XML Namespaces

What does it EXACTLY mean?

Why Do We Need Namespaces?

1. To distinguish between elements and attributes
   from different vocabularies
   with different meanings.
2. To group
   all related elements and attributes together
   so that a parser can easily recognize them.

The Need for Namespaces (I)

- The XLink specification defines an attribute with the name *href*.
  The XHTML specification uses *href* attributes on some elements.
  The XInclude specification uses *href* attributes.
- An XSLT style sheet that will transform XHTML documents containing both Scalable Vector Graphics (SVG) pictures and MathML equations into XSL-Formatting object documents.
- The *a, title, script, style* and *font* elements in XHTML and SVG

The Need for Namespaces (II)

- The table element in XHTML and XSL-FO
- The text element in XSLT and SVG
- The set element in MathML and SVG
- An XSLT stylesheet that transforms a style sheet in an older version of the XSLT specification to a style sheet in a newer version of the XSLT specification.

Namespaces Disambiguate Elements

- Namespaces disambiguate elements
  with the same name
  from each other
  by attaching different prefixes
  to names from different XML applications.

- Each prefix is associated with a URI.
  - Names whose prefixes are associated
    with the same URI are in the same namespace.
  - Names whose prefixes are associated
    with different URIs are in different namespaces.

URIs

- **U**niform **R**esource **I**dentifier
- Two kinds:
  - URLs: Uniform Resource *Locators*
  - URNs: Uniform Resource *Names*
- Which should be used for namespaces?

Namespace Syntax

- Elements and attributes that are in namespaces have names that contain exactly one colon. They look like this:

```
rdf:description
xlink:type
xsl:template
```

- Everything before the colon is called the *prefix*
- Everything after the colon is called the *local part*.
- The complete name including the colon is called the *qualified name*.

Binding Prefixes to Namespace URIs

- Prefixes are bound to namespace URIs by attaching an xmlns:*prefix* attribute to the prefixed element or one of its ancestors.

```
<svg:svg xmlns:svg="http://www.w3.org/2000/svg"
         width="12cm" height="10cm">
 <svg:ellipse rx="110" ry="130" />
 <svg:rect x="4cm" y="1cm" width="3cm" height="6cm" />
</svg:svg>
```

Bindings have scope within the element where they're declared.

An SVG processor recognizes all three of these elements as SVG elements because they all have prefixes bound to the particular URI defined by the SVG specification.

Binding Prefixes to Namespace URIs: Example

```
<xhtml:html xmlns:xhtml="http://www.w3.org/1999/xhtml"
  xmlns:mathml="http://www.w3.org/1998/Math/MathML">
  <xhtml:head>
    <xhtml:title> Three Namespaces </xhtml:title>
  </xhtml:head>
  <xhtml:body>
    <xhtml:h1 align="center">An Ellipse and a
        Rectangle</xhtml:h1>
    <svg:svg xmlns:svg="http://www.w3.org/2000/svg"
        width="12cm" height="10cm">
    <svg:ellipse rx="110" ry="130" />
    <svg:rect x="4cm" y="1cm" width="3cm" height="6cm"
  />
    </svg:svg>
  <xhtml:p>The equation for ellipses</xhtml:p>
  <mathml:math>
  <mathml:apply> <mathml:eq/> <mathml:cn> 1 </mathml:cn>
  <mathml:apply> <mathml:plus/>
  <mathml:apply> <mathml:divide/> <mathml:apply>
  ...
```

Binding with #FIXED ATTLIST Declarations

```
<!ATTLIST svg:svg xmlns:svg (CDATA) #FIXED
"http://www.w3.org/2000/svg">

<svg:svg width="12cm" height="10cm">

<svg:ellipse rx="110" ry="130" />

<svg:rect x="4cm" y="1cm" width="3cm" height="6cm" />
</svg:svg>
```

Namespaces and Attributes (I)

- An attribute can be placed in a namespace by prefixing its name.
- Normally only used for global attributes that can appear on any element such as *XLink* attributes rather than attributes that are tied to a specific element.
- Unprefixed attributes are never in any namespace.
- Being an attribute of an element in the http://www.w3.org/1999/xhtml namespace is not sufficient to put the attribute in the http://www.w3.org/1999/xhtml namespace.
- The only way an attribute belongs to a namespace is if it has a declared prefix, like *xlink:type* and *xlink:href*.

Namespaces and Attributes (II)

```
<xhtml:html xmlns:xhtml="http://www.w3.org/1999/xhtml"
   xmlns:xlink="http://www.w3.org/1999/xlink">
<xhtml:head>
<xhtml:title>Three Namespaces</xhtml:title>
</xhtml:head>
<xhtml:body>
 <xhtml:h1 align="center">An Ellipse and a
    Rectangle</xhtml:h1>
 <svg:svg xmlns:svg="http://www.w3.org/2000/svg"
    width="12cm" height="10cm">
    <svg:ellipse rx="110" ry="130" />
    <svg:rect x="4cm" y="1cm" width="3cm"
    height="6cm" />
 </svg:svg>
 <xhtml:p xlink:type="simple" xlink:href="ellip.html">
    More about ellipses </xhtml:p>
 <xhtml:p xlink:type="simple" xlink:href="rects.html">
    More about rectangles </xhtml:p>
...
```

URIs Matter, not Prefixes

- Many XML applications have recommended prefixes. For example, SVG elements often use the prefix svg and Resource Description Framework (RDF) elements often have the prefix rdf. However, these prefixes are simply conventions, and can be changed based on necessity, convenience or whim.
- Before a prefix can be used, it must be bound to a URI.
- These URIs are standardized, not the prefixes.
- The prefix can change as long as the URI stays the same.

