NPRG036

XML Technologies

Lecture 1

Introduction, XML, DTD

19. 2. 2018

Author: **Irena Holubová**
Lecturer: **Martin Svoboda**

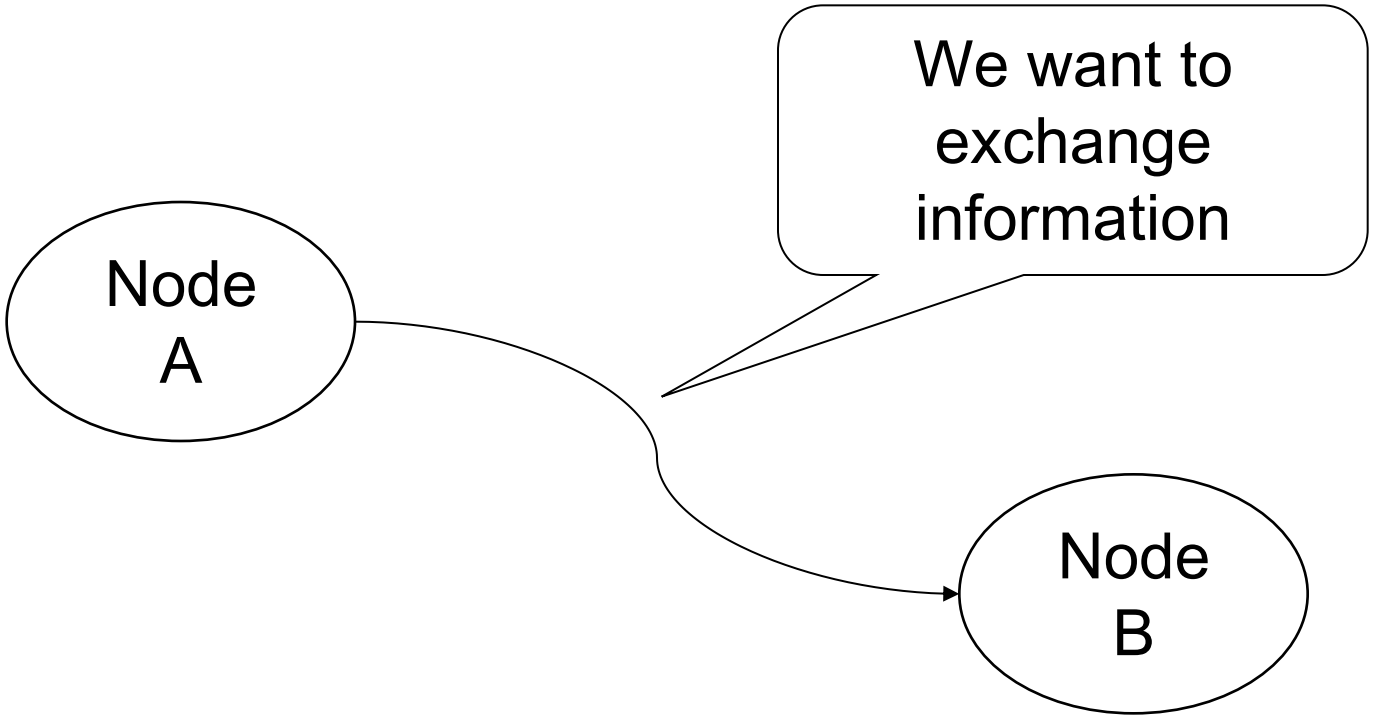
http://www.ksi.mff.cuni.cz/~svoboda/courses/172-NPRG036/

Lecture Outline

- Introduction
- **XML**
- XML technologies
- **DTD**

Introduction to XML format

Motivation



E.g. we want to send a message…

```
Tim Berners-Lee,
Robert Cailliau

Hi!
  My Internet does not work!
Steve J.

P.S. Help me!
```

… as a unstructured text?

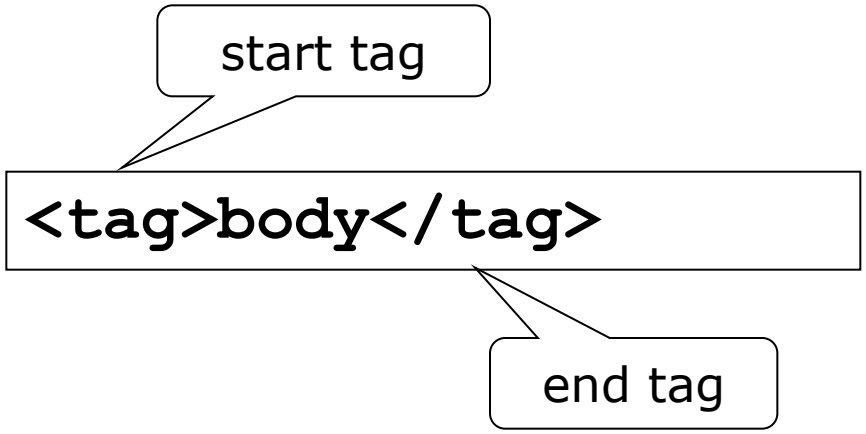
```
Tim Berners-Lee, Robert Cailliau Hi!
My Internet does not work! Steve J.
P.S. Help me!
```

