XML and its validation: XML DTD, XML Schema

Web Engineering

utad UNIVERSIDADE DE TRÁS-OS-MONTES E ALTO DOURO

XML Applications?

- Specialized markup languages (eg: MathML, Office Open XML)
- Communication Protocols (SOAP)
- Configuration (setup) parameters
- …

```
<math xmlns="http://www.w3.org/1998/Math/MathML" display="block">
 <munderover>
    <mo>&Sum;</mo>
    <mrow>
        <mi>i</mi>
        <mo>=</mo>
        <mn>1</mn>
    </mrow>
    <mi>n</mi>
  </munderover>
  <mfenced separators=''>
    <msub>
    <mi>a</mi>
    <mi>i</mi>
    </msub>
  </mfenced>
</math>
```

What does it means?

$$\sum_{i=1}^{n}(a_i)$$

What are the elements of an XML document?

```
<?xml version='1.0' encoding='UTF-16' standalone='yes'?>
<name nickname='Shiny John'>
    <first>John</first>
    <!--John lost his middle name in a fire-->
    <middle/>
    <?nameprocessor SELECT * FROM blah?>
    <last>Doe</last>
</name>
```

- XML Declaration
- Elements
- Empty elements
- Attributes
- Comments
- Processing instructions

View in https://www.codeproject.com/Articles/845/Beginning-XML-Chapter-2-Well-Formed-XML

Rules for Elements

- Every start-tag must have a matching end-tag
- Tags can't overlap
- XML documents can have only one root element
- Element names must obey XML naming conventions

  Element names must start with a letter or underscore
  Element names cannot start with the letters xml (or XML, or Xml, etc.)
  Element names can contain letters, digits, hyphens, underscores, and periods
  Element names cannot contain spaces
- XML is case-sensitive
- XML will keep white space in your text

- What determines that an XML document is well formed?
- Are all well formed documents valid? Why?
- Are all valid documents well formed? Why?

Is this XML document well formed?

YES

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Is it valid?

NO

Why?

“An XML document is ***well formed*** *if it includes document declaration, satisfies element nesting and other syntactic constraints, and does not include undefined entities. A* *<u>well-formed XML document</u>* *is* ***valid*** *if it satisfies constraints as specified in the associated DTD or a Schema. Of course, only a well-formed document may be valid.”*

