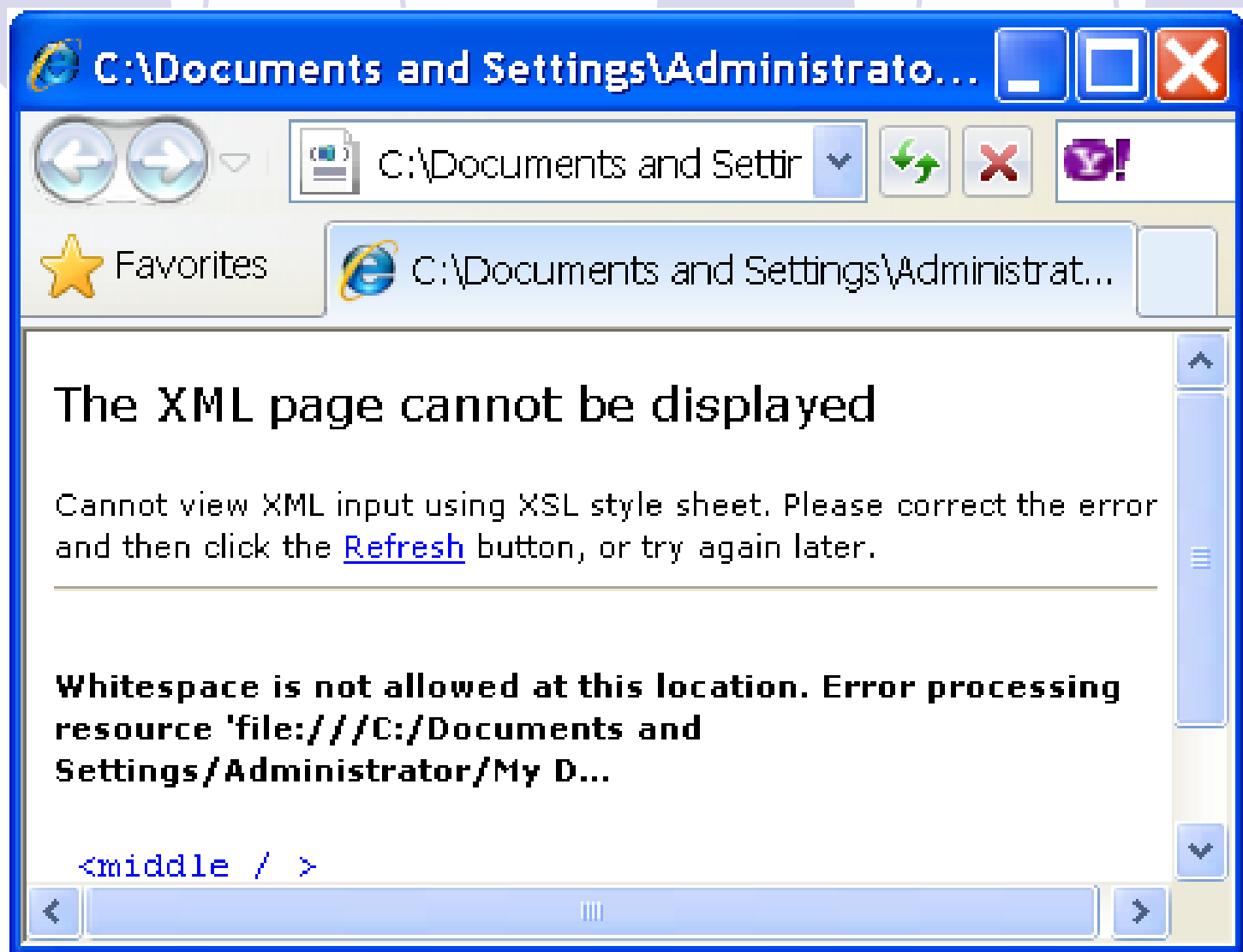
- Имената в XML са чувствителни към регистъра, т.е. XML парсерът прави разлика между използването на малки и големи букви в имената (case-sensitive) – затова `<person>`, `<Person>`, `<PerSon>` и `<PERSON>` са различни начални тагове;
- Имената трябва да започват с малка или главна буква или `'_'`;
- По-нататък, освен букви и `'_'`, в имената могат да бъдат използвани цифри, `'-'` и `'.'`;
- Имената не могат да започват с думите XML, xml, Xml, xmL и т.н.;
- Празни пространства се допускат само след имената на елементите, т.е. преди знака `'>'`;

