Markup, XML and namespaces

BIBLIOTEKSHÖGSKOLAN
HÖGSKOLAN I BORÅS

Mikael Gunnarsson

October, 2018

BHS/GU — Borås/Göteborg

HÖGSKOLAN I BORÅS

Context

This presentation is targeting several groups of students at both Gothenburg University and University College of Borås.

Its aim is to give a theoretical introduction, with a somewhat humanities perspective, on a quite technical notion — $\mathcal{XML}$.

HÖGSKOLAN I BORÅS

Contents

What characterizes XML

(Almost) everything being XML

- realizes some kind of **markup**
- implies a certain generalized **data structure**
- implies a certain **notation**
- realizes a certain **application** of abstractions in terms of one or more applied data structures, where the structural elements are to be used in a certain predetermined way – in many cases published as a so called **namespace**

Some applications/namespaces are realized as XML, but can also be realized in other formalisms/notations. E. g. HTML, RDF, MARC21 ...

Some applications are only XML-based, such as XSL, and are not applicable in other formalisms.

HÖGSKOLAN I BORÅS

Markup - the classical typology[1]

- Interpunctional
- Presentational
- Procedural
- Descriptive

  \+ Referential markup; Meta markup

[1] Coombs, J. H., Renear, A. H., DeRose, S. J. (1987). Markup Systems and the Future of Scholarly Text Processing. *Communications of the Association for Computing Machinery*, 30(11), 933–947.