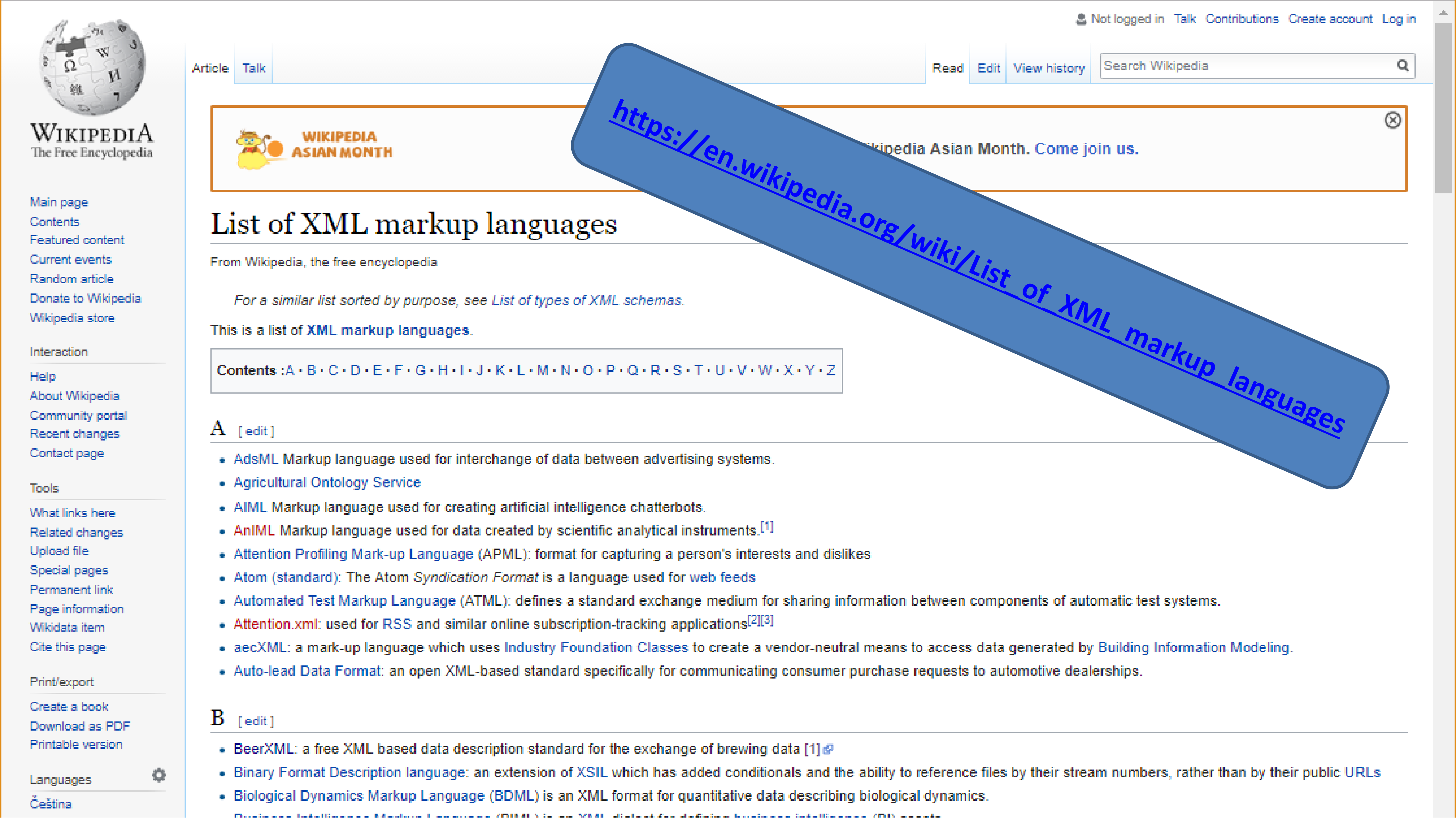
page 87

Is this XML document valid?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Note.dtd

```
<!DOCTYPE note [
        <!ELEMENT note (to,from,heading,body)>
        <!ELEMENT to (#PCDATA)>
        <!ELEMENT from (#PCDATA)>
        <!ELEMENT heading (#PCDATA)>
        <!ELEMENT body (#PCDATA)>
]>
```



Not logged in Talk Contributions Create account Log in

Article Talk Read Edit View history Search Wikipedia

WIKIPEDIA The Free Encyclopedia

Main page
Contents
Featured content
Current events
Random article
Donate to Wikipedia
Wikipedia store

Interaction
Help
About Wikipedia
Community portal
Recent changes
Contact page

Tools
What links here
Related changes
Upload file
Special pages
Permanent link
Page information
Wikidata item
Cite this page

Print/export
Create a book
Download as PDF
Printable version

Languages
Čeština

WIKIPEDIA ASIAN MONTH ...kipedia Asian Month. Come join us.

https://en.wikipedia.org/wiki/List_of_XML_markup_languages

List of XML markup languages

From Wikipedia, the free encyclopedia

For a similar list sorted by purpose, see List of types of XML schemas.

This is a list of **XML markup languages**.