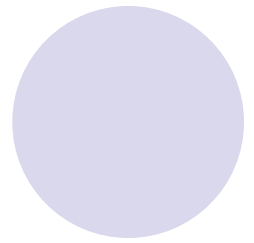
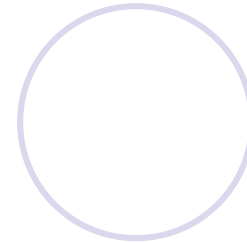
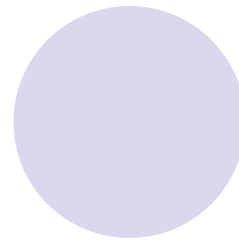
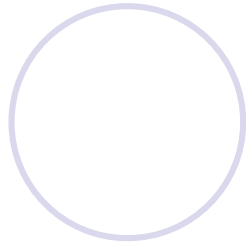
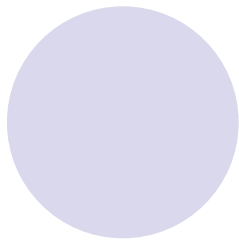
Синтактични правила за елементи 2/2

- Използването в имената на символа за двоеточие ':' е разрешено, резервирано – по-точно, знакът ':', използван в име на таг, отделя име на пространство от имена от локалното име на тага.
- В XML имената не могат да бъдат използвани други символи освен букви (вкл. и от азбуки, различни от латинската), цифри, празно пространство, '_', '-', ':' и '.';
- В XML имената празното пространство може да бъде интервал, нов ред или табулация, и е допустимо да се зададе само преди затварящия разделител '>';
- използването на символите '<' и '>' е резервирано за разделители;
- символите '&' и ';' се използват за указване на рефериране към съдържание посредством единица (entity) - наричана още семантична единица или същност.

Примерни XML имена на тагове

Valid names	Invalid names
<BobSun>	<Bob Sun>
<BobSun2003>	<2003BobSun>
<Bob-Sun>	<xmlTag>
<Bob.Sun>	<my=tag>
<résumé>	<a&b>
</middle >	</ middle>
<i>Question: Why correct?</i>	<i>Question: Why incorrect?</i>





Важно:

Поради чувствителността на XML към регистъра, в един XML документ е допустимо да се използват елементи с различно значение, имащи за начални тагове примерно `<title>` и `<Title>`, но това може да представлява причина за объркване или двусмислие.

Празните пространства (whitespace) се задават в текстовото съдържание посредством интервал (space), нов ред (чрез клавиша Enter) или табулация (Tab). Докато при маркиращи езици като HTML и WML поредица от празни пространства се запазва като едно (освен ако не са зададени като ** **), то при XML празните пространства се запазват в PCDATA елементите така, както са въведени от създателя на документа.

Така например елементът

```
<person>Aleko Konstantinov</person>
```

ще има различно съдържание от елемента

```
<person>Aleko           Konstantinov</person>
```

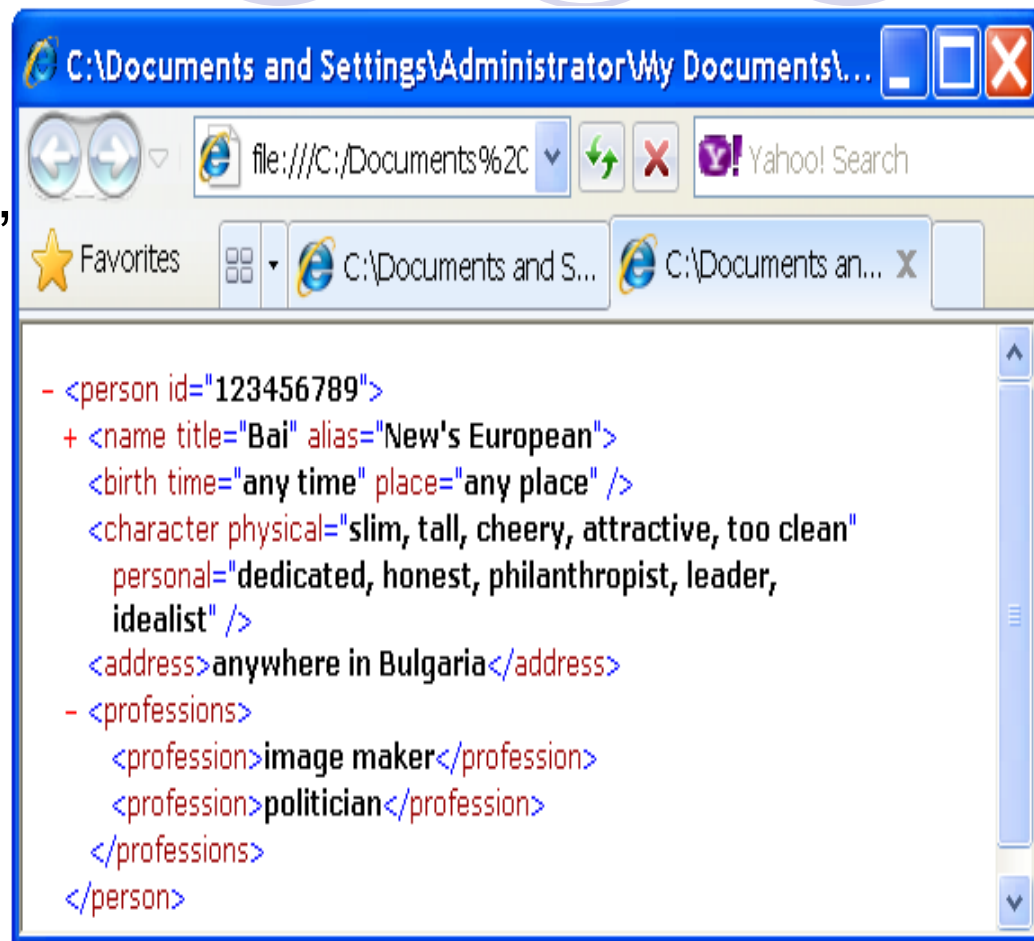
Важно: XML запазва празните пространства.

