<?xml?>

# XML – Part A

## Introduction, Specification

# Part 1 – What is XML?

# Question 1

**What does XML stand for?**

a) eXchange Modern Links
b) eXample Markup Language
c) eXtensible Markup Language
d) eXchange Management Language

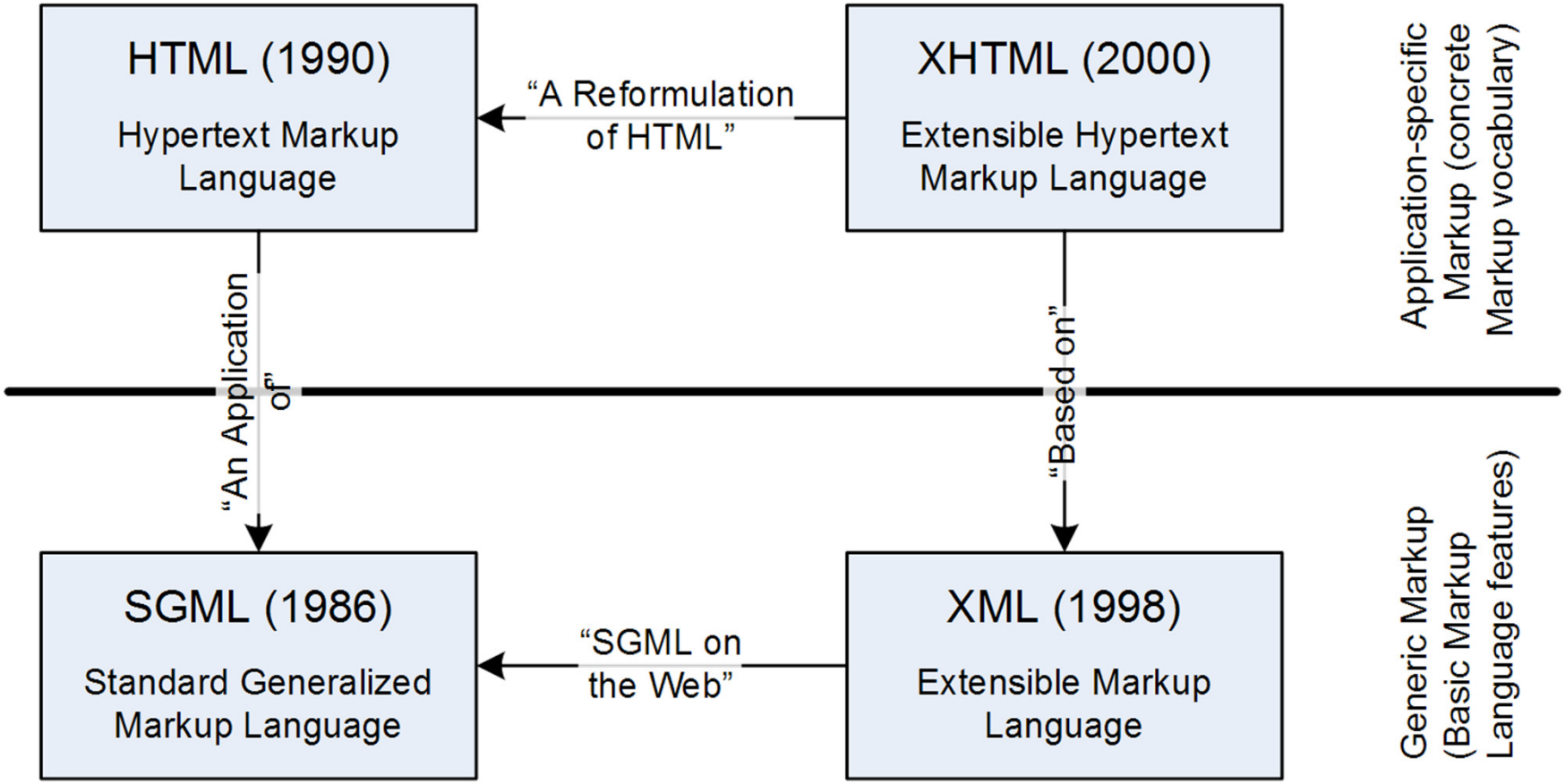
# Question 1

**What does XML stand for?**

a) eXchange Modern Links
b) eXample Markup Language
c) eXtensible Markup Language
d) eXchange Management Language