Contents : A · B · C · D · E · F · G · H · I · J · K · L · M · N · O · P · Q · R · S · T · U · V · W · X · Y · Z

A [edit]

- AdsML Markup language used for interchange of data between advertising systems.
- Agricultural Ontology Service
- AIML Markup language used for creating artificial intelligence chatterbots.
- AnIML Markup language used for data created by scientific analytical instruments.[1]
- Attention Profiling Mark-up Language (APML): format for capturing a person's interests and dislikes
- Atom (standard): The Atom *Syndication Format* is a language used for web feeds
- Automated Test Markup Language (ATML): defines a standard exchange medium for sharing information between components of automatic test systems.
- Attention.xml: used for RSS and similar online subscription-tracking applications[2][3]
- aecXML: a mark-up language which uses Industry Foundation Classes to create a vendor-neutral means to access data generated by Building Information Modeling.
- Auto-lead Data Format: an open XML-based standard specifically for communicating consumer purchase requests to automotive dealerships.

B [edit]

- BeerXML: a free XML based data description standard for the exchange of brewing data [1]
- Binary Format Description language: an extension of XSIL which has added conditionals and the ability to reference files by their stream numbers, rather than by their public URLs
- Biological Dynamics Markup Language (BDML) is an XML format for quantitative data describing biological dynamics.
- Business Intelligence Markup Language (BIML) is an XML dialect for defining business intelligence (BI) assets.

What is the importance of the CDATA field?

Characters like "<" and "&" are illegal in XML elements.

- "<" will generate an error because the parser interprets it as the start of a new element.
- "&" will generate an error because the parser interprets it as the start of an character entity.

treated as special characters

Symbol (name)	Escape Sequence
< (less-than)	&#60; or &lt;
> (greater-than)	&#62; or &gt;
& (ampersand)	&#38;
' (apostrophe or single quote)	&#39;
" (double-quote)	&#34;

What is the importance of the CDATA field?

How to represent in XML the function `matchwo`?

```
<script type="text/javascript">

     function matchwo(a,b)
     {
        if (a < b && a < 0){
           return 1;
        } else {
           return 0;
        }
     }

</script>
```

```
<script type="text/javascript"><![CDATA[
    function matchwo(a,b)
    {
        if (a < b && a < 0){
            return 1;
        } else {
            return 0;
        }
    }
]]></script>
```

*“In addition to comments, XML possesses an even stronger mechanism for hiding XML fragments—**CDATA** sections. CDATA sections exclude enclosed text from XML parsing—all text is interpreted as **character data**. It is useful for hiding document fragments that contain comments, quotes, or ‘&’ characters.* (…) *The only character sequence that may not occur in the CDATA section is ‘]]>’, which terminates the section.”*

page 88

What are the limitations of DTD?

“DTDs do not provide easy ways of defining constraints that go beyond element nesting and recurrence. Similarly, while it is easy to associate elements with attributes and to define basic constraints on attribute types (e.g. predefined name token constraints, or enumeration), more complex cases are often an exercise in futility. And of course, DTD is no help in trying to define semantics.”

page 90

Present a valid XML document for this DTD:

```
<!DOCTYPE note [
          <!ELEMENT note (to,from,heading,body)>
          <!ELEMENT to (#PCDATA)>
          <!ELEMENT from (#PCDATA)>
          <!ELEMENT heading (#PCDATA)>
          <!ELEMENT body    (#PCDATA)>
]>
```

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE note SYSTEM "Note.dtd">
<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Is there an alternative to DTD?