Scriptio continua

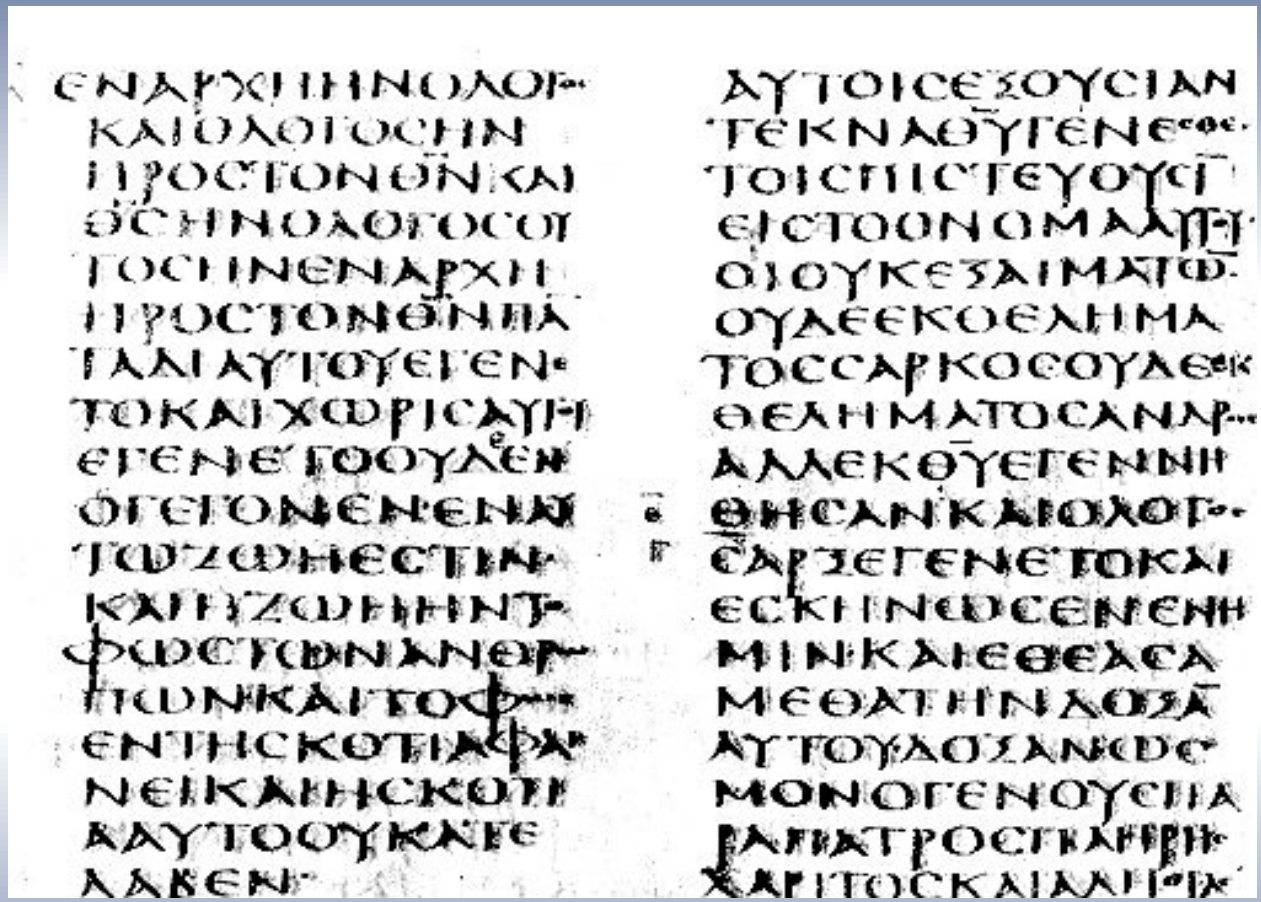


Figure: Scriptio continua from Codex Sinaiticus, 300 AD

HÖGSKOLAN I BORÅS

Interpunctuation and white space

Not only words differ and changes according to cultural adherence and the development of spoken and written languages.

“My name is Steve”	$\Longleftarrow$	«Je m’appelle Steve»
What’s your name?	$\Longleftarrow$	¿cómo te llamas?

HÖGSKOLAN I BORÅS

Interpunctuation and white space

Not only words differ and changes according to cultural adherence and the development of spoken and written languages.

"My name is Steve"	$\Longleftarrow$	«Je m'appelle Steve»
What's your name?	$\Longleftarrow$	¿cómo te llamas?
`Kua kite te taŋata i te maoriŋoo` (Maori)	$\Longleftarrow$	The man saw the shark
`nitakupenda` (Swahili)	$\Longleftarrow$	I-will-you-love
En sjuk syster (Swedish)	$\neq$	En sjuksyster
A web site	$\equiv$	A website

Early signs of interpunctional use

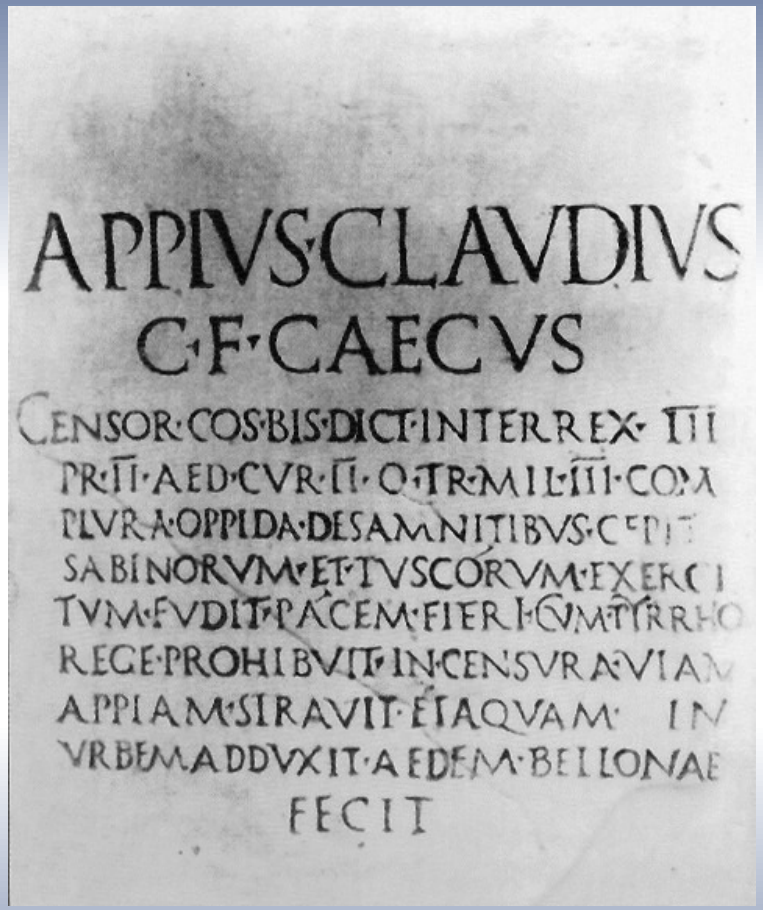


Figure: Roman inscription. Note the dots and the apex for marking a long vowel

Some words of advice

The following three glyphs are usually considered allographs (i.e. the same grapheme),

$\mathcal{A}$ $\mathbb{A}$ A

It doesn't change the meaning of the text if we substitute one for the other. It doesn't have to do with content, but form. Neither do the substitution of ” for «. Form should be separated from content.