… as a unstructured text?

```
Tim Berners-Lee, Robert Cailliau Hi!
My Internet does not work! Steve J.
P.S. Help me!
```

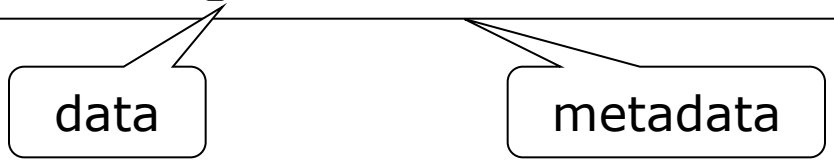
But how to find out (automatically)
who sends the message?

Let us introduce tags…



We can tag parts of the message…

```
<address>Tim Berners-Lee</address>
<address>Robert Cailliau</address>
<intro>Hi!</intro>
<text>My Internet does not work!</text>
<signature>Steve J.</signature>
<PS>Help me!</PS>
```

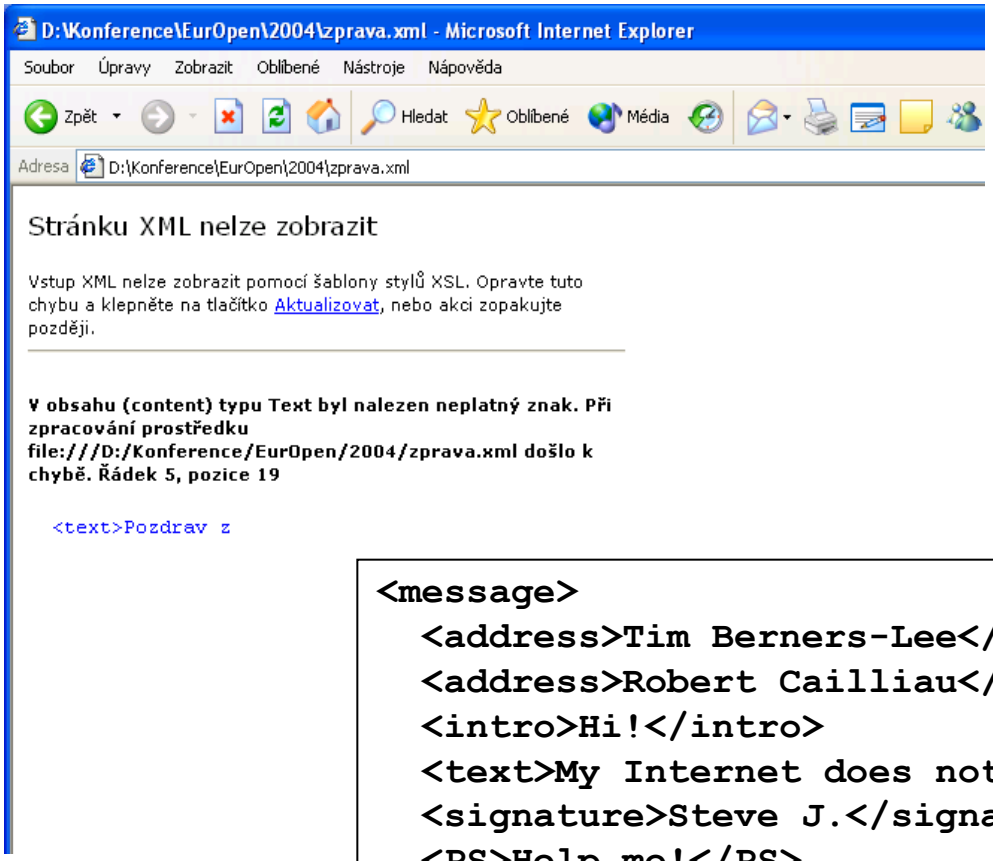


And the whole message…

```
<message>
  <address>Tim Berners-Lee</address>
  <address>Robert Cailliau</address>
  <intro>Hi!</intro>
  <text>My Internet does not
work!</text>
  <signature>Steve J.</signature>
  <PS>Help me!</PS>
</message>
```