**ALR1** Laube-Rosenpflanzer Annett; 28.01.2013

# Entwicklung von XML und HTML



# Question 2

**What is XML ?**

a) a programming language
b) a formatting language
c) a markup language
d) a scripting language

# Question 2

**What is XML ?**

a) a programming language
b) a formatting language
c) a markup language
d) a scripting language

# XML main characteristics

- It is a textual representation of data and/or text containing markup information.

- As opposed to formatting languages, this markup names the text elements or data and describes their meaning rather than their formatting (as for example in HTML).

# Question 3

**XML …**

a) has to be viewed and changed with special programs (editors, viewers)
b) is easy to understand
c) can be processed only with Java
d) self explaining

# Question 3

**XML …**

a) has to be viewed and changed with special programs (editors, viewers)
b) is easy to understand
c) can be processed only with Java
d) self explaining

# Question 4

**XML …**

a) has a fix number of tags and labels
b) is extendable
c) both

# Question 4

**XML …**

a) has a fix number of tags and labels
b) is extendable
c) both

# Question 5

**XML is used for …**

a) security protocols
b) text documents
c) UML diagrams
d) school certificates

# Question 5

**XML is used for …**

a) security protocols
b) text documents
c) UML diagrams
d) school certificates

And more….
XML is used for all kind of structured data.

# XML languages

SAML v2.0
Security Assertion Markup Language

XMI

PMML
Predictive Model Markup Language

XHTML

SOAP

WSDL

LaTeX

See more:
http://en.wikipedia.org/wiki/List_of_XML_markup_languages

# Hello World

```
<?xml version="1.0"?>
<world>
   <hello>Hello World!!!</hello>
</world>
```

Header

Nested structure: Start and end tags

# A first example

An address of a person looks like the following example:

John Smith
Master of Arts
12, 45th Avenue, Apt #15
New York, NY 10003
USA

Human beings easily understand that the first line contains the name, the subsequent one „apparently“ an academic title, the next one a street and an apartment number, and finally a city, followed by its ZIP code, and by a country indication.

However, this ordering isn’t applicable everywhere (cf. Russia = ± inverse order, Europe: ZIP in front of city etc.).

If this information has to be treated automatically, we need some means to recognize formally the parts of this address.

# A first example

Using XML this address may be written as follows:

```
<address>
  <firstname>John</firstname> <name>Smith</name>
  <title>Master of Arts</title>
  <street>45th Avenue</street> <number>12</number>
  <apartment-number>15</apartment-number>
  <city>New York</city> <zip>NY 10003</zip>
  <country>USA</country>
</address>
```

What's going on here?

- Contiguous data/values are grouped by <xxx> … </xxx>
- Such structures may be nested
- The names in the <xxx> tags may be selected by the user; the only condition is that the tag's content <xxx> must be the same as in </xxx>.

# Basic XML Notions

<xxx> is called a *tag.*
<xxx> is the *start tag*, </xxx> the corresponding *end tag*

A corresponding pair of <xxx> … </xxx> including their enclosed content is an *element.*

If a corresponding pair of <xxx> … </xxx> contains neither text nor other elements, it can be written as <xxx/> (empty element)

The whole ensemble is called a *document*; thus, a document consists of one single element (the *root element*) with its contents

(However, a complete document contains some formal elements in addition.)