Single white space characters (unvisible glyphs, so to say) used for the separation of words is another matter. They are (obviously) best considered part of the text semantics.

Other interpunctional characters (glyphs) found in a manuscript for digitisation may always have to be reconsidered, and a decision may be made to use XML-markup for them in stead.

`inform<lb rend='hyphen'/>ation` rather than `inform-<lb/>ation`

Presentational markup

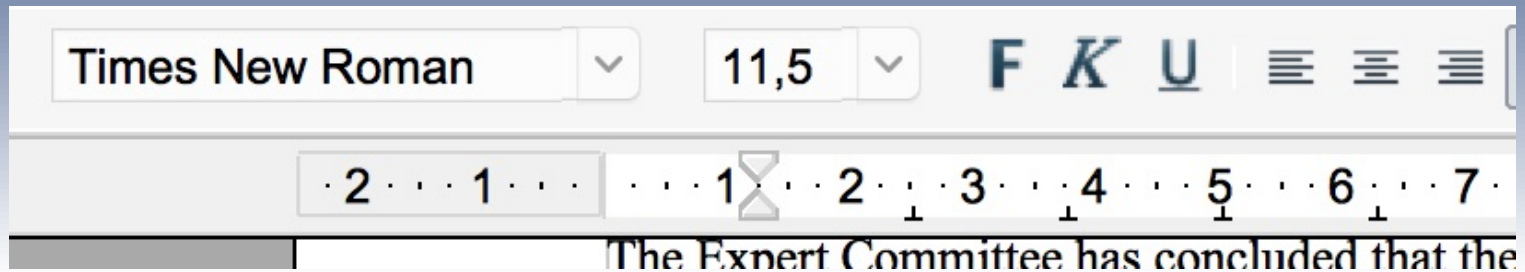


Figure: We often use the visually oriented facilities in MS Word: adjusting font size, font family, margins etc at each and every point in the text

The sometimes overwhelming disadvantage with this strategy is that each and every point needs to be explicitly manipulated if the appearance of a category of elements needs to be revised . . . not to mention that we cannot be sure why something is given in e.g. italics. Is it emphasis, a name, a title, a place, a quote, . . . ?

HÖGSKOLAN I BORÅS

Procedural markup

There is no sharp distinction between presentational and procedural, but this example may serve as an illustration:

```
\begin{center} Markup, XML and namespaces \\
Workshop for Digital Humaniora
\end{center}
```

Note how the instruction `center` is directed to some typesetting processor, in order to center anything between the two "tags", and that \\is for inserting a line break.

HÖGSKOLAN I BORÅS

Descriptive markup[2]

```
\section{Markup, XML and
namespaces}
\label{sec:greetings}
\par Lorem ipsum dolor sit amet,
consectetuer ...
    \par Nam dui ligula, fringilla a,
        euismod sodales ...
```

Figure: LaTeX markup

```
<h1>Markup, and namespaces</h1>
<p>
Lorem ipsum dolor sit amet,
consectetuer ...
</p> <p> Nam dui ligula, fringilla a,
    euismod sodales ... </p>
```

Figure: HTML markup

[2] The distinction between descriptive and procedural has in fact been questioned, and a distinction between performative and declarative introduced, borrowed from speech-act theory.

Descriptive markup, contin.

- "Pure" descriptive markup:
  The creator describes titles, chapters, paragraphs, title pages, **line and page breaks** etc detected in an existing physical edition (e.g.: TEI)

Descriptive markup, contin.

- "Pure" descriptive markup:
  The creator describes titles, chapters, paragraphs, title pages, **line and page breaks** etc detected in an existing physical edition (e.g.: TEI)
- Performative markup:
  The creator declares title, sections, paragraphs, title pages, quotes, equations etc in a text coming-into-being (e.g.: HTML)

In the first case, it is meaningful to identify correctness and errors in the encoding, in the second case it is not.