In general to process the data automatically

To show the correct content in a browser it is not sufficient…

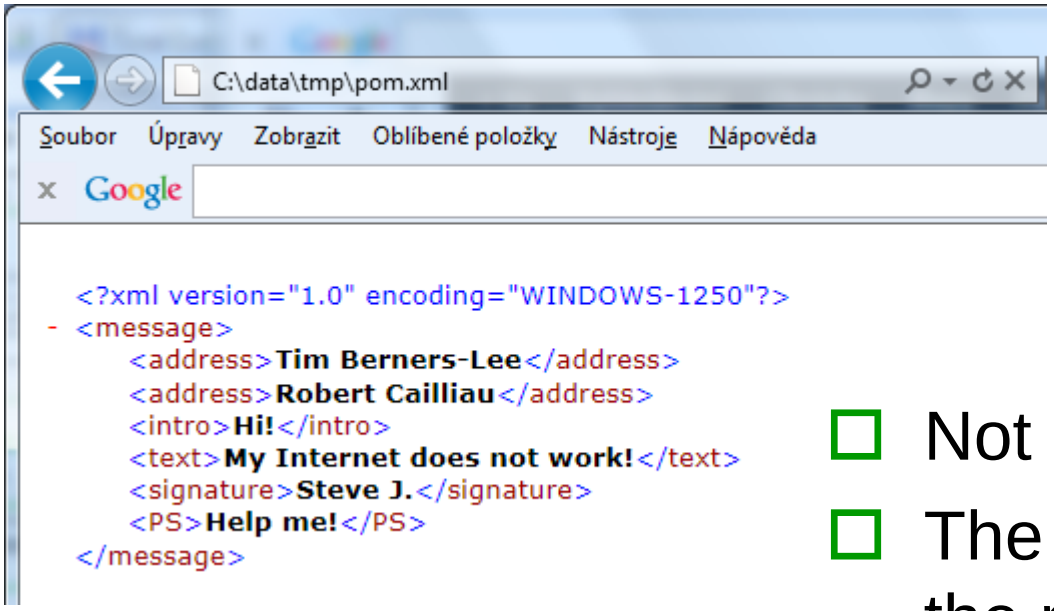



```
<message>
  <address>Tim Berners-Lee</address>
  <address>Robert Cailliau</address>
  <intro>Hi!</intro>
  <text>My Internet does not  work!</text>
  <signature>Steve J.</signature>
  <PS>Help me!</PS>
</message>
```

We need more information

- Format:
  - XML + version
- Encoding:
  - By default the document is in ISO 10646 (Unicode)
  - To communicate with the whole world we can use UTF-8
    - Compatible with ASCII
    - Contains all characters of all languages
  - For the Czech language we have ISO-8859-2 or Windows-1250

Better, but…

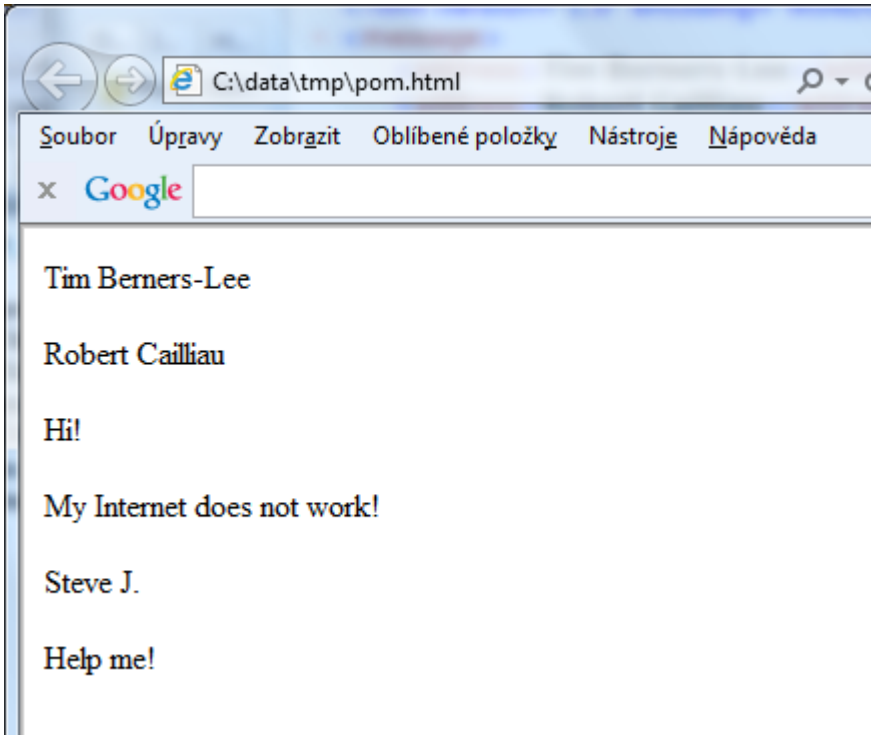


- Not much user friendly
- The browser shows also the meta data

We can, e.g., transform the document into HTML

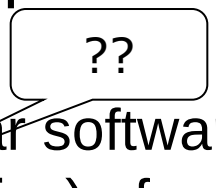
```
<html encoding="windows-1250">
  <head>
    <title>Message from: Steve J.</title>
  </head>
  <body>
    <p>Tim Berners-Lee</p>
    <p>Robert Cailliau</p>
    <p>Hi!</p>
    <p>My Internet does not work!</p>
    <p>Steve J.</p>
    <p>Help me!</p>
  </body>
</html>
```

Now the browser “knows” what to do with the data



What is the general aim?