# XML Naming Conventions

XML elements have to be named according the following rules

- Names start with letters or underscores (_)
- After the first character, numbers, hyphens (-), underscores (_) and periods (.) are allowed
- Names can't contain spaces
- Names can't contain the colon (:)
- Names can't start with the letters xml, in uppercase, lowercase or mixed

# XML Naming Conventions

| Name | Correct | Not correct | Why |
|---|---|---|---|
| <first.name> | | | |
| <xml-tag> | | | |
| <123> | | | |
| <a-b> | | | |
| <résumé> | | | |
| <fun=xml> | | | |
| <my tag> | | | |
| <CaMeLcAsE> | | | |

# XML Attributes

Values may not only be represented as the *content* of a *tag* but also as an *attribute*

```
<book signature="DR20-4537" >
  <author>
      <firstname>Daniel</firstname>
      <name>Defoe</name>
  </author>
   <title>Robinson Crusoe</title>
   <chapter title="Embarking" >
     Here starts the text of the first chapter.
   </chapter>
   <chapter title="Afloat" >
     This is the second chapter.
   </chapter>
   ...
</book>
```

Note:
Attribute
Values must
be quoted !!!

Tag content ↔ Attribute ?
No fixed rule; empirical decision

# XML Comments

Between the data elements we may also insert comments:

```
<book signature="DR20-4537" >
  <!-- Data about the book as a whole -->
  <author>
     <firstname>Daniel</firstname>
     <name>Defoe</name>
  <author>
  <title>Robinson Crusoe</title>

  <!-- Text of the book -->
  <chapter title="Embarking" >
    Here starts the text of the first chapter.
  </chapter>
  <chapter title="Afloat" >
    This is the second chapter.
  </chapter>
  ...
</book>
```

# The XML Header

At the beginning of each xml document at least the following header line must be added:

```
<?xml version="1.0" encoding="UTF-8"?>
<book signature="DR20-4537" >
   <!-- Data about the book as a whole -->
   <author>
      <firstname>Daniel</firstname>
       <name>Defoe</name>
   </author>
   <title>Robinson Crusoe</title>

   <!-- Text of the book -->
   <chapter title="Embarking" >
    Here starts the text of the first chapter.
   </chapter>
   <chapter title="Afloat" >
    This is the second chapter.
   </chapter>
   ...
</book>
```

# The XML Header

```
<?xml version="1.0" encoding="UTF-8"
                                  standalone="no"?>
```

There are versions 1.0 and 1.1;
however, v 1.1 is not well accepted,
And v 1.0 is still preferred.

Rarely used;
Values are yes or no (default).
Yes means there is no external DTD.

There are several encoding standards;

- UTF-8 and UTF-16: Standardized by ISO/IETF, used worldwide, represents full Unicode, using variable-length multibyte coding
- ISO-8859-1: used mostly in US, ≈ extended ASCII
- Shift-JIS: used mostly in Japan, for Japanese alphabet.

Be aware of your editors or processing programs since they might tacitly assume a (different!!) standard!