Синтактични правила за атрибути 1/3

- Всеки атрибут трябва да има стойност, дори и тя да бъде празен стринг;
- Атрибутът **id="123456789"** е добавен към елемента **<person>**, защото той характеризира като идентификатор целият физически индивид, а не само отделни елементи от описанието му;
- За разделител между два и повече атрибута се използва празно пространство, а не запетая;
- Възможно е едновременно използване на кавички и апострофи за ограждане на стойностите на атрибутите на един елемент, както е направено за елемента **<name>**. *Не е допустимо* обаче ограждане на стойност на един атрибут с кавички и апостроф или обратно, например **id="123456789'**.

Синтактични правила за атрибути 2/3

- Независимо от начина на ограждане на стойностите на атрибутите с документа, те се показват в браузер като **оградени с кавички**
- При сгъването в браузер на йерархията на елемент със структурно съдържание, атрибутите продължават да се показват (елемент **`<name title='Bai' alias="New's European">`** на фигурата)

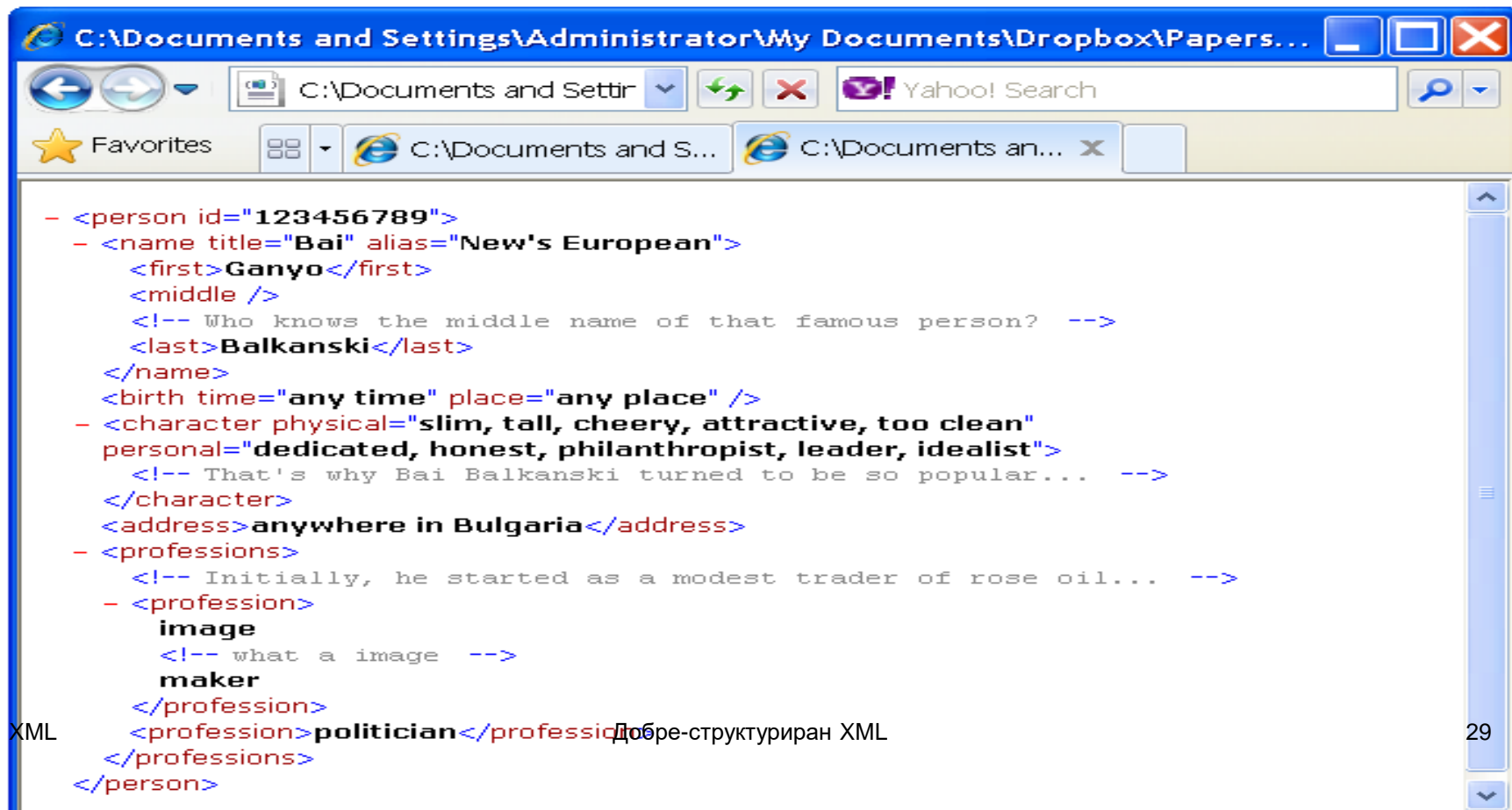


Синтактични правила за атрибути 3/3

- Дублиращи се атрибути (в същия елемент) не са разрешени в добре оформен XML документ.
- Атрибутите могат да следват произволен ред на подреждане в елемента.
- Това е решено да бъде така, защото те описват характеристики на елемента, които не са структурни и нямат релация за подреждане.

Коментари

- Не са видими за форматираната версия
- **<!-- this is a comment -->**
- Не се обработват от парсера



```
- <person id="123456789">
- <name title="Bai" alias="New's European">
