Chapter 3:
XML Namespaces

References:

- Tim Bray, Dave Hollander, Andrew Layman: Namespaces in XML. W3C Recommendation, World Wide Web Consortium, Jan 14, 1999. [http://www.w3.org/TR/1999/REC-xml-names-19990114], [http://www.w3.org/TR/REC-xml-names].
- Tim Bray, Jean Paoli, C.M. Sperberg-McQueen: Extensible Markup Language (XML) 1.0, 1998. [http://www.w3.org/TR/REC-xml] See also: [http://www.w3.org/XML].
- Elliotte R. Harold, W. Scott Means: XML in a Nutshell, 3rd Ed. O'Reilly, 2004, ISBN 0596007647.
- Jonathan Borden, Tim Bray: Resource Directory Description Language (RDDL). Feb. 18, 2002. [http://www.rddl.org/]

Objectives

After completing this chapter, you should be able to:

- explain why namespaces are needed.
- determine the namespace of any element or attribute in a given XML document.
- use namespaces in your own XML documents.

Overview

Motivation (1)

- XML is a meta language that permits to define markup languages for specific applications.
- Many different sets of element types/tags (DTDs) for many different applications have been proposed.
- This all works well as long as each document adheres to a single, known document type.
- However, there are cases where documents contain a mixture of tags from different DTDs.

  Programs that have to process such documents know only the subset of the occurring element types that is relevant for the program.

Motivation (2)

- For instance, an XSLT stylesheet that translates from a given DTD into HTML contains tags from this DTD, from XSLT, and from HTML.
- It is possible to develop DTDs in a modular way by composing different sets of tags.

  It is not necessary to redevelop everything from scratch. Instead, one should try to reuse existing DTDs.
- It is possible that an XML document is processed by different programs, where each considers only its own tags.

Motivation (3)

- When a document contains element types from independently developed markup languages, there can be name clashes: The same name is defined in different vocabularies with different meaning.
- In such cases it is not clear what program should process this tag.

  A program that processes documents not restricted to a fixed, known DTD must be able to determine which tags are intended for this program.

Overview

Globally Unique Names (1)

- Thus, globally unique names for element types are sometimes necessary.
- However, a new committee that ensures the uniqueness of names would be expensive and slow.

  Furthermore, it is not clear that names should be assigned in a "first come, first served" fashion: Then DTDs that might later be widely used would have to select long, unintuitive names because some obscure DTD already registered the obvious name.
- Therefore, element types are uniquely identified via a URI/URL (the "namespace"), and a local name.

Globally Unique Names (2)

- It is obviously not practical to require that each time when an element type is used, the complete URI/URL has to be written in the tag.

  This would significantly increase the document size and decrease the readablility. Furthermore, e.g. the "/" is not allowed in XML names, so one could not simply make a URL a prefix of an element name.

- The solution is to define an abbreviation for the URI/URL within the document, and then to use this abbreviation as a prefix for the element name.

Globally Unique Names (3)

- For instance, one can declare "`xsl`" as an abbreviation for the namespace

  `http://www.w3.org/TR/WD-xsl`

- Then one can use the element name "`xsl:template`".
- The prefix "`xsl`" is arbitrary: If one declares instead "`abc`" as an abbreviation for the above namespace, the XSLT processor recognizes "`abc:template`".

  This is important, because the abbreviations are not enforced to be globally unique: It is possible that in the future, another markup language is defined, for which "`xsl`" is a natural abbreviation. Of course, within a single document, the abbreviations must be unique.

Globally Unique Names (4)

- Vice versa, the prefix "`xsl`" does not help, if it is defined as an abbreviation for a different URI.

  The XML parser makes this URI and the local name (the part after the colon ":") available to the applications. The application program (e.g. the XSLT processor) recognizes its tags by the URI.

- The namespace URI must only be unique. It is not required that anything specific is stored under this web address.

  Of course, the domain owner should be stable, so that not later somebody else wants to use this URI for a different namespace. It is proposed to store an RDDL description under the URI, but the namespace would also work if nothing is stored under the URI.

Equality of Namespaces

- Two namespaces are equal if their URIs are exactly identical (character by character, including uppercase/lowercase).
- For instance, two URIs that differ only in the case of the domain name are normally equivalent, but they denote different namespaces.
- Thus, the exact case is important when defining a namespace.

  Otherwise, the application, e.g. the XSLT processor, will not recognize "its" tags.