Summary

- Markup serves the purpose of structuring a set of data by means of encoding its structure as to what type each component is an instance of, in other words, conveying the semantics of its structure

Summary

- Markup serves the purpose of structuring a set of data by means of encoding its structure as to what type each component is an instance of, in other words, conveying the semantics of its structure
- . . . not only text needs structural declarations . . .

Components of an xml instance

Elements

Elements carry all the contents of an instance (a document) . . .

```
<title>Introduction to XML  </title>
<image URI="logo.jpg"/>
     /.../
```

Components of an xml instance

Elements

Elements carry all the contents of an instance (a document) . . .

```
<title>Introduction to XML  </title>
<image URI="logo.jpg"/>
     /.../
```

Elements may contain other elements and text data and there is always a root node that contains one and only one root element and all other stuff therein. In HTML this is the element html.

Components of an xml instance

Attributes

... together with attributes and attribute values ...

```
<title type="mainTitle">Introduktion till XML</title>
<image URI="logo.jpg"/>
```

Components of an xml instance

Attributes

... together with attributes and attribute values ...

```
<title type="mainTitle">Introduktion till XML</title>
<image URI="logo.jpg"/>
```

Terminology:

<title>, </title> and <image/> are termed **tags** or (in Swedish) **märken**. Tags and the content between the same tags form an **element**.

Components of an xml instance

Entiteter

... and possible entity references.

```
&amp;   &gt;    &lt;
```

Components of an xml instance

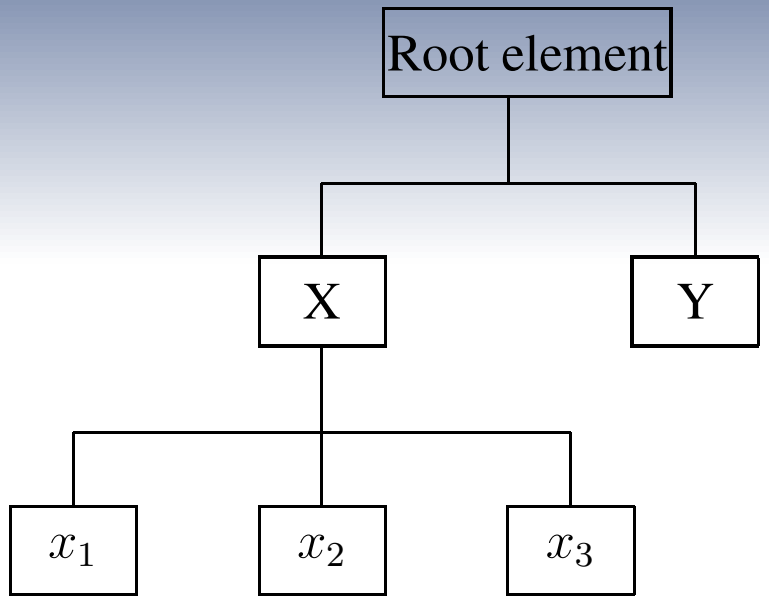
Entiteter

... and possible entity references.

```
&amp;   &gt;    &lt;
```

&, < och > have special meaning and must be given as entitety references. (This is called *escaping* in other contexts, such as in programming.)

The structure of an xml instance



1) x_1, x_2, x_3 is a (*child*) of X

2) X is a (*parent*) of x_1, x_2, x_3

3. x_1, x_2, x_3 are (*siblings*)

4) x_1 is a (*preceding-sibling*) of x_2

5) The root element is (*ancestor*) of x_1, x_2, x_3

6) x_1, x_2, x_3 are (*descendants*) of the root element and of X

Example - shopping list

```
<?xml version="1.0" encoding="utf-8"?>
<groceryList>
 <product><quantity>1 l</quantity>
          <name>Milk</name></product>
 <product><name>Coffee</name></product>
 <product><name>Bread</name></product>
 <product><quantity>3 dl</quantity>
          <name>Cream</name></product>
 <product><quantity>1 package</quantity>
          <name>Yeast</name></product>
 <product><quantity>2 kg</quantity>
          <name>Flour</name></product>
</groceryList>
```

Natural vs artificial languages

- A **language** may be defined as a finite set meaningful **words** (a vocabulary) allowed to be combined according to certain predetermined **rules** (syntactic rules) in order to form meaningful phrases and sentences that can be used for meaningful statements or to accomplish an effect.