  <first>Ganyo</first>
  <middle />
  <!-- Who knows the middle name of that famous person? -->
  <last>Balkanski</last>
</name>
<birth time="any time" place="any place" />
- <character physical="slim, tall, cheery, attractive, too clean"
  personal="dedicated, honest, philanthropist, leader, idealist">
  <!-- That's why Bai Balkanski turned to be so popular... -->
</character>
<address>anywhere in Bulgaria</address>
- <professions>
  <!-- Initially, he started as a modest trader of rose oil... -->
  - <profession>
    image
    <!-- what a image -->
    maker
  </profession>
  <profession>politician</profession>
</professions>
</person>
```

При- мер

```
Back Address C:\Boyan\Lectures\XML\books\BeginningXML\Code Download\5598_chapter02_well_formed\cd3.xml Go Links »
Google Search Web Search Site PageRank Options

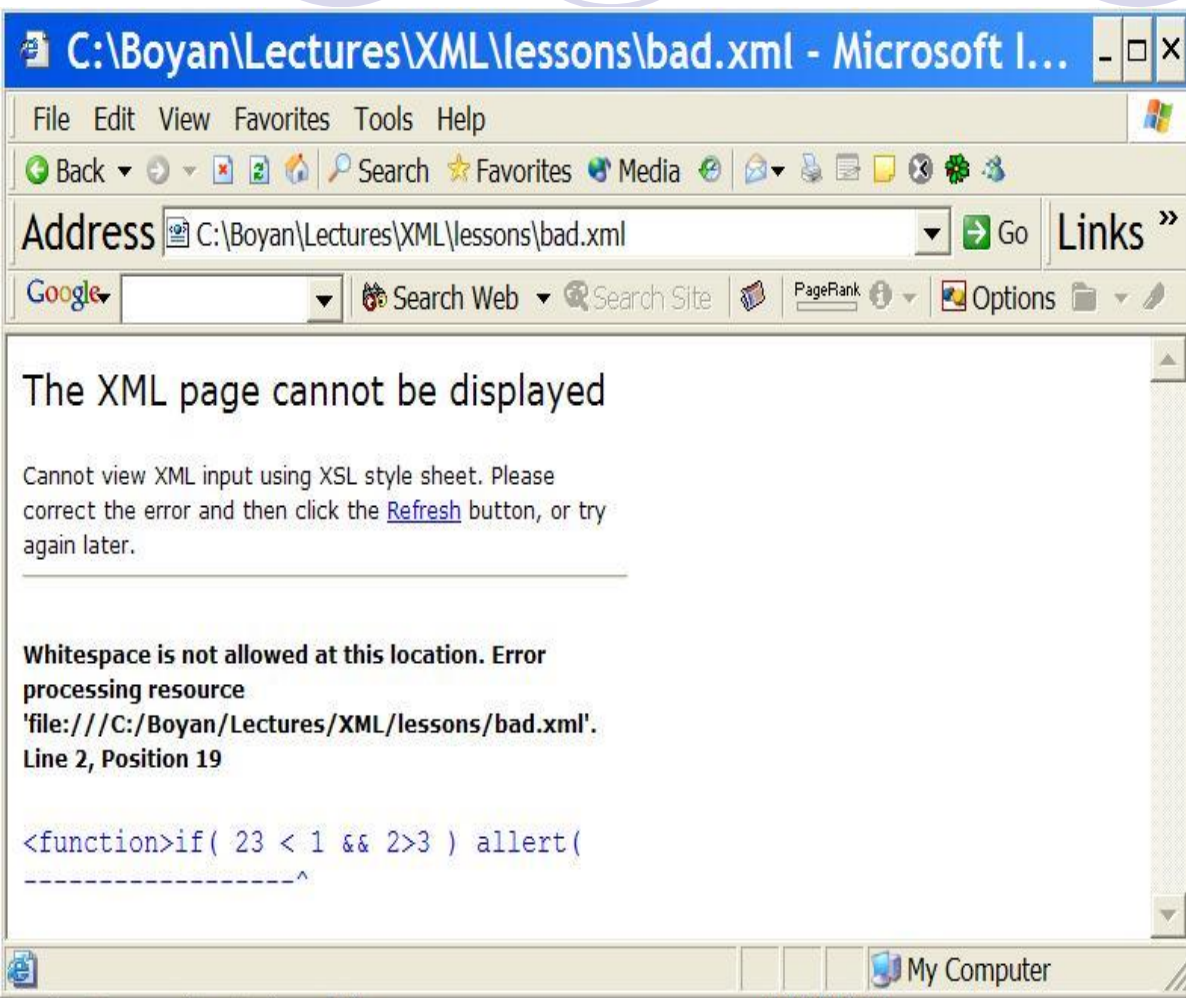
- <CD serial="B6B41B" disc-length="36:55">
  <artist>"Weird Al" Yankovic</artist>
  <title>Dare to be Stupid</title>
  <genre>parody</genre>
  <date-released>1990</date-released>
  <!-- date-released is the date released to CD, not to record -->
- <song>
  <title>Like A Surgeon</title>
  - <length>
    <minutes>3</minutes>
    <seconds>33</seconds>
  </length>
  - <parody>
    <title>Like A Virgin</title>
    <artist>Madonna</artist>
  </parody>
</song>
- <song>
  <title>Dare to be Stupid</title>
  - <length>
    <minutes>3</minutes>
    <seconds>25</seconds>
  </length>
  <parody />
</song>
<!-- there are more songs on this CD, but I didn't have time to include
them -->
</CD>
```