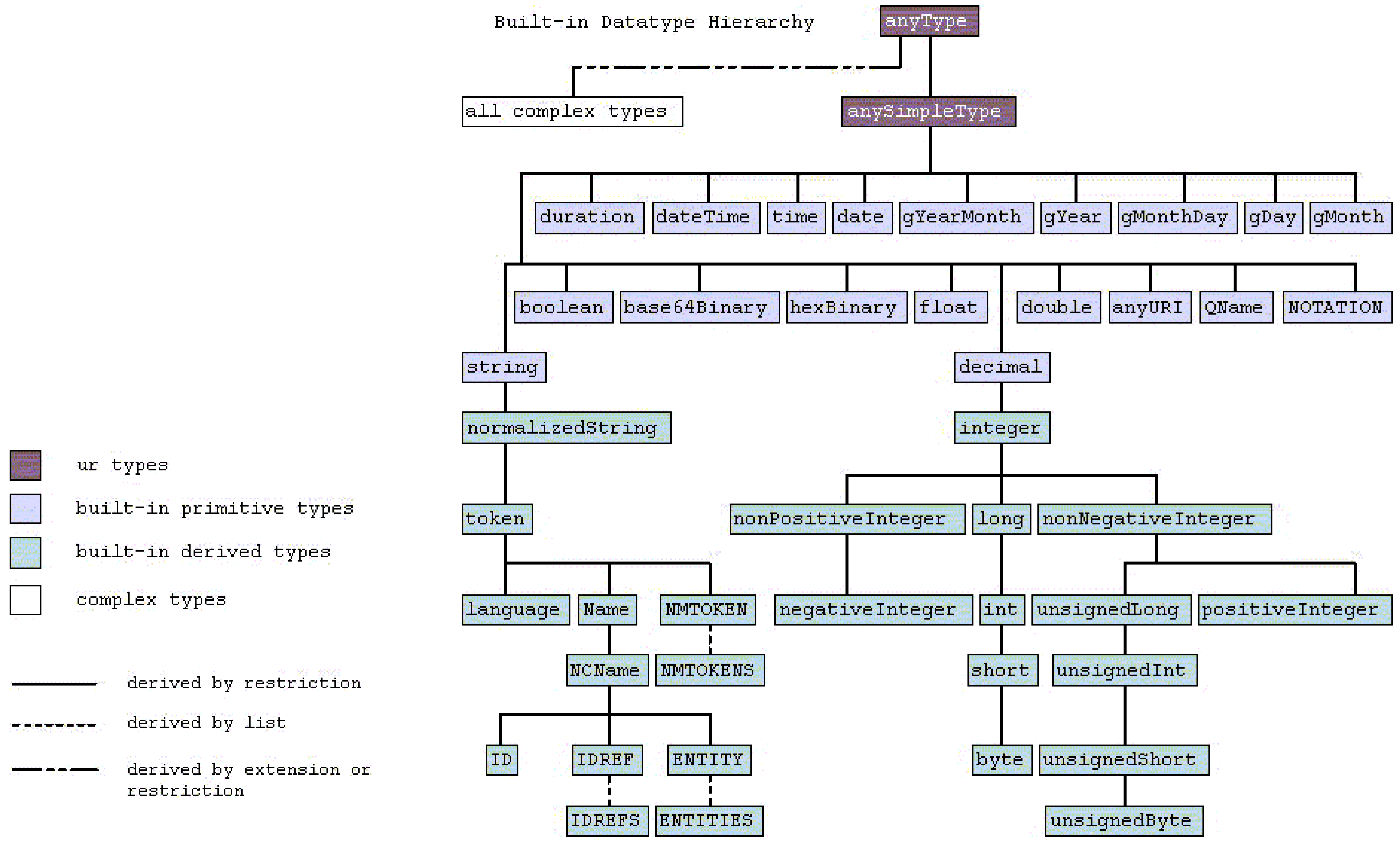
*“XML Schema (…) does not have an analog in the SGML world (…) was designed as an alternative to the DTD mechanism; **it adds stronger typing and utilizes XML syntax**. XML Schema contains additional constructs that make it **a lot easier to define sophisticated constraints** (…) supports (…) **‘complex’ and ‘simple’ types**. Simple types are defined by imposing additional constraints on built-in types, while complex types are composed recursively from other complex and simple types. **Built-in types available in the XML Schema are much richer and more flexible than those available in the DTD context**. This is very important because it means that the XML Schema makes it possible to express sophisticated constraints without resorting to application logic, which must be coded using a procedural language (e.g. Java) and is much more expensive to maintain.”*

page 91

XML Schema vs XML DTD

- XML schemas are written in XML while DTD are derived from SGML syntax.
- XML schemas define datatypes for elements and attributes while DTD doesn't support datatypes.
- XML schemas allow support for namespaces while DTD does not.
- XML schemas define number and order of child elements, while DTD does not.
- XML schemas can be manipulated on your own with XML DOM but it is not possible in case of DTD.
- using XML schema user need not to learn a new language but working with DTD is difficult for a user.
- XML schema provides secure data communication i.e. sender can describe the data in a way that receiver will understand, but in case of DTD data can be misunderstood by the receiver.
- XML schemas are extensible while DTD is not extensible.