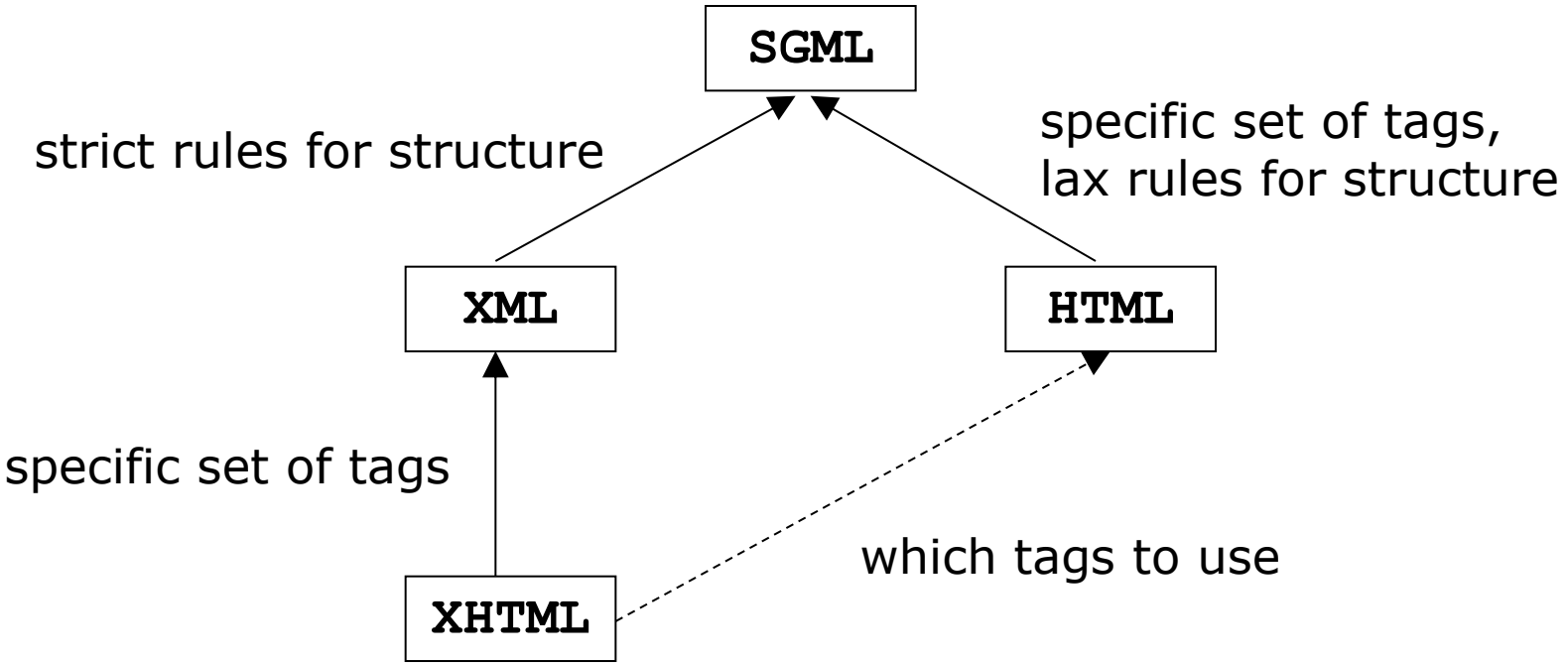
- Pure data are hard to process automatically
- We need to:
  - Ensure that a particular software understands the data
  - Add meaning (semantics) of particular data fragments
- E.g. HTML – describes visualization of data for an HTML browser
  - Problem 1: What if we are not interested just in visualization?
  - Problem 2: HTML has lax rules for structure
    - Complex processing
- Solution: XML



XML

- XML (eXtensible Markup Language) is a format for transfer and exchange of general data
  - Extensible Markup Language (XML) 1.0 (Fifth Edition)
    - http://www.w3.org/TR/xml/
  - Extensible Markup Language (XML) 1.1 (Second Edition)
    - http://www.w3.org/TR/xml11/
- XML is a subset (application) of SGML (Standard Generalized Markup Language - ISO 8879) – from 1986
- XML does not deal with data presentation
  - It enables to tag parts of the data
  - The meaning of the tags depends on the author
    - Presentation is one possible example

SGML vs. XML vs. HTML vs. XHTML



XML Document

- XML document is well-formed, if:
  - It has introductory prolog
  - Start and end tags nest properly
    - Each element has a start and an end tag
    - Corresponding tags have the same name (case sensitivity)

      <a></A>
    - Pairs of tags do not cross

      <a>
    - The whole document is enclosed in a single root element

Prolog

- An information for the SW that it works with an XML document
  - It must contain declaration of XML version
    - We have 1.0 and 1.1
  - It can contain information about encoding and if the document is standalone
- Version:

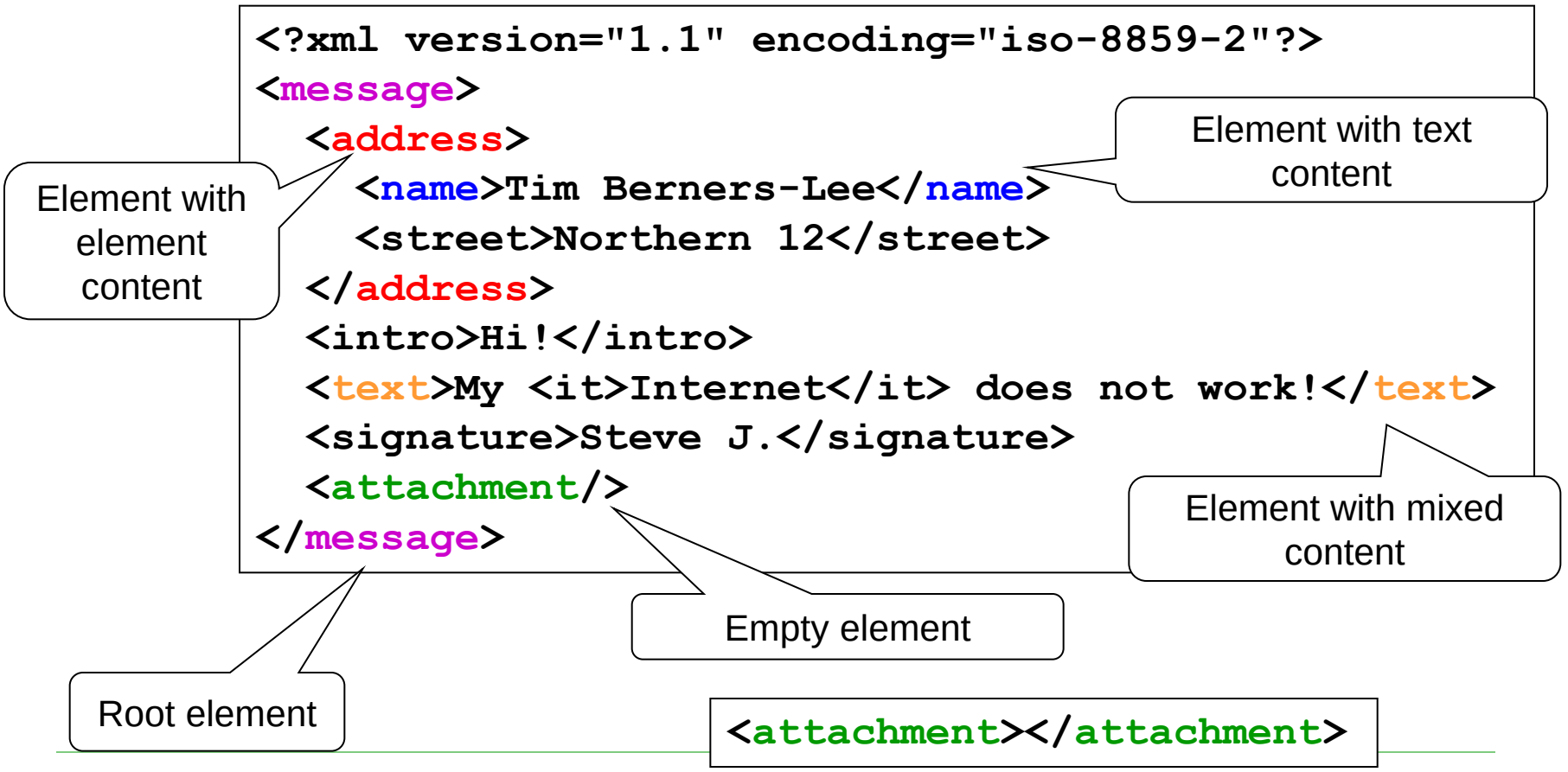
  <?xml version="1.1"?>
- Encoding other than UTF-8:

  <?xml version="1.1" encoding="iso-8859-2"?>
- Standalone document:

  <?xml version="1.1" standalone="yes"?>

always lowercase

Elements


<?xml version="1.1" encoding="iso-8859-2"?>
<message>
<address>
<name>Tim Berners-Lee</name>
<street>Northern 12</street>
</address>
<intro>Hi!</intro>
<text>My <it>Internet</it> does not work!</text>
<signature>Steve J.</signature>
<attachment/>
</message>
Element with element content
Element with text content
Element with mixed content
Empty element
Root element
<attachment></attachment>

Attributes

```
<?xml version="1.1" encoding="iso-8859-2"?>
<message>
  <address>
    <name>Tim Berners-Lee</name>
    <street>Northern 12</street>
  </address>
  <intro>Hi!</intro>
  <text>My <it>Internet</it> does not work!</text>
  <signature>Steve J.</signature>
  <attachment fig="image01.jpg"/>
</message>
```

