Introduction to XML

Justus Klingemann

What is this?

```
Johann Wolfgang von Goethe
Faust
```


Metadata for Meaning and Structure

```
<Book>
   <Author>
      Johann Wolfgang von Goethe
   </Author>
   <Title>
      Faust
   </Title>
</Book>
```


Or more detailed

```
<Book>
   <Author>
      <Name>
         <FirstName>Johann</FirstName>
         <FirstName>Wolfgang</FirstName>
         <Affix>von</Affix>
         <Surname>Goethe</Surname>
      </Name>
   </Author>
   <Title>
      Faust
   </Title>
</Book>
```

What is XML?

EXtensible Markup Language
- Erweiterbare Auszeichungssprache

Tim Bray: "XML will be the ASCII of the Web - basic, essential, unexciting"

Syntax for the description of structured information
- standardized by the W3C (World Wide Web Consortium)

Architecture for hypermedia based on this syntax

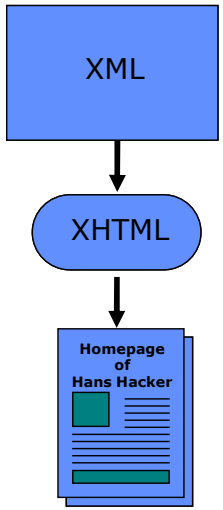
Difference to HTML
- HTML … Fixed set of elements with a particular structure
- XML … Architecture to define domain-specific document structures

Comparison HTML-XML

HTML
- Structure for presentation
- Meaning extern
- Browser can display HTML but it is not suited for processing the information contained in the document

XML
- Structure for meaning
  - Presentation extern
- Meaning represented in metainformation of markup
- Parser can process XML and make the information available for subsequent processing

XML Hierarchy



Meta-Language

Document-type "Schema"

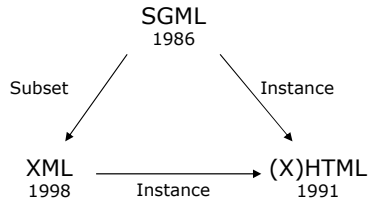
Dokument-instance

History of XML

SGML
- Standard Generalized Markup Language
- 1986 ISO-Standard
- Properties
  - Separation of content and formatting
  - very complex
  - Used only in the publishing domain

HTML
- Instance of SGML
- 1991 W3C Standard

XML
- Subset of SGML
- 1996 Design Principles for XML
- 1998 W3C Standard

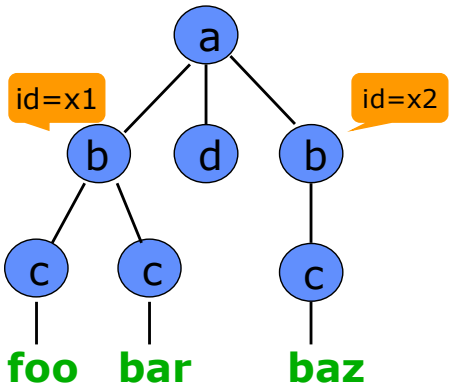


XML-Syntax: Elements + Attributes

```
<?xml version="1.0"?>

<a>
  <b id="x1">
    <c>foo</c>
    <c>bar</c>
  </b>
  <d/>
  <b id="x2">
    <c>baz</c>
  </b>
</a>
```

Physical Structure
"Instance"



Logical Structure
"Element-tree"

Distributed Systems

Composition of an XML-Document

Start with a prolog
- At least: **<?xml version="1.0"?>**

Each document has a tree-structure

Two dimensions
- parent-child (adjacent nodes)
- siblings (nodes with the same parent)

Siblings have an order

The nodes of the tree are called elements

There is exactly one root-element that encloses all other elements

Distributed Systems

Prolog

Each XML-Document starts with a prolog

minimal prolog: specification of the version number:
`<?xml version="1.0"?>`

Optional declarations
- Encoding: the used character set
- Standalone: whether there exist references to external documents

Example: `<?xml version="1.0" encoding="ISO-8859-1" standalone="yes" ?>`

In addition to this we can provide a DTD or a schema to specify a particular structure for the document

Distributed Systems

Elements

Elements are the basic building block of an XML-document

Elements start with a start-tag and end with an end-tag

Start-tags contain the name of the element that is enclosed within < and >

End-tags also contain the character / before the name of the element

Names are case-sensitive

Elements can contain
- other elements
- text: PCDATA (Parsable Character Data)

Syntax of an empty element: **<Name_of_element/>**

Attributes

Elements can be extended by means of attributes

The attribute is contained in the start-tag

- After the name of the element
- consists of
  - name of attribute
  - equal sign
  - value (in quotation marks)

Example

- ```
  <Book language=´´german´´>
  ...
  </Book>
  ```


When to use Element, when Attribute?