Natural vs artificial languages

- A **language** may be defined as a finite set meaningful **words** (a vocabulary) allowed to be combined according to certain predetermined **rules** (syntactic rules) in order to form meaningful phrases and sentences that can be used for meaningful statements or to accomplish an effect.
- We discern between **natural languages** and **artificial languages**. Natural languages can be **described** in a lexico-grammar, while artificial languages usually are **defined** by means of some meta language. (programming languages are artificial languages)

Natural vs artificial languages

- A **language** may be defined as a finite set meaningful **words** (a vocabulary) allowed to be combined according to certain predetermined **rules** (syntactic rules) in order to form meaningful phrases and sentences that can be used for meaningful statements or to accomplish an effect.
- We discern between **natural languages** and **artificial languages**. Natural languages can be **described** in a lexico-grammar, while artificial languages usually are **defined** by means of some meta language. (programming languages are artificial languages)
- HTML is an artificial language for publishing text (natural language) on the web, originally defined by means of SGML (ca 1989). TEI is an artificial language for the description of e.g. cultural heritage material that resides in tangible form. It was conceptualized in SGML form, but is now in XML form. Both HTML and TEI requires that each instance complies with the rules set up in their grammars.

Meta languages

- XML is a **meta languages** in the sense that it is used to define other languages – often termed **XML applications**. XML is **not** a programming language.

Meta languages

- XML is a **meta languages** in the sense that it is used to define other languages – often termed **XML applications**. XML is **not** a programming language.
- An XML application is defined while its elements and attributes are defined, and how they are allowed to be used. Hence, we'll get a certain data structure, or class of objects, defined.

Meta languages

- XML is a **meta languages** in the sense that it is used to define other languages – often termed **XML applications**. XML is **not** a programming language.
- An XML application is defined while its elements and attributes are defined, and how they are allowed to be used. Hence, we'll get a certain data structure, or class of objects, defined.
- An XML application can be defined by means of several formalisms. Thereby we get **rules** that are **compelling**.

Formalisms for XML

- Document Type Definition (DTD)
- XML Schema
- Relax NG (https://en.wikipedia.org/wiki/RELAX_NG)
- Schematron

Grammars

- Grammars (in XML sense) are intended as prescriptive rules for how an instance of an XML document (type) is allowed to be structured — how its components form a tree structure with a suitable predictability. (Cf. the relatively predictable structure of a research article in a scientific journal)

Grammars

- Grammars (in XML sense) are intended as prescriptive rules for how an instance of an XML document (type) is allowed to be structured — how its components form a tree structure with a suitable predictability. (Cf. the relatively predictable structure of a research article in a scientific journal)
- Some document types are more predictable than others. (Think about personal home pages some 10 years ago and how the increased dominance of CMSes like Wordpress has affected the structure of blogs and homepages of today. Compare to the yearly tax reports you need to hand in every year.)

