Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Module 2

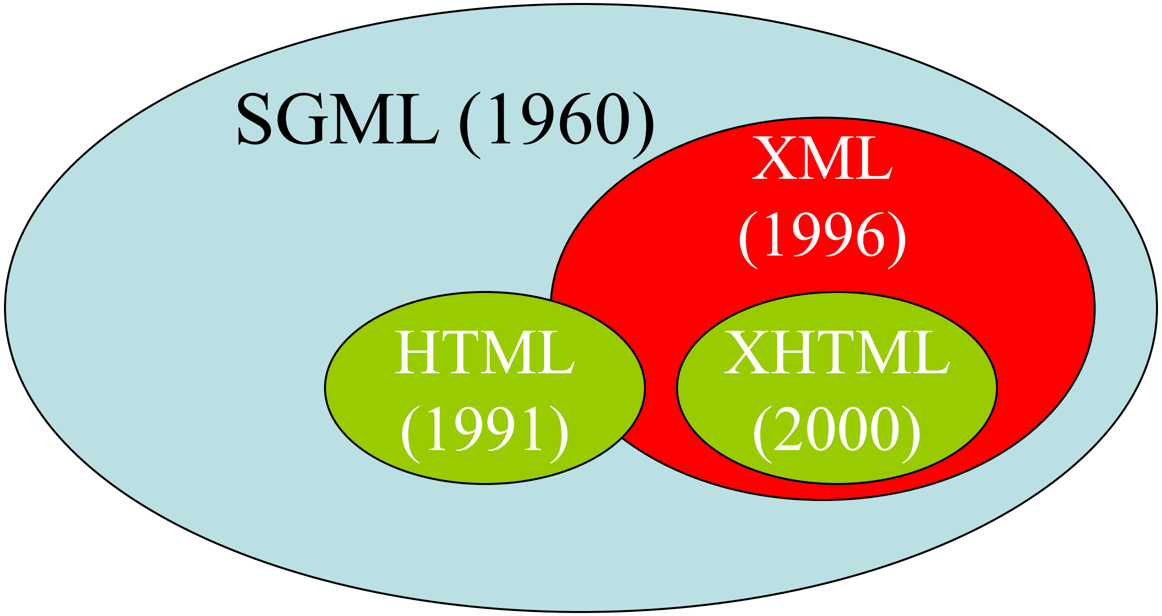
XML Basics

Part 1: XML, Namespaces, DTD

Agenda

- XML Core
- Namespaces
- DTD: Structural Constraints for XML
- Next week: XML Schema: Complex constraints, data types

SGML vs. HTML vs. XML



Why XML?

- HTML is to be interpreted by browsers
  - Shown on the screen to a human
- Desire to separate the “content” from “presentation”
  - Presentation has to please the human eye
  - Content can be interpreted by machines
  - Presentation is a handicap for machines
- Semantic markup of the data

Information about a book in HTML

```
<td><h1 class=”Books">Politics of experience by Ronald Laing,
    published in 1967</h1></td><td align="right" nowrap> Item
    number:320070381076</td><td align="right" valign="top"><img
    src="http://pics.booksstatic.com/aw/pics/globalAssets/rtCurve.gif"
    width="8" height="8"></td></tr><tr><td colspan="6" valign="middle"
    bgcolor="#5F66EE"><img
    src="http://pics.booksstatic.com/aw/pics/s.gif" width="1"
    height="4"></td></tr></table><table width="100%" border="0"
    cellpadding="0" cellspacing="0"><tr><td bgcolor="#CCCCFF"><img
    src="http://pics.booksstatic.com/aw/pics/s.gif" width="1"
    height="1"></td><td bgcolor="#EEEEFF"><div
    id="FastVIPBIBO"><table border="0" cellpadding="0"
    cellspacing="0" width="100%">
```

Same Data as XML

```
<book year="1967">
    <title>The politics of experience
    </title>
     <author>
            <firstname>Ronald</firstname>
            <lastname>Laing</lastname>
     </author>
</book>
```

- Information is (1) decoupled from presentation, then (2) chopped into smaller pieces, and then (3) marked with semantic meaning
- It can be processed by machines
- Like HTML, only syntax, not logical abstract data model

XML Components

- Elements
- Attributes
- Text
- Comments
- Processing Instructions
- Namespace Declarations
- Document = Prolog + Root Element (+ Text)
  + Comments + Processing Instructions
- All inherited from SGML, then HTML

Elements

- Enclosed in Tags
  - Begin Tag: e.g., <bibliography>
  - End Tag: e.g., </bibliography>
  - Element without content: e.g., <bibliography />
- Elements can be nested
  <bib> <book> Wilde Wutz </book> </bib>
- Subelements can implement multisets
  <bib> <book> ... </book> <book> ... </book> </bib>
- Documents must be well-formed
  <a> <b> </a> </b> is forbidden!
  <a> <b> </b> is forbidden!