Element with an attribute

Attribute name

Attribute value

Other Items of XML Document

```
<?xml version="1.1" encoding="iso-8859-2"?>
<message>
  <!-- to whom the message should be sent? -->
  <address>Jan Amos</address>
  <text>
    <![CDATA[
        for (i=0; i < 10; i++)
            {
            document.writeln("<p>Hi!</p>");
            }
    ]]>
  </text>
  <signature>Steve J.</signature>
  <date><?php echo Date("d.m.Y")?></date>
</message>
```

Comment

CDATA section

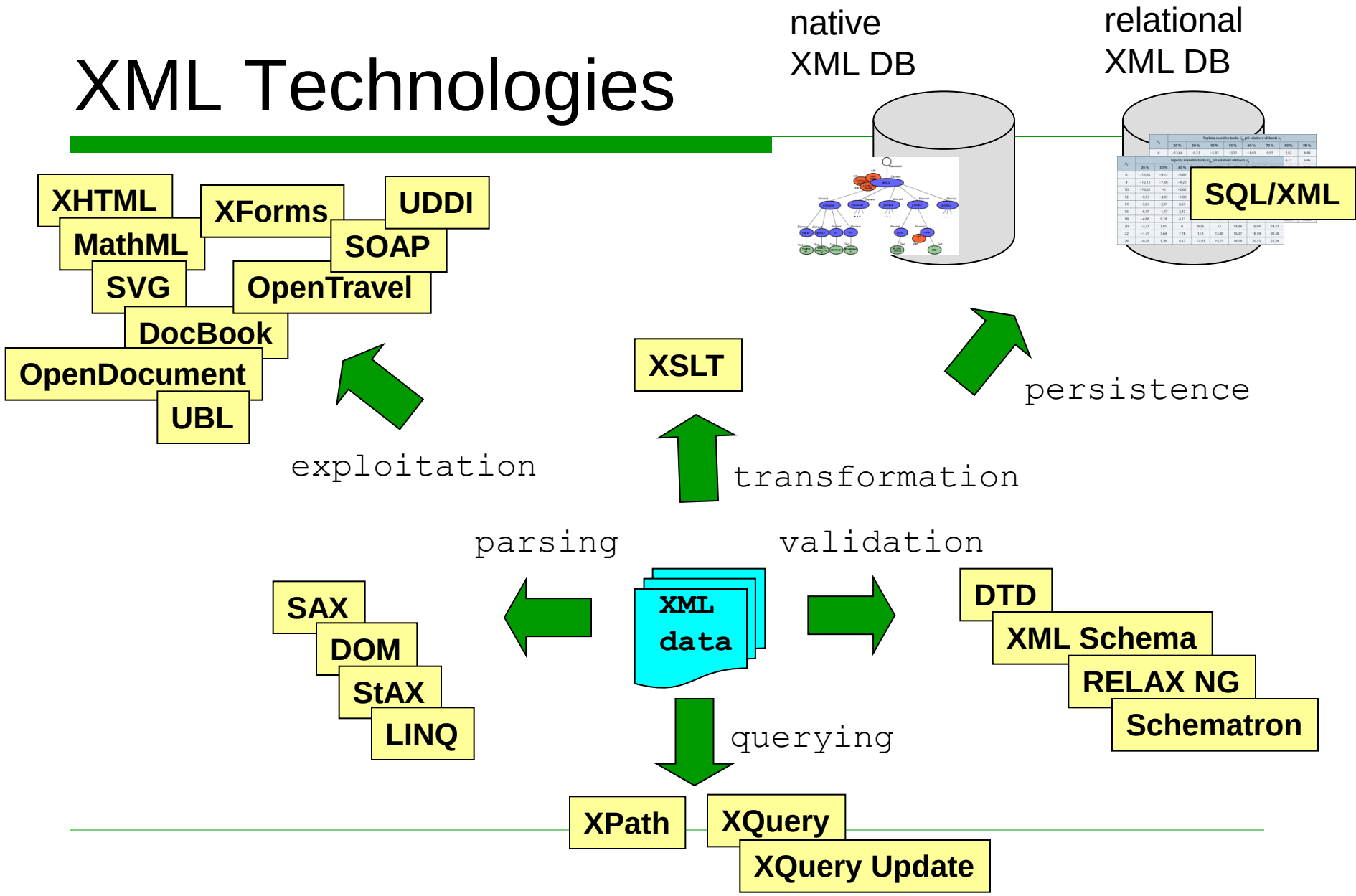
Processing instruction

XML Technologies

- XML is not only about the tags
  - XML = basic format for data description
    - XML documents
- XML technologies = a family of technologies to process XML data
  - Description of allowed content, data interface, parsing of data, information extraction (querying), transformation into other formats,…
    - W3C (WWW Consortium) standards
- Efficient implementation of the standards
  - Parsers, validators, query evaluators, XSL transformers, data persistence, …
- Standard XML formats
  - Where XML is used

http://www.w3.org/

XML Technologies



DTD

DTD

- Problem: Well-formedness is insufficient
  - We need to restrict the set of tags and their content
- Document Type Definition (DTD) describes the structure (grammar) of an XML document
  - Using regular expressions
- Valid XML document = well-formed XML document corresponding to a given grammar
  - There are also other languages – XML Schema, Schematron, RELAX NG, …