XML Schema - primitive data types



XML Schema - Constraints for data types

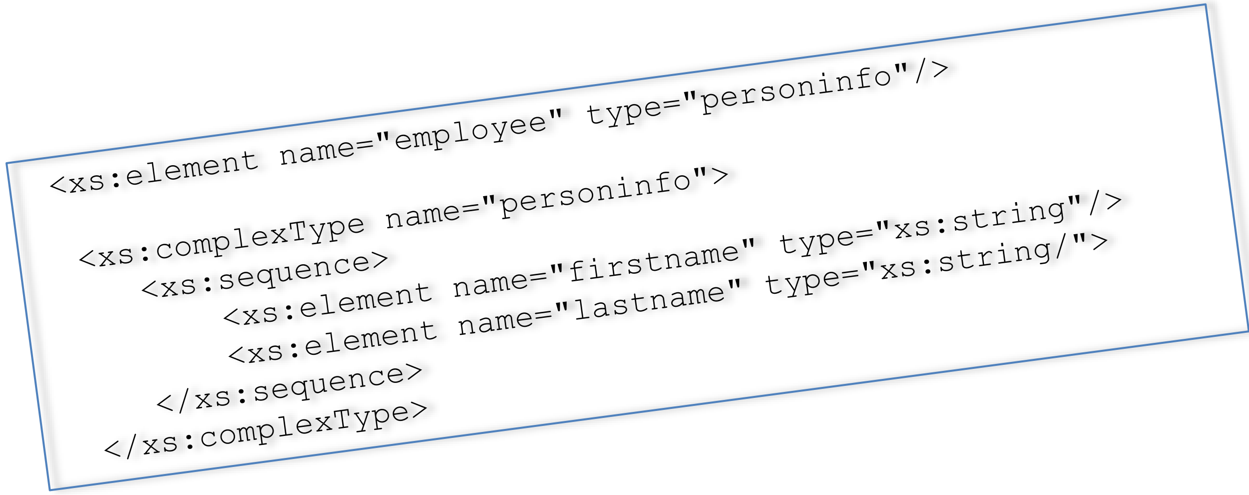
Restriction	Description
enumeration	Define a list of valid values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the maximum value for numeric values (value must be less than this value)
maxInclusive	Specifies the maximum value for numeric values (value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or items allowed. Must be equal to or greater than zero
minExclusive	Specifies the minimum value for numeric values (value must be greater than this value)
minInclusive	Specifies the minimum value for numeric values (value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters allowed
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how empty characters (tabs, spaces and carriage returns) are handled

XML Schema - Complex XML Elements

A complex element is an XML element that contains other elements and/or attributes.

There are four types of complex elements:

- empty elements
- elements that contain only other elements
- elements that contain text only
- elements that contain both other elements and text



```
<xs:element name="employee" type="personinfo"/>

 <xs:complexType name="personinfo">
    <xs:sequence>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string/">
    </xs:sequence>
 </xs:complexType>
```

Most common data types in XML Schema

- xs:string
- xs:decimal
- xs:integer
- xs:boolean
- xs:date
- xs:time
- …

Elements can have default values and fixed values.

- If nothing is specified, the attribute “valid” is set to true.

```
<xs:element name="valid" type="xs:string" default="true"/>
```

- You cannot change a value that is set to “fixed”.

```
<xs:element name="valid" type="xs:string" fixed="true"/>
```

Attributes may be optional or required.

```
<xs:attribute name="height" type="xs:string" use="optional"/>

<xs:attribute name="height" type="xs:string" use="required"/>
```

Restrictions

- You can specify constraints for the values of elements or attributes.