Attributes

- Attributes are associated to Elements

```
<book  price = "55" currency = "USD" >
    <title> ... </title>
    <author> ... </author>
</book>
```

- Element that only has attributes

```
<person name = "Wutz" age = "33"/>
```

- What is the difference between a nested element and an attribute? Are attributes useful?
- Attribute names must be unique! (No Multisets)

  <person name = "Wilde" name = "Wutz"/> is illegal!

Text and Mixed Content

- Text appears in element content
  - <title>The politics of experience</title>
- Can be mixed with other subelements
  - <title>The politics of <em>experience</em></title>
- Mixed Content
  - For “document – oriented“ XML – very useful
  - The need does not arise in „data“ processing, only entities and relationships
  - People speak in sentences, not E/R. XML allows to preserve the structure of natural language, while adding semantic markup that can be interpreted by machines.

Spectrum: Natural language, semi-structured data, and structured data

1. Dana said that the book entitled „The politics of experience“ is really excellent !
2. <citation author=„Dana“> The book entitled „The politics of experience“ is really excellent ! </citation>
3. <citation author=„Dana“> The book entitled <title> The politics of experience</title> is really excellent ! </citation>
4. <citation>
   <author>Dana</author>
   <aboutTitle>The politics of experience</aboutTitle>
   <rating> excellent</rating>
   </citation>

Comments, PIs, Prolog

- Comment: Syntax as in HTML
  <!-- This is just a comment -->
- Processing Instructions
  - Contain no data - interpretation by processor
  - Syntax: <?pause 10 secs ?>
  - Pause is "Target"; 10secs is "Content"
  - xml is a reserved target for prolog (all case variants)
- Prolog
  <?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
  <?xml-stylesheet href="demo.css" type="text/css"?>
  - Standalone defines whether there is a DTD
  - Encoding is usually Unicode.

Whitespace

- Whitespace = Text of Space, Tabs and Returns
- Special Attribute xml:space to control use
- Human-readible XML (with Whitespace)

```
<book xml:space="preserve" >
    <title>Die wilde Wutz</title>
    <author>D.A.K.</author>
</book>
```

- (Efficient) machine-readible XML (no WS)

```
 <book xml:space="default" ><title>Die wilde
Wutz</title><author>D.A.K.</author></book>
```

- Performance improvement: ca. Factor 2.

CDATA sections

- Sometimes we would like to preserve the original characters, and not interpret them as markup
- CDATA sections
  - Not parsed as XML

```
<message>
   <greeting>Hello,world!</greeting>
</message>
<message> <![CDATA[<greeting>Hello, world!</greeting>]]>
</message>
```

Language declaration

Express that certain elements are in a formal/natural language

```
<p xml:lang="en">The quick brown fox jumps over the lazy dog.</p>
<p xml:lang="en-GB">What colour is it?</p>
<p xml:lang="en-US">What color is it?</p>
```

Universal Resource Identifiers

- URL (Universal Resource Locators): deferenceable identifier on the Web
  - The target of an URL pointer is an file (virtual or materialized)
- URIs (Unique Resource Identifier): general purpose key to resources on the Web
  - Uniquely identifies a resource
  - Target is not an HTML file, can be anything (schema, table, file, entity, object, tuple, person, physical item, etc)
  - Lifetime and scope of this “key” is user dependent
- IRI (Internationalized Resource Identifiers)
  - Allow non Latin characters (Chinese, Arabic, Japanese, etc)
- URL, URI, IRIs
  - All strings
  - Very LONG strings

Namespaces

- Integration of Data from diverse data sources
- Integration of different XML Vocabularies (aka Namespaces)
- Each „vocabulary“ has a unique key, identified by a URI/IRI
- Same local name, from different vocabularies can have
  - Different meaning
  - Different structure associated with it