Sometimes we have the choice how to model our data

Instead of attributes we can use alternatively elements

Some design principles

- If the information has a complex structure: Element
- If the information has an own identity: Element
- Just a property of an elements: Attribute

XML Correctness

well-formed (wohlgeformt)

- A document is in line with the syntax rules of XML
- In particular:
  - Correct parenthesis: for each start-tag <T> we have to have an end-tag </T>
  - No overlap of elements
  - Each document has exactly one root-element
  - Unique attributes: each attribute exists at most once per element

valid (gültig)

- The document is well-formed and in line with specified schema
- A schema can be written by means of a document type definition (DTD)

DTD Element-Declaration (1)

<!ELEMENT elementname (content)>

Element as content

```
<!ELEMENT example ( a )>
```

Text as content

```
<!ELEMENT example (#PCDATA)>
```

mixed content

```
<!ELEMENT example (#PCDATA | a)*>
```

Empty element

```
<!ELEMENT example EMPTY>
```

Element with arbitrary content

```
<!ELEMENT example ANY>
```
```

DTD Element-Declaration (2)

Sequence

```
<!ELEMENT example     ( a, b )>
```

Alternative

```
<!ELEMENT example     ( a | b )>
```

Optional (zero or one)

```
<!ELEMENT example ( a )?>
```

Optional but repeatable (zero or more)

```
<!ELEMENT example ( a )*>
```

Required and repeatable (one or more)

```
<!ELEMENT example ( a )+>
```

Content model can be grouped by means of parentheses

DTD Attribute-Declaration

```
<!ATTLIST Elementname
                Attributename Type Default
                Attributename Type Default
                ...
>
```

Possible defaults

Mandatory attribute:	#REQUIRED
Optional attribute:	#IMPLIED
Fixed attribute value:	#FIXED “value"
Default value	“value"

DTD Attribute: Examples for Types

CDATA

- String
- `<!ATTLIST example width CDATA #REQUIRED>`

Enumeration

- Token from a predefined set of values, a default value can be specified
-
```
<!ATTLIST example
          choice ( yes | no | maybe ) "yes">
```

ID, IDREF, IDREFS

- ID is an identifier that is unique within the whole document
- IDREF is a reference to an ID
- Referential integrity is checked by a parser
-
```
<!ATTLIST example
  identity ID     #IMPLIED
  reference       IDREF #IMPLIED>
```


ID / IDREF

Example

```
<project responsible=´p0001´>
    <title>...</title>
</project>
<project responsible=´d0001´>
    <title>...</title>
</project>
...
<person id=´p0001´>
  <name>Meyer</name>
</person>
...
< devision div_id=´d0001´/>
```

```
<!ELEMENT project (title)>
<!ATTLIST project responsible IDREF
          #REQUIRED>

<!ELEMENT person (name)>
<!ATTLIST person id ID
          #REQUIRED>
...
<!ELEMENT devision (EMPTY)>
<!ATTLIST devision div_id  ID
          #REQUIRED>
```

Declaration of DTDs in Documents

External DTD Declarations

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE test PUBLIC "-//Test GmbH//DTD test V1.0//EN"
               SYSTEM "http://www.test.de/test.dtd">
<test> "test" is the document-element </test>
```

Internal DTD Declaration

```
<!DOCTYPE test [ <!ELEMENT test EMPTY> ]>
<test/>
```


Correctness? (1)

```
<!ELEMENT Auto (Bezeichnung, Motor, Kofferraum?)>
<!ELEMENT Bezeichnung (Typ, Marke?)>
<!ELEMENT Typ (#PCDATA)>
<!ELEMENT Marke (#PCDATA)>
<!ELEMENT Motor (PS | kW)>
<!ELEMENT PS (#PCDATA)>
<!ELEMENT kW (#PCDATA)>
<!ELEMENT Kofferraum (#PCDATA)>
<!ATTLIST Motor Treibstoff (Benzin | Diesel) #REQUIRED>


<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Auto SYSTEM "auto.dtd">
<Auto>
<Bezeichnung>
<Typ>Golf</Typ>
</Bezeichnung>
<Kofferraum>330</Kofferraum>
</Auto>
```