  Example: The value of the “maximum_speed" element must be between 50 and 120 inclusive.

```
<xs:element name="maximum_speed">
  <xs:simpleType>
    <xs:restriction base="xs:integer">
      <xs:minInclusive value="50"/>
      <xs:maxInclusive value="120"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions (cont.)

- Values can be restricted to a enumerated set.

  Example: only accept values **Yes**, **No**, **Maybe**.

```
<xs:element name="answer">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:enumeration value="Yes"/>
      <xs:enumeration value="No"/>
      <xs:enumeration value="Maybe"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Restrictions (cont.)

- Values can be restricted by a pattern.

  Example: define the "first_name" element that should only allow na uppercase letter followed by lowercase letters.

```
<xs:element name="first_name">
  <xs:simpleType>
    <xs:restriction base="xs:string">
      <xs:pattern value="[A-Z][a-z]+"/>
    </xs:restriction>
  </xs:simpleType>
</xs:element>
```

Are there any restrictions that can be expressed using an XML Schema but not with an DTD? Give some examples.

What would be an XML Schema for this document?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<note xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="example.xsd">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

```
<xs:element name="note">

<xs:complexType>
  <xs:sequence>
    <xs:element name="to"      type="xs:string"/>
    <xs:element name="from"    type="xs:string"/>
    <xs:element name="heading" type="xs:string"/>
    <xs:element name="body"    type="xs:string"/>
  </xs:sequence>
</xs:complexType>

</xs:element>
```

- In XML, tags are not predefined. We have to define our tags.

Is it True?

- Define a DTD and a Schema to validate the following XML!

```
<athlete>
<name>Mantorras</name>
<age>44</age>
<birthdate>1979-03-27</birthdate>
</athlete>
```

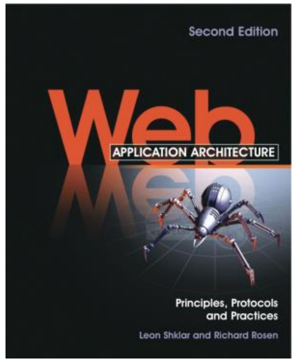
- XML DTD

```
<!ELEMENT athlete (name, age, birthdate)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT age (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
```

- XML Schema

```
<xs:element name="athlete">
     <xs:complexType>
          <xs:sequence>
          <xs:element name="name" type="xs:string"/>
          <xs:element name="age" type="xs:integer"/>
          <xs:element name="birthdate" type="xs:date"/>
          </xs:sequence>
     </xs:complexType>
</xs:element>
```

Bibliography


Second Edition
Web
APPLICATION ARCHITECTURE
Principles, Protocols and Practices
Leon Shklar and Richard Rosen

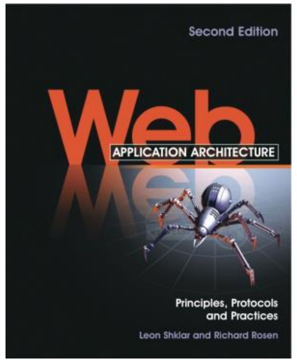
Shklar, Leon & Rosen, Rich (2009). *Web Application Architecture: Principles, Protocols and Pratices*. Chichester, Reino Unido: John Wiley & Sons.

Pages: 85 to 96

Chapter 5

5.1 CoreXML
5.2 XHTML

Next class


Second Edition
Web
APPLICATION ARCHITECTURE
Principles, Protocols and Practices
Leon Shklar and Richard Rosen

Shklar, Leon & Rosen, Rich (2009). Web Application Architecture: Principles, Protocols and Pratices. Chichester, Reino Unido: John Wiley & Sons.

Pages: 100 to 119

Chapter 5

5.3 Web Services

5.4 XSL