QNames

- Qualified Names (QName) to attach a „name“ to its „vocabulary“
  - for all nodes in an XML document that has names (Attributes, Elements, Pis
- QName ::=   triple (URI [prefix:] localname)
  - Binding (prefix, URI) is introduced in elements start tag
  - Later only the prefix is used, not the long URIs
  - Prefix is optional, default namespaces
  - Prefix and localname a separated by „:“

Namespace Definitions

- Namespace definitions look like Attributes
  - Identified by „xmlns:prefix“ or „xmlns“ (default namespace)
  - Bind the Prefix to the URI
- Scope is the entire element in which it is declared
  - Includes the element itself, its attributes and its subtrees
  - Scopes can be nested
  - Redeclaration possible (same prefix to different URI, different prefixes to same URI)
  - „Undeclare“ only with newer versions of XML+Namespace
- Example

```
<ns:a xmlns:ns=„someURI“ ns:b=„foo“>
        <ns:b>content</ns:b>
</ns:a>
```

Example: Namespaces

- DQ1 defines dish for tableware
  - Diameter, Volume, Decor, ...
- DQ2 defines dish for satellite
  - Diameter, Frequency
- How many "dishes" are there?
- Better ask for:
  - "How many dishes are there?"

    or
  - "How many dishes are there?"

Example: Namespaces

```
<gs:dish  xmlns:gs = "http://tableware.com" >
    <gs:dm gs:unit = "cm">20</gs:dm>
    <gs:vol gs:unit = "l">5</gs:vol>
    <gs:decor>Gelsenkirchener Barock</gs:decor>
</gs:dish>
<sat:dish xmlns:sat = "http://television.com" >
    <sat:dm>200</sat:dm>
    <sat:freq>20-2000MHz</sat:freq>
</sat:dish>
```

Mixing Several Namespaces

```
<gs:dish  xmlns:gs = "http://tableware.com"
                  xmlns:uom = "http://units.com">
  <gs:dm uom:unit = "cm">20</gs:dm>
  <gs:decor>Gelsenkirchener Barock</gs:decor>
  <gs:vol  uom:unit = "l">5</gs:vol>
  <comment>I am unqualified</comment>
</gs:dish>
```

Default Namespaces

tableware.com if not specified otherwise.

```
<dish  xmlns = "http:// tableware.com"
                xmlns:uom = "http://units.com">
  <dm uom:unit = "cm">20</dm>
  <decor>Gelsenkirchener Barock</decor>
  <vol  uom:unit = "l">5</vol>
  <comment>I am qualified</comment>
</dish>
```

Namespace Notes

- URIs used to identify a namespace uniquely
  - But nobody interprets them
  - www.dangling.com is okay
  - Aliases irrelevant
- Default Namespace does not apply to attributes

```
<e  xmlns = "http://n.com">
   <e attn = "unq"/>
<e>
```

- Scope is complete Element

```
<n:e  xmlns n="http://n.com"  n:attl = "legal " >
   <n:e n:attn = "legal"/>
</n:e>
```

(Older versions of namespace definition consider n:attl illegal!)

Documents Format Descriptions

- XML is a meta-format
- Easy to start with, use your own tags
  - Contrast to relational DB, OO languages
- Only restriction: XML needs to be well-formed
- At some point, this is too much freedom
  - Use same syntax for different documents
  - Facilitate the writing of applications that process data
  - Exchange data with other parties

➢ Need to restrict the amount of freedom

➢ Document Description Methods

Correctness of XML documents

- Well formed documents
  - Verify the basic XML constraints, e.g. <a></b>
- Valid documents
  - Verify the additional constraints (structure, values)
- Non well formed XML documents cannot be processed
- Non-valid documents can still be processed (queried, transformed, etc)

Overview of XML Schema Languages

- Several standard Schema Languages
  - DTDs, XML Schema, RelaxNG, Schematron
- Schema languages have been designed after, and in an orthogonal fashion, to XML itself
- Schemas and data are decoupled in XML
  - Data can exist with or without schemas
  - Or with multiple schemas
  - Schema evolutions rarely impose evolving the data
  - Schemas can be designed before the data, or extracted from the data (DataGuide -- Stanford)
- Makes XML the right choice for manipulating semi-structured data, or rapidly evolving data, or highly customizable data