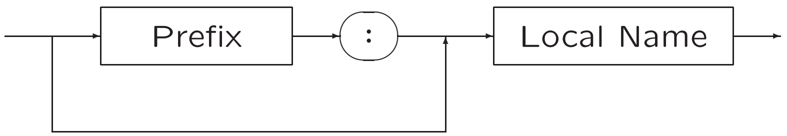
XML and Namespaces

- Namespaces are not defined in the XML specification itself, but in a separate short specification (11 pages).
- The XML specification permits the colon ":" in names without restrictions or a specific meaning.
- However, it already mentions that the colon should be used only according to the XML namespace recommendation.

  The XML Namespace recommendation was still work in progress when the first edition of the XML recommendation was published.

Syntax of Names (1)

Qualified Name:



- Qualified names can be used as element type names and as attribute names in the XML data.

  They cannot be used in the DTD (there is no possibility to declare a namespace that is applicable to the DTD). Entity names, goals of processing instructions and names of notations are not permitted to contain a colon ":".

Syntax of Names (2)

- Prefix and Local Name are sequences of letters, digits, underscore "_", hyphen "-", period ".".

  Plus certain extended characters from the unicode character set. They must start with a letter or an underscore "_".
- The prefix must be declared as an abbreviation for a namespace (see below).
- Programs that check only the well-formedness of XML data do not need to know about namespaces.

  Since XML names can contain ":", it is still legal XML. However, XML validors must be "namespace-enabled" to match the element type name from the DTD (without namespace) and the element type name in the data.

Overview

1. Motivation
2. Method: Globally Unique Names
3. Declaration of Namespaces

Namespace Declaration (1)

- A namespace can be defined in a start tag by means of an attribute with prefix "`xmlns`", e.g.

```
<example xmlns:xsl="http://www.w3.org/TR/WD-xsl"
         example_attr="...">
    ...
</example>
```

All names starting with "`xml`" are reserved for the XML-standards. Therefore the new special meaning of "`xmlns`" is no problem.

- Then the abbreviation (prefix) "`xsl`" for the namespace "`http://www.w3.org/TR/WD-xsl`" is defined for the entire element including all its contents.

Namespace Declaration (2)

- Since namespace declarations are inherited to all nested subelements, one would typically define the needed namespaces in the root element.

One can redefine a namespace prefix to stand for a different namespace in a subelement. Then the subtree rooted in this subelement is a hole in the scope of the namespace declaration above. However, one should avoid such tricks.

- The namespace declaration in a tag is already valid for the element name in this tag.

This is not obvious because the namespace declaration comes syntactically after its use. However, without this rule, there would be no possibility to define a namespace for the root element.

Namespace Declaration (3)

- Of course, one can define several namespaces in a single tag:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/TR/WD-xsl"
    xmlns:html="http://www.w3.org/TR/REC-html40"
    version="1.0">
```

- This defines the prefixes/abbreviations "`xsl`" and "`html`" for the two namespaces.
- Note that here "`xsl`" is already used in the element itself.

Namespace Declaration (4)

- It is possible to define two different prefixes for the same namespace (should be avoided — confusing):

```
<example
    xmlns:x="http://www.mynamespace.de"
    xmlns:y="http://www.mynamespace.de">
        <a:same_element_type/>
        <b:same_element_type/>
        <error x:a="1" y:a="2"/>
```

- One cannot give two values for the same attribute in an element, and `x:a` and `y:a` are the same attribute, since `x` and `y` stand for the same namespace.

Default Namespace

- One can also specify a default namespace by defining the attribute "`xmlns`" (without suffix):

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/TR/WD-xsl"
    xmlns="http://www.w3.org/TR/REC-html40"
    version="1.0">
```

- Now all element types without namespace prefix are automatically assigned to the HTML namespace.

  Like any normal namespace declaration, the default namespace declaration is inherited to all subelements, unless it is explicitly redefined. If one wants to define elements in a subtree to have no namespace, one can define `xmlns` as the empty string.

Namespaces of Attributes (1)

- If an attribute is specified in a tag without namespace prefix, it belongs to the element type of the tag ("local attribute").
- Default namespace declarations are not applied for attributes. E.g., consider again:

```
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/TR/WD-xsl"
    xmlns="http://www.w3.org/TR/REC-html40"
    version="1.0">
```

- Here, `version` is an attribute of `xsl:stylesheet`. It is not assigned to the HTML namespace.

Namespaces of Attributes (2)

- Different element types can have attributes with the same name and totally different meaning.
- However, attributes with namespace prefix are called "global attributes".
- They should always have the same meaning, even if they are specified in elements of different types. Examples are "`xml:lang`" and "`xml:space`".

  These remarks about local and global attributes are contained in a non-normative appendix of the namespace recommendation. It is only important that it is not necessary and not right to use a namespace prefix for normal attributes.