Екранни последователности

- За предефинирани единици:

○&	&	(ampersand)
○<	<	(less than)
○>	>	(greater than)
○'	'	(apostrophe)
○"	"	(quotation mark)

Забранени PCDATA симболи



XML

Добре-структуриран XML

● Пример:

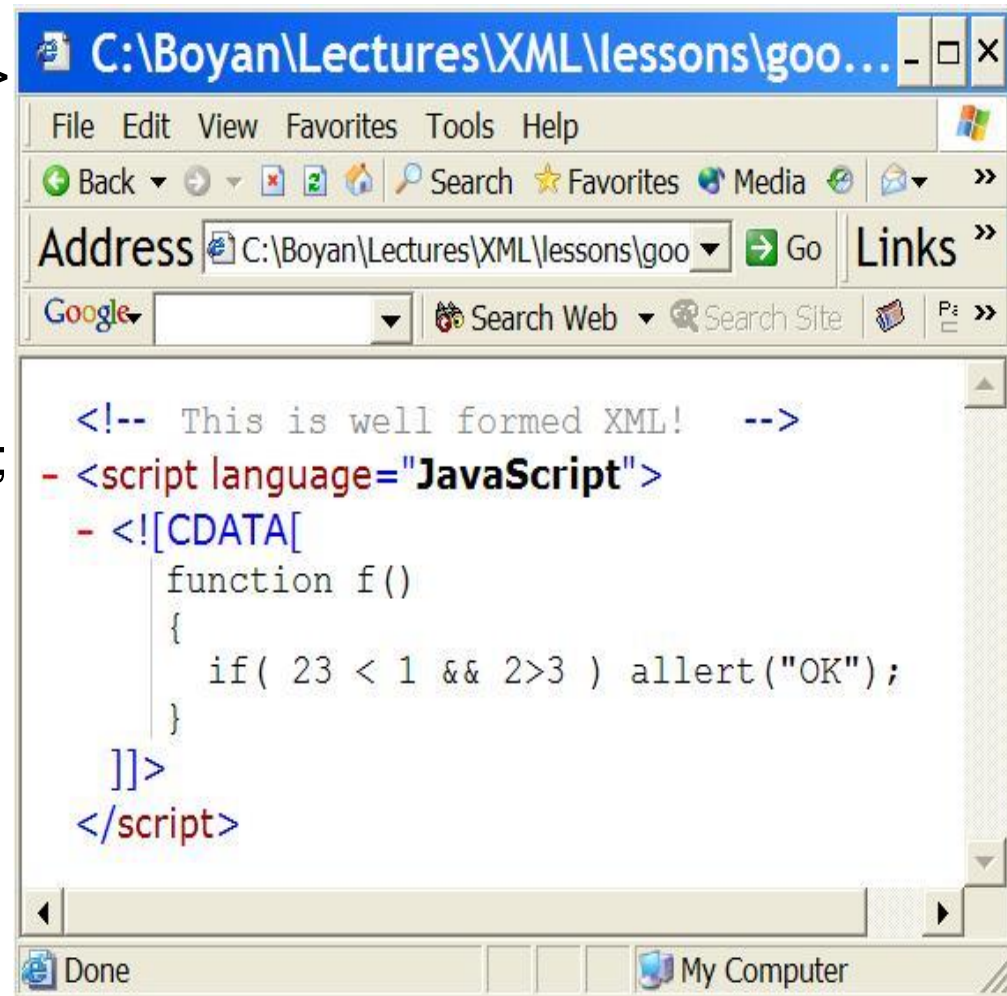
1. '<', '&', '>'
2. `<!-- This is not well formed XML! -->`
3. `<function>if(23 < 1 && 2>3) alert("OK");`
4. `</function>`