Structure of a Valid Document

```
<?xml version="1.0" ?>
<!DOCTYPE root-element [
...
]>
<root-element> ... </root-element>
```

Declaration of a document type

- Can be internal (grammar specified within DOCTYPE) or external (a reference to a separate file with the grammar)
  - There is no significant use for internal rules
    - Usually only for testing
  - Both can be used at the same time
    - Internal declarations have higher priority

Example: external and internal DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE greeting [
    <!ELEMENT greeting (#PCDATA)>
]>
<greeting>Hello, world!</greeting>
```

```
<?xml version="1.0"?>
<!DOCTYPE greeting SYSTEM "greeting.dtd">
<greeting>Hello, world!</greeting>
```

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html> … </html>
```

PUBLIC “public identifier" "URI"

Basic DTD Tags

- Document type declaration
  <!DOCTYPE … >
- Element type declaration
  <!ELEMENT … >
- Declaration of a list of attributes
  <!ATTLIST … >
- Declaration of an entity
  <!ENTITY … >
- Declaration of a notation
  <!NOTATION … >

upper case!!

Declaration of an Element Type

```
<!ELEMENT parent (child*)>
```

```
<parent>
  <child> ... </child>
  <child> ... </child>
  ...
</parent>
```

- Element name + declaration of allowed content
  - Empty, any, text, mixed, element

Declaration of an Element Type

,	… sequence
|	… selection
?	… iteration (0 or 1)
+	… iteration (1 or more)
*	… iteration (0 or more)

- Empty content
  <!ELEMENT attachment EMPTY>
- Any content
  <!ELEMENT container ANY>
- Text content
  <!ELEMENT surname (#PCDATA)>
- Mixed content
  <!ELEMENT text (#PCDATA | it | bold)*>
- Element content
  <!ELEMENT message (address, text)>

```
(name, (author | editor)?, p*, (title, p+)*)
```

Declaration of an Attribute

The order is in the document arbitrary

```
<!ATTLIST person number ID #REQUIRED
                 employed CDATA #FIXED "yes"
                 holiday (yes|no) "no">
```

- Attributes of element person
- Attribute number is a unique ID and compulsory (#REQUIRED)
- Attribute employed contains text (CDATA), it has a constant value (#FIXED) „yes“
- Attribute holiday can have one of the given values („yes“ or „no“), implicit value is „no“

???

Data Types of Attributes

- CDATA – arbitrary text string
- Enumeration of values
- ID – unique identifier (within the content of the document), it must be a string of letters, numbers and characters „-“, „_“, „:“, „.“, preferably in ASCII, it must start with a letter or character „_“
- IDREF – reference to an ID of any element in the document
- IDREFS – list of references (delimited with white spaces) to IDs
- NMTOKEN – string similar to ID, not unique, can start with a number
- NMTOKENS – list of NMTOKENs
- ENTITY – link to an external entity
- ENTITIES – list of links to external entities

later

Presence of Attributes

- #REQUIRED – the attribute is compulsory
- #IMPLIED – the attribute is optional
- #FIXED – the attribute has a fixed value

Entity Declaration

- In practice only the trivial cases are usually used
- Entity = association of a name and a value which is later (repeatedly) used
- Classification 1:
  - Parsed = the text which replaces the link to an entity becomes a part of the document
    - We refer using references
  - Unparsed = a resource which can contain anything (e.g. binary data = an image, a video)
    - We refer using attributes of type ENTITY/ENTITIES
    - It must be associated with a notation
- Classification 2:
  - General – in XML documents
  - Parameter – in DTDs
- Classification 3: Internal vs. external

later

Character Entities

- A possibility to insert a character with any code
  - Hexadecimal or decimal

```
Solve inequality 3x &#x3C; 5
```

- Pre-defined entities for special characters

```
Solve inequality 3x &lt; 5
```

&	... amp
<	... lt
>	... gt
'	... apos
"	... quot

General Entity

- Internal (hence of course parsed) entity
  - Usage: Repeating parts of XML documents

```
<!ENTITY status "working draft">
```

```
<note>The current status of the
document is &status;</note>
```

- External parsed entity
  - Usage: Modularity of XML documents

```
<!ENTITY xml-serial SYSTEM "xml-serial.txt">
```