Document Type Definition (DTD)

- Inherited from SGML
- Part of the original XML 1.0 specification
- Defines Structure of Documents
  - Nesting of Elements, possible and mandatory
  - Attributes of Elements
  - Possible Values of Attributes
- Four kinds of Declarations
  - Notation, Entity, **Element Type**, **Attribute List**
- Checking the structural constraints: DTD validation (valid vs. invalid documents)
- DTD very useful for a while, not used anymore, several major limitations

DTD Example

```
<!ELEMENT book (title, (author+ | editor), publisher?)>
<!ATTLIST book
    year CDATA #REQUIRED
    isbn    ID            #REQUIRED
    price   CDATA   #IMPLIED
    curr    CDATA   #FIXED "EUR"
    index  IDREFS    "" >
<!ELEMENT author (firstname, lastname)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT lastname (#PCDATA)>
<!ELEMENT title (#PCDATA)>
```

Element Type Declaration

- Structure: <!ELEMENT *name content*>
- Beispiel

```
<!ELEMENT book (title, (author+ | editor), publisher?)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author EMPTY>
<!ELEMENT publisher ANY>
```

- Valid document according to this DTD

```
<book >
    <title>Die wilde Wutz</title>
    <author/>  <author></author>
 <publisher><anything>...</anything></publisher>
</book>
```

Element Type Declarations

- Element Types are composed of:
  - Subelements (identified by Name)
  - Attribute lists (identified by Name)
  - Selection of Subelemente (choice)
  - PCDATA
- Quantifier for Subelements and Choice
  - "+" for at least 1
  - "*" for 0 or more
  - "?" for 0 or 1
  - Default: exactly 1
- EMPTY and ANY are special predefined Types

Attribute Lists

- Structure: <!ATTLIST *ElementName definition*>
- <!ATTLIST book

```
          isbn    ID            #REQUIRED
          price   CDATA   #IMPLIED
          curr    CDATA   #FIXED "EUR"
          index  IDREFS    "" >
```

- Valid and Not-valid Books

```
<book isbn="abc" curr="EUR"/>  !! no price
<book isbn="abc" price="30"/>  !! Curr, index default
<book index="DE" isbn="abc" curr="EUR"/>
<book/>   !! Missing isbn Attribute
<book isbn="abc" curr="USD"/>  !! wrong currency
```

Attribute Types

- CDATA: normal text
- ID
  - Value is unique within document
  - Element has at most one attribute of this type
  - No default values allowed
- IDREF, IDREFS
  - References to other elements within the document
  - IDREFS: Enumeration, " " as separator
- ENTITY, ENTITIES, NOTATION
  - See Entity and Notation Declarations in DTD

Attribute Defaults

- #REQUIRED
  - Document must specify a value for attribute
- #IMPLIED
  - Attribute is optional, there is no default
- *value*
  - Default value, if no other value specified
- #FIXED *value*
  - Default value, if no other value specified
  - If value specified, it must be the fixed value

Declarations of DTDs in XML document

- None (well-formed Documents)
- In Document:
  <!DOCTYPE name *[definitions]* >
- External, specified by URI:
  <!DOCTYPE name SYSTEM "demo.dtd">
- External, Name and optional URI:
  <!DOCTYPE name PUBLIC "Demo">
  <!DOCTYPE name PUBLIC "Demo" "demo.dtd">
- In Document + External:
  <!DOCTYPE name SYSTEM "demo.dtd"
  [local definitions] >
- *name* = root Element

Limitations of DTDs

- DTDs describe only the “grammar” of the XML file, not the detailed structure and/or types
- This grammatical description has some obvious shortcomings:
  - we cannot express that a “length” element must contain a non-negative number *(constraints on the type of the value of an element or attribute)*
  - **The “unit”** element should only be allowed when **“amount”** is present *(co-occurrence constraints)*
  - the “**comment”** element should be allowed to appear anywhere *(schema flexibility)*

Summary

- XML as inheriting from the Web history
  - SGML, HTML, XHTML, XML
- XML key concepts
  - Documents, elements, attributes, text
  - Order, nested structure, textual information
- Namespaces
- DTDs and the need for describing the “structure” of an XML file
- Next: XML Schemas