Решение: CDATA секции

- CDATA (Character DATA) идва от SGML
- Започват с **<![CDATA[**
- Завършват с **]]>**
- CDATA съдържанието се игнорира от парсъра и се подава към приложенията *as-is*

CDATA пример

```
1. <!-- This is well formed XML! -->
2. <script language="JavaScript">
3. <![CDATA[
4. function f()
5. {
6.   if( 23 < 1 && 2>3 ) alert("OK");
7. }
8. ]]>
9. </script>
```



Атрибути

- Атрибутът се задава като наредена двойка име-стойност
- Доставя инфо за елемента
- Стойността се огражда от " или '
- Чувствителност към регистъра
- Видове
 - `Value = "Blue Peter"` (character data)
 - `value = "blue"` (single token)
 - `value = "red green blue"` (tokens)
 - enumerated or defaulted (DTD)

Избор между XML елементи и атрибути 1/2

- Ако дадено съдържание представлява неструктурирани или неподредени метаданни за друго, то най-лесно да се представи чрез атрибути. Атрибутите заемат по-малко място и се обработват много лесно от софтуерните приложения посредством интерфейси като DOM, SAX и StAX;
- Ако дадено съдържание представлява структурирани данни или метаданни, то трябва да се представи чрез елементи вместо атрибути. При атрибутите липсва релация за подреждане и затова те не могат да представят нито линейна, нито йерархична структура;

Избор между XML елементи и атрибути 2/2

- При използване на изброими стойности е удачно да се използват атрибути;
- Ако документът ще се ползва най-вече от хора, за предпочитане е да се използват елементи, понеже структурираната йерархия от елементи е по-четима от множество атрибути, изредени един след друг.

Още по дилемата: Elements or Attributes?

- 1] Does the value have one of an enumeration of values or is the value free-form?
 - [1a] - enumerated values can be name token groups in attributes
 - [1e] - no restrictions on values for the content of elements
- [2] Is the value to be specified, manipulated, organised, consumed by a program or by a human?
 - [2a] - I use attributes for computer-manipulated values
 - [2e] - I use elements for human-manipulated values
- [3] Does the information represent information *about* content, or is the information the content itself?
 - [3a] - I typically put meta-data in attributes
 - [3e] - I typically put content into elements
- [4] Is the information flat or hierarchical?
 - [4a] - attributes are flat and a value has no hierarchy
 - [4e] - elements can be either flat or hierarchical
- [5] Is the information unordered or ordered?
 - [5a] - multiple attribute values in a single start element have no prescribed order
 - [5e] - multiple child elements of an element can be modelled in a prescribed order

XML

Добре-структуриран XML

38

Source: <http://www.oasis-open.org/cover/holmanElementsAttrs.html>

Инструкции за обработка

- Инструкциите за обработка (processing instructions, или съкратено PIs) представляват начин за предаване на скрита информация към дадено приложение.
- Също както коментарите, те не са част от действителното съдържание на документа, но XML процесорите са длъжни да ги взимат под внимание и да ги предават на приложението.
 - Формат- `<? ... ?>`
 - `<?xml version='1.0' ?>`
 - `<?xml:stylesheet type="text/css" href="sample.css" ?>`

XML декларация

The **XML declaration** is mandatory if the encoding of the document is anything other than UTF-8 or UTF-16.

XML Latest version = **1.1**
(Second Edition)
September 29, 2006.

- `<?xml version= "1.0"?>`
- `<?xml version= "1.0" encoding="UTF-8"?>`
- `<?xml version= "1.0" encoding="UTF-8" standalone="yes"?>`

standalone="yes" means that the **XML** processor must use the DTD for validation only. „**no**“ is the default value. The attribute is irrelevant when using a schema instead of a DTD.

The **XML Declaration** takes the form of a Processing Instruction and can have the attributes version, encoding and standalone.