General Entities

- External **un**parsed entity
  - Usage: Reference to non-XML data

or PUBLIC

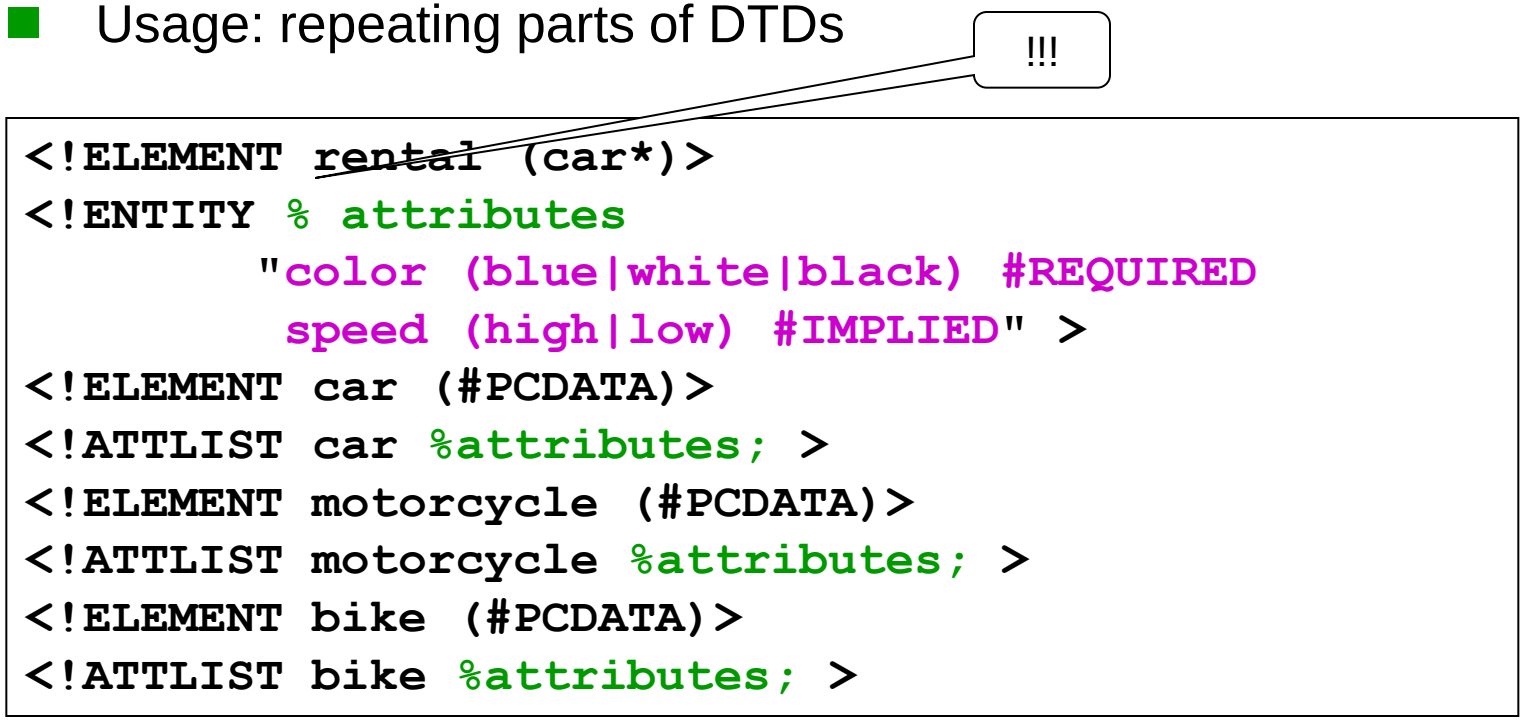
Declaration of a notation

```
<?xml version="1.0" encoding="windows-1250"?>
<!DOCTYPE message [
  <!NOTATION avi SYSTEM
      "C:/Program Files/Video Player/Player.exe">
  <!ENTITY video SYSTEM "video.avi" NDATA avi>
  <!ELEMENT video-holiday (#PCDATA)>
  <!ATTLIST video-holiday src ENTITY>
]>

<message>I enclose the <video-holiday
src="video">video</video-holiday> from
holiday.</message>
```

Parameter Entity

- Internal entity
  - Usage: repeating parts of DTDs



Parameter Entity

- External entity
  - Usage: Modularity of DTDs

```
<!ENTITY % ISOLat2 SYSTEM "iso-pub.ent">
...
%ISOLat2;
...
```

Conditional Sections

```
<!ENTITY % draft 'INCLUDE' >
<!ENTITY % final 'IGNORE' >

<![%draft;[
<!ELEMENT book (comments*, title, body,
supplements?)>
]]>
<![%final;[
<!ELEMENT book (title, body, supplements?)>
]]>
```

DTD – Bigger Example

```
<!ELEMENT employees (person)+>
<!ELEMENT person (name, email*, relations?)>
    <!ATTLIST person id ID #REQUIRED>
    <!ATTLIST person note CDATA #IMPLIED>
    <!ATTLIST person holiday (yes|no) "no">
<!ELEMENT name ((first, surname)|(surname, first))>
<!ELEMENT first (#PCDATA)>
<!ELEMENT surname (#PCDATA)>
<!ELEMENT email (#PCDATA)>
<!ELEMENT relations EMPTY>
    <!ATTLIST relations superior IDREF #IMPLIED>
    <!ATTLIST relations subordinates IDREFS #IMPLIED>
```