# Exercise 1-1

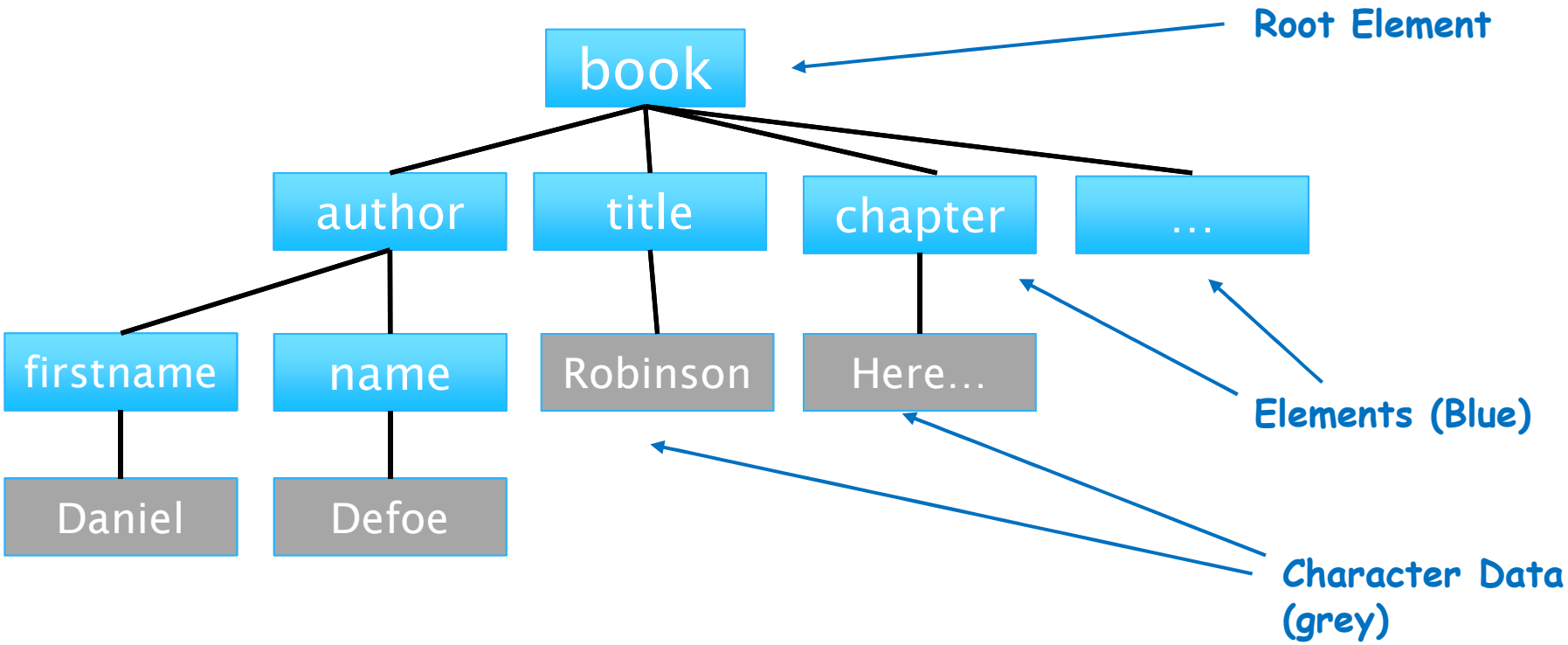
Write down the data characterizing a bank account as an XML document.

The following data should be contained:

- account number
- name and address of the owner
- account status
- type of account (current account, savings account etc.).

If you see further data to be enclosed … go ahead!

# XML documents have a tree structure




An XML document is an **ordered, labeled tree**

# XML documents have a tree structure

An XML document is an **ordered, labeled tree**, consisting of

- **elements** nodes, each labeled with
  - a name (often called the element *type*),
  - (optionally) a set of **attributes**, each consisting of a name and a value,
  - and these nodes can have child nodes
- **character data** leaf nodes contain the actual data (text strings)
  - normally, character data nodes are **non-empty** and **non-adjacent to other character data nodes**

# Representing Relations between Data

Representation of data =

- text values between element tags, or
- values of attributes

Representing relations between data =

- Sequence

```
<author>Anthony First</author>
<author>John Second</author>
```

sequence = equivalent + ordered

- Nesting

nesting = „is part of"

```
<author>
  <email>Maggie.Herrick@bbs.mhv.net</email>
</author>
```

# XML Entities

What if you want to write a value containing a < sign, e.g.

```
<condition> e < 2 </condition> ?
```

XML defines some standard ***entities*** to avoid this problem:

| | | |
|---|---|---|
| `<` | `<`, | a „less than“ sign |
| `>` | `>`, | a „greater than“ sign |
| `"` | `"`, | a „double quote“ sign |
| `'` | `'`, | a „single quote“ or „apostroph“ sign |
| `&` | `&`, | an „ampersand“ sign |

```
<condition> e &lt; 2 </condition>
```

XML processors (→ XSLT) will automatically convert these entities to the actual characters for output but NOT treat them according to their normal XML meaning

Users may define their own entities (→ DTD/Schema)

# Namespaces

What‘s the Problem?