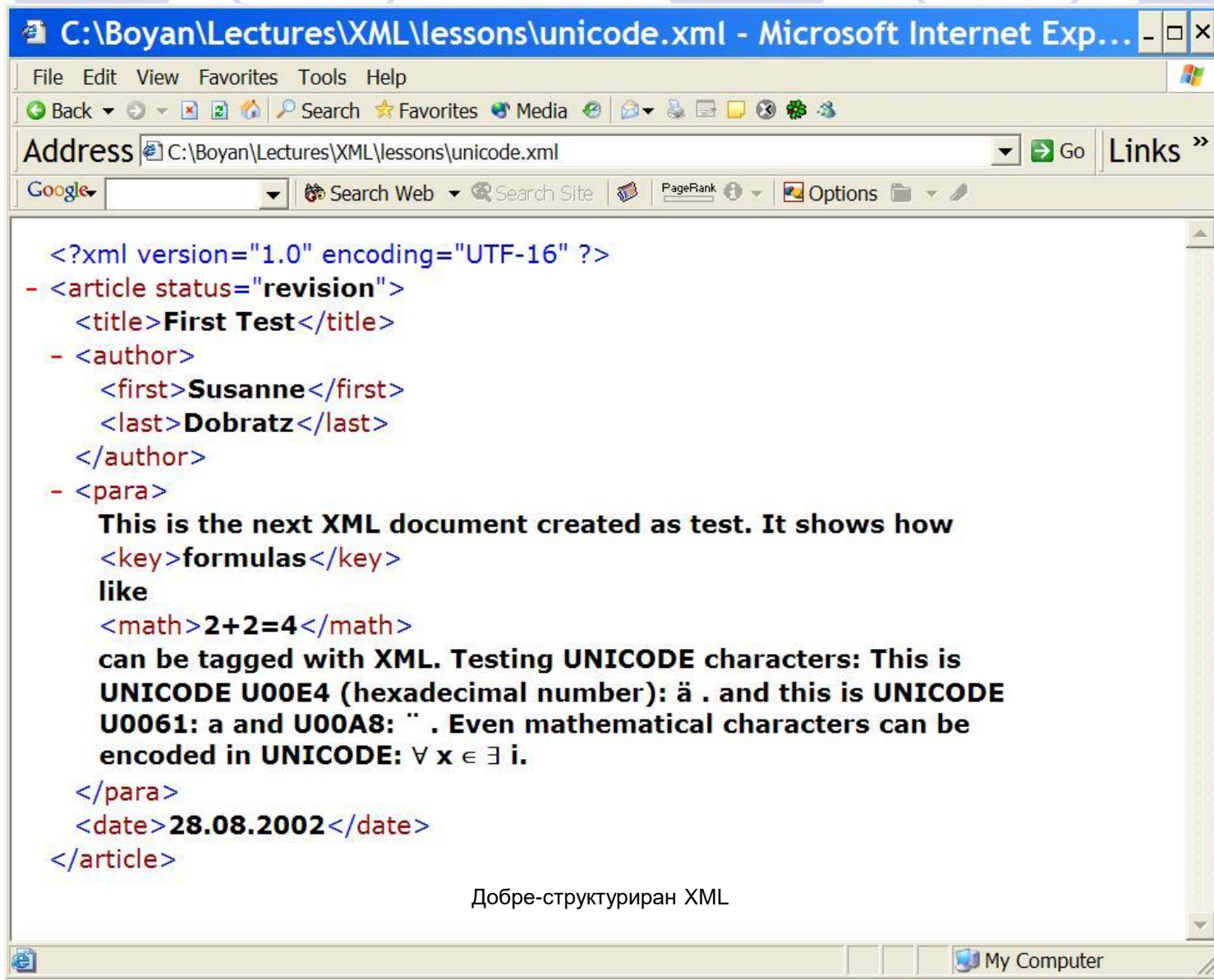
Кодиране: ISO 8859-1 спрямо UNICODE

- Първи ред от XML документ
 - `<?xml version="1.0" encoding="UTF-8" ?>`
- Не е задължително декларирането на кодиране
- UTF-16 или UTF-8 (Unicode Transformation Format) е кодирането по подразбиране
- Ако кодирането не е указано и то не е нито UTF-8, нито UTF-16 => ERROR!
- UNICODE
 - UTF-8 (включва 7-битовия ASCII)
 - UTF16 (2 байта за всички символи => 65536 chars)
- ISO 8859-1
 - 8 битов код (ANSI)
- Може да се конфигурира в браузъра

UNICODE пример – XML

- `<?xml version="1.0" encoding="UTF-16"?>`
- `<article status="revision">`
- `<title>First Test</title>`
- `<author><first>Susanne</first><last>Dobratz</last></author>`
- `<para>`This is the next XML document created as test. It shows how `<key>`formulas`</key>`like `$2+2=4$`can be tagged with XML.
- Testing UNICODE characters: This is UNICODE U00E4 (hexadecimal number): `ä` . and this is UNICODE U0061: `a` and U00A8: `¨` . Even mathematical characters can be encoded in UNICODE:
- `∀` x `∈` `∃` i.
- `</para><date>28.08.2002</date>`
- `</article>`