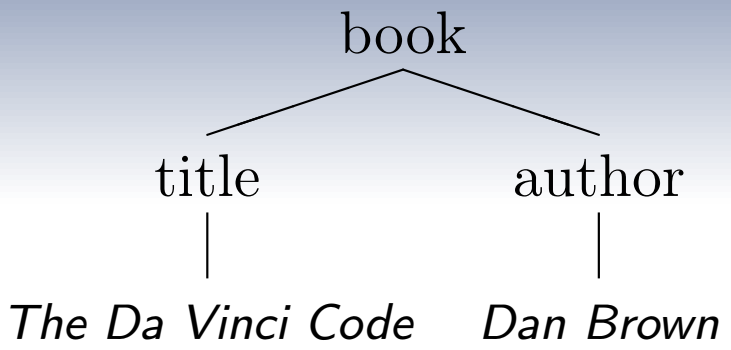
Grammars

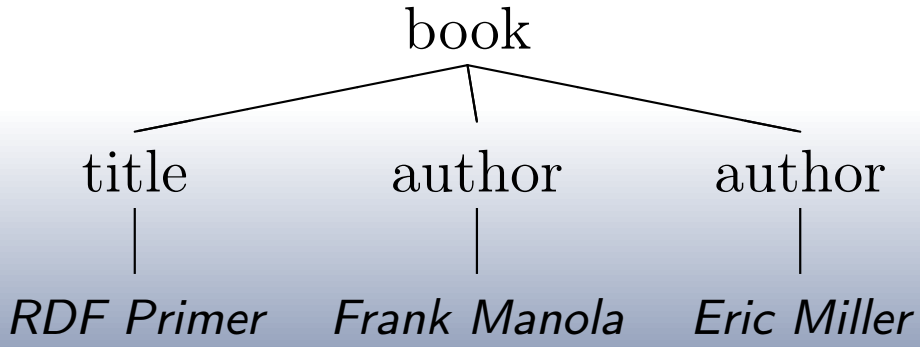
- Grammars (in XML sense) are intended as prescriptive rules for how an instance of an XML document (type) is allowed to be structured — how its components form a tree structure with a suitable predictability. (Cf. the relatively predictable structure of a research article in a scientific journal)
- Some document types are more predictable than others. (Think about personal home pages some 10 years ago and how the increased dominance of CMSes like Wordpress has affected the structure of blogs and homepages of today. Compare to the yearly tax reports you need to hand in every year.)
- Any meaningful data structure can be generalized as a document grammar.
- The creator (of a grammar) needs to decide upon e.g.
  - which elements to be allowed,
  - by what names they should be referenced,
  - and in what order they may occur

HÖGSKOLAN I BORÅS

The DTD formalism

```
<!ELEMENT book (title,author+)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
```





The XML Schema formalism

The root element: the skeleton

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDef
    ... A set of definitions ...
</xs:schema>
```

Some content models

Simple elements with text content

```
<xs:element name="title" type="xs:string"/>
<xs:element name="creator" type="xs:string"/>
```

```
<creator>August Strindberg</creator>

<title>Röda rummet</title>
```

Some content models

ComplexType element with element content

```
<xs:element name="book">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="creator"/>
            <xs:element ref="title"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

```
<book>
<creator>August Strindberg</creator>
<title>Röda rummet</title>
</book>
```

HÖGSKOLAN I BORÅS

Implications

- The order between creator and title is mandatory
- Exactly one instance of each element is required

Some content models

ComplexType element with optional element content

```
<xs:element name="book">
    <xs:complexType>
        <xs:choice>
            <xs:element ref="creator"/>
            <xs:element ref="title"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

Only one of the enumerated elements is allowed, but the order does not matter!

Some content models

ComplexType element with optional element content, contin.

```
<xs:element name="book">
    <xs:complexType>
        <xs:choice maxOccurs="unbounded">
            <xs:element ref="creator"/>
            <xs:element ref="title"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

Only one of the enumerated elements is allowed, but the whole choice-pattern can be repeated infinitely!

Mixed content

ComplexType element interspersed with element content

```
<xs:element name="creator">
    <xs:complexType mixed="true">
        <xs:choice minOccurs="0">
                <xs:element ref="familyName"/>
        </xs:choice>
    </xs:complexType>
</xs:element>
```

```
<book>
<creator>August <familyName>Strindberg</familyName></creator>
<title>Röda rummet</title>
</book>
```

HÖGSKOLAN I BORÅS

Mixed content

Mixed content with attribute

```
<xs:element name="creator">
    <xs:complexType mixed="true">
        <xs:choice minOccurs="0">
                <xs:element ref="familyName"/>
        </xs:choice>
        <xs:attribute name="id" use="required" type="xs:ID"/>
    </xs:complexType>
</xs:element>
```

```
<book>
<creator>August <familyName id="b1">Strindberg</familyName></creator>
<title>Röda rummet</title>
</book>
```

Namespaces

- If several languages are to be used simultaneously, there need to be unambiguous indications on what language is used (referenced) at a certain point.
- A namespace is simply a particular application of XML with a name (rdf and dc in the example below) and identifier (URIs below) referenced by a **namespace prefix**.

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
        /.../
<rdf:Description
    rdf:about="http://libris.kb.se/resource/bib/3095664">
    <dc:creator
       rdf:resource="http://libris.kb.se/resource/auth/94541"/>
</rdf:Description>
      /.../
```

HÖGSKOLAN I BORÅS

XML is not the end

- Apart from XHTML every application of XML for static data representation are best seen as either meant for archive purposes or for data exchange (not for direct user access). Mostly, XML sources are transformed for presentation och other kinds of processing.
- **XSL** (**Extensible Stylesheet Language**) — XPath, XSL Transformations, XSL Formatting Objects — being itself an XML application, provide ways of processing XML instances.
- XQuery provide ways of searching for data in XML repositories
- ...