Since the tag names and document structures may be defined specifically for each application, the **same tag names** might get used to denote **different things**. Joining such documents will lead to clashes:

| | |
|---|---|
| Geometry: | <plane> … </plane> |
| Air traffic: | <plane> … </plane> |
| Geography: | <plane> … </plane> |

# Namespaces

The solution

In order to avoid such name clashes, we define name spaces and we prefix each tag name with the namespace indicator:

Geometry: <geom:plane> … </geom:plane>

Air traffic: <fly:plane> … </fly:plane>

Geography: <geog:plane> … </geog:plane>

**HOWEVER:**
**These namespace indicators (geom, fly …) really must be different! How do we meet this requirement?**

# Namespaces

How can we assure disjunct namespaces ?

Using „geog“ is just a shorthand form for a longer name, and it is this longer name which really must be different from any other. The idea is to use URLs from the Internet which indeed are guaranteed to be unique:

```
<?xml version="1.0" encoding="UTF-8"?>
<flight-geometry
  xmlns:geom="http://www.mycompany.com/homepage/subpage"
  xmlns:fly="http://www.anyairline.com/homepage/subpage">
...
```

Shorthand name of the namespace

Any/my existing URL as a guarantee for uniqueness...

... possible extended by some (fictitious) page/subpage name

# Namespaces

An example of the usage of namespaces:

```
<?xml version="1.0" encoding="UTF-8"?>
<flight-geometry
  xmlns:geog="http://www.belpmoos.com/homepage/subpage"
  xmlns:fly="http://www.anyairline.com/homepage/subpage">

  <rule>
    The Berne airport is located in the <geog:plane> Belpmoos
    </geog:plane>. <fly:plane> Jets </fly:plane> are always
    landing from North while <fly:plane> small planes
    </fly:plane> may land from both sides.
  </rule>
  …
</flight-geometry>
```

# Default Namespaces

When many elements use the same namespace

→ Define a default namespace: attribute `xmlns`.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<flight-geometry
  xmlns="http://www.belpmoos.com/homepage/subpage"
  xmlns:fly="http://www.anyairline.com/homepage/subpage">

  <rule>
    The Berne airport is located in the <plane> Belpmoos
    <plane>. <fly:plane> Jets </fly:plane> are always
    landing from North while <fly:plane> small planes
    </fly:plane> may land from both sides.
  </rule>
  …
```

# Links to more information

- Internet
  - **www.w3.org/TR/REC-xml.html**
    the XML 1.0 specification
  - **www.w3.org/TR/xml11**
    XML 1.1 (Candidate Recommendation), minor changes to reflect Unicode revisions
  - **www.w3.org/XML**
    W3C's XML homepage
  - **www.xml.com**
    XML information by O'Reilly: articles, software, tutorials
  - **www.oasis-open.org/cover**
    The XML Cover Pages: comprehensive online reference
  - **www.w3schools.com/xml**
    XML School: an XML tutorial
- Books
  - David Hunter et al.: **Beginning XML (Programmer to Programmer)** Wrox.
    (An excellent introductory book.)
  - Elliotte Rusty Harold, W. Scott Means: **XML in a Nutshell, Third Edition**. O'Reilly. (Reference, not a tutorial)
  - Daniel Koch: **XML für Webentwickler**. Ein praktischer Einstieg. Hanser 2010

# Exercise 1-2

Correct the given XML document.

```
<?xml version="1"?>
<document>
<-- Dieses Dokument enthält einige Fehler -->
<Information> Dieses Dokument
contains some < bold>information</bold>. </br> Wenn
es korrigiert wurde, kann es
mit Hilfe eines Parser eingelesen
werden.</Information>
</Document>
```