Correctness? (2)

```
<!ELEMENT Auto (Bezeichnung, Motor, Kofferraum?)>
<!ELEMENT Bezeichnung (Typ, Marke?)>
<!ELEMENT Typ (#PCDATA)>
<!ELEMENT Marke (#PCDATA)>
<!ELEMENT Motor (PS | kW)>
<!ELEMENT PS (#PCDATA)>
<!ELEMENT kW (#PCDATA)>
<!ELEMENT Kofferraum (#PCDATA)>
<!ATTLIST Motor Treibstoff (Benzin | Diesel) #REQUIRED>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Auto SYSTEM "auto.dtd">
<Auto>
<Bezeichnung>
<Typ>Golf</Typ>
</Bezeichnung>
<Kofferraum>330</Kofferraum>
<Motor Treibstoff="Benzin"><PS>75</PS></Motor>
</Auto>
```


Correctness? (3)

```
<!ELEMENT Auto (Bezeichnung, Motor, Kofferraum?)>
<!ELEMENT Bezeichnung (Typ, Marke?)>
<!ELEMENT Typ (#PCDATA)>
<!ELEMENT Marke (#PCDATA)>
<!ELEMENT Motor (PS | kW)>
<!ELEMENT PS (#PCDATA)>
<!ELEMENT kW (#PCDATA)>
<!ELEMENT Kofferraum (#PCDATA)>
<!ATTLIST Motor Treibstoff (Benzin | Diesel) #REQUIRED>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Auto SYSTEM "auto.dtd">
<Auto>
<Bezeichnung>
<Typ>Golf</Typ>
</Bezeichnung>
<Motor Treibstoff="Benzin"><PS>75</PS></Motor>
<Kofferraum>330</Kofferraum>
</Auto>
```

Correctness? (4)

```
<!ELEMENT Auto (Bezeichnung, Motor, Kofferraum?)>
<!ELEMENT Bezeichnung (Typ, Marke?)>
<!ELEMENT Typ (#PCDATA)>
<!ELEMENT Marke (#PCDATA)>
<!ELEMENT Motor (PS | kW)>
<!ELEMENT PS (#PCDATA)>
<!ELEMENT kW (#PCDATA)>
<!ELEMENT Kofferraum (#PCDATA)>
<!ATTLIST Motor Treibstoff (Benzin | Diesel) #REQUIRED>

<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE Auto SYSTEM "auto.dtd">
<Auto>
<Bezeichnung>
<Typ>Golf</Typ>
<Marke>
VW
</Bezeichnung>
</Marke>
<Motor Treibstoff="Benzin"><PS>75</PS></Motor>
</Auto>
```


Excercise: DTD?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE House SYSTEM "house.dtd">
<House>
<Address>
<Street>Park Avenue</Street>
<Number>23</Number>
</Address>
<Room>living room</Room>
<Room>bathroom</Room>
<Type>Apartment</Type>
</House>


<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE House SYSTEM "house.dtd">
<House squaremeters="107">
<Address>
<Street>Bakerstreet</Street>
<Number>138</Number>
</Address>
<Room>bedroom</Room>
</House>
```


Problems with DTDs

A DTD is not an XML-Document

- additional notation
- no processing with XML-Tools

DTDs have only a limited support for data types

- only strings

DTDs do not support namespaces

DTDs do not allow a straightforward specification of complex document structures

- Example: a particular element shall occur exactly 20 times

Solution: XML Schema

CDATA-Section

CDATA is skipped by the parser

Entities and tags within a CDATA-section are not recognized

]]> ends the CDATA-section

Often used for Code examples, e.g. XML-in-XML

Can also be used to integrate Multimedia data

```
...
<script>
   <![CDATA[
      if ( a < b ) {
         &subroutine(a,b)
      } else {
         &subroutine(b,a)
      }
   ]]>
</script>
...
```

Comments

Can be used within DTDs and in XML-Documents

Usage:

- Structuring of documents,
- Increase readability for the human user

Comments are allowed everywhere except before the prolog (<?xml ...)

Comments must not contain "—“

```
...
<!--
A Comment can contain
things like <tags> or
&entities;
-->
...
```