UNICODE пример – XML в MS IE

A screenshot of a Microsoft Internet Explorer browser window. The title bar reads "C:\Boyan\Lectures\XML\lessons\unicode.xml - Microsoft Internet Exp...". The address bar shows the file path "C:\Boyan\Lectures\XML\lessons\unicode.xml". The main content area displays an XML document with the following structure: a root element <?xml version="1.0" encoding="UTF-16" ?>, followed by an <article status="revision"> element. Inside the article, there is a <title>First Test</title>, an <author> element with sub-elements <first>Susanne</first> and <last>Dobratz</last>, a <para> element containing text about testing UNICODE characters and mathematical formulas, and a <date>28.08.2002</date>. The document ends with </article>. The browser's status bar at the bottom shows "My Computer".

```
<?xml version="1.0" encoding="UTF-16" ?>
- <article status="revision">
  <title>First Test</title>
  - <author>
    <first>Susanne</first>
    <last>Dobratz</last>
  </author>
  - <para>
    This is the next XML document created as test. It shows how
    <key>formulas</key>
    like
    <math>2+2=4</math>
    can be tagged with XML. Testing UNICODE characters: This is
    UNICODE U00E4 (hexadecimal number): ä . and this is UNICODE
    U0061: a and U00A8: " . Even mathematical characters can be
    encoded in UNICODE:  $\forall x \in \exists i$ .
  </para>
  <date>28.08.2002</date>
</article>
```

JSON (JavaScript Object Notation)

- Лесен за използване - JSON API предлага фасада на ВИСОКО НИВО.
- Производителност - JSON е доста бърз, тъй като консумира много по-малко място в паметта, което е особено подходящо за големи обектни графики или системи.
- Безплатен инструмент - библиотеката JSON е с отворен код и безплатна за използване.
- Не се изисква създаване на mapping - Jackson API осигурява mapping за много обекти за сериализиране.
- Clean JSON - чист и съвместим JSON резултат, който е лесен за четене.
- Зависимости - JSON библиотеката не изисква друга библиотека за обработка.

```

<?xml version="1.0" encoding="UTF-8"?>
<DatabaseInventory>
  <DatabaseName>

  <GlobalDatabaseName>production.cubicrace.com</
GlobalDatabaseName>
    <OracleSID>production</OracleSID>
    <Administrator EmailAlias="piyush"
Extension="6007">Piyush
Chordia</Administrator>
    <DatabaseAttributes Type="Production"
Version="9i" />
    <Comments>All new accounts need to be
approved.</Comments>
  </DatabaseName>
  <DatabaseName>

  <GlobalDatabaseName>development.cubicrace.com<
/GlobalDatabaseName>
    <OracleSID>development</OracleSID>
    <Administrator EmailAlias="kalpana"
Extension="6008">Kalpana
Pagariya</Administrator>
    <DatabaseAttributes Type="Development"
Version="9i" />
  </DatabaseName>
</DatabaseInventory>

```

```

{
  DatabaseInventory: {
    DatabaseName: [
      {
        GlobalDatabaseName: "production.cubicrace.com",
        OracleSID: "production",
        Administrator: [
          {
            EmailAlias: "piyush",
            Extension: "6007",
            value: "Piyush Chordia"
          }
        ],
        DatabaseAttributes: {
          Type: "Production",
          Version: "9i"
        },
        Comments: "All new accounts need to be approved."
      },
      {
        GlobalDatabaseName: "development.cubicrace.com",
        OracleSID: "development",
        Administrator: [
          {
            EmailAlias: "kalpana",
            Extension: "6008",
            value: "Kalpana Pagariya"
          }
        ],
        DatabaseAttributes: {
          Type: "Development",
          Version: "9i"
        }
      }
    ]
  }
}

```

JSON vs XML 1/2

JSON	XML
JSON doesn't use end tag, is shorter & quicker to read and write, and can use arrays	XML uses end tag, is longer & slower to read and write; no arrays
JSON can be parsed by a standard JavaScript function.	XML has to be parsed with an XML parser.
JSON has no display capabilities	XML offers the capability to display data
JSON is supported by most browsers.	Cross-browser XML parsing can be tricky
JSON has no display capabilities.	XML offers the capability to display data because it is a markup language.
JSON supports only text and number data type.	XML support various data types such as number, text, images, charts, graphs, etc. It also provides options for transferring the structure or format of the data with actual data.
Retrieving value is easy	Retrieving value is difficult

JSON vs XML 2/2

JSON	XML
Supported by many Ajax toolkit	Not fully supported by Ajax toolkit
A fully automated way of deserializing/serializing JavaScript.	Developers have to write JavaScript code to serialize/de-serialize from XML
Native support for object.	The object has to be express by conventions – mostly missed use of attributes and elements.
It supports only UTF-8 encoding.	It supports various encoding.
It doesn't support comments.	It supports comments.
JSON files are easy to read as compared to XML.	XML documents are relatively more difficult to read and interpret.
It does not provide any support for namespaces.	It supports namespaces.
It is less secured.	It is more secure than JSON.
JSON schema	XML schema