The Default Namespace

- Indicates that an unprefixed element and all its unprefixed descendant elements belong to a particular namespace by attaching an xmlns attribute with no prefix:

```
<DATASCHEMA xmlns="http://www.w3.org/2000/P3Pv1">
<DATA name="vehicle.make" type="text" short="Make"
category="preference" size="31"/>
<DATA name="vehicle.model" type="text" short="Model"
category="preference" size="31"/>
<DATA name="vehicle.year" type="number" short="Year"
category="preference" size="4"/>
<DATA name="vehicle.license.state." type="postal."
short="State" category="preference" size="2"/>
<DATA name="vehicle.license.number" type="text"
short="License Plate Number" category="preference"
size="12"/>
</DATASCHEMA>
```

- Default namespaces apply only to elements, not to attributes. The name, type, short, category, and size attributes are not in any namespace.

Multiple Default Namespaces

- You can change the default namespace within a particular element by adding an xmlns attribute to the element.

```
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:xlink="http://www.w3.org/1999/xlink">
<head><title>Three Namespaces</title></head>
<body>
<h1 align="center">An Ellipse and a Rectangle</h1>
<svg xmlns="http://www.w3.org/2000/svg" width="12cm"
height="10cm">
<ellipse rx="110" ry="130" />
<rect x="4cm" y="1cm" width="3cm" height="6cm" />
</svg>
<p xlink:type="simple" xlink:href="ellipses.html">
More about ellipses </p>
<p xlink:type="simple" xlink:href="rectangles.html">
More about rectangles </p> <hr/> <p>Last Modified
February 13, 2000</p> </body> </html>
```

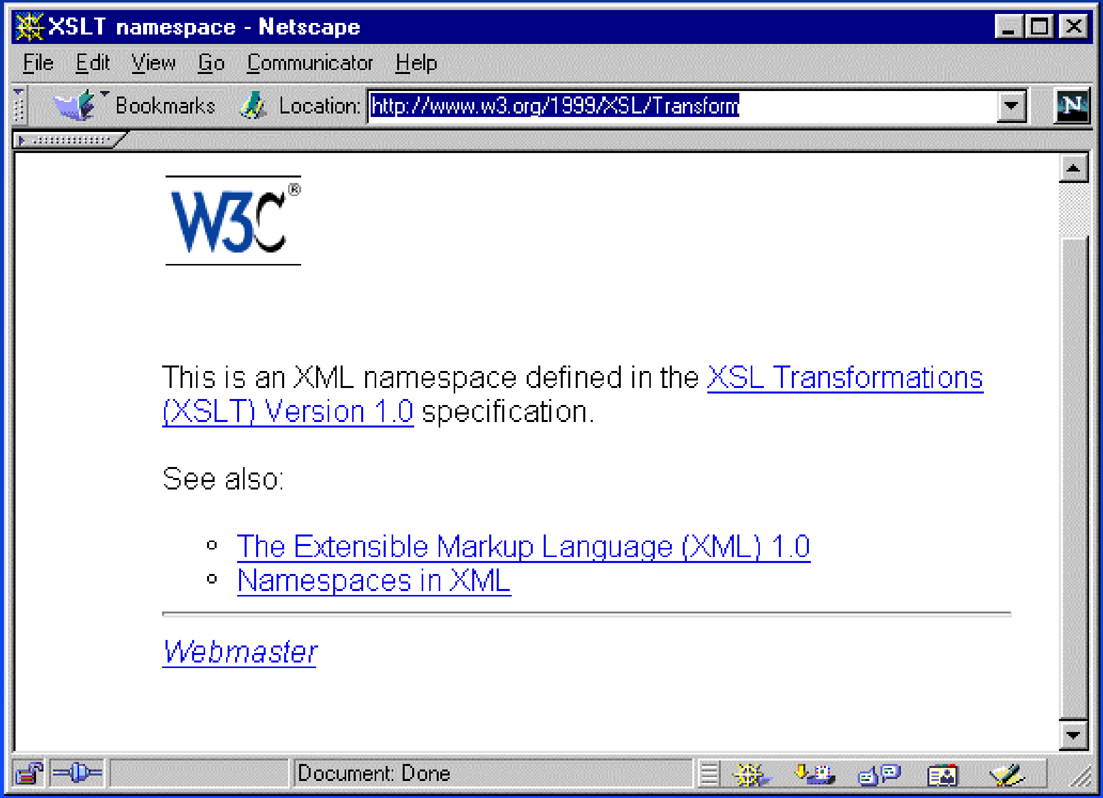
Binding with #FIXED ATTLISTs

```
<!ATTLIST svg xmlns (CDATA) #FIXED
  "http://www.w3.org/2000/svg">


<svg width="12cm" height="10cm">
<ellipse rx="110" ry="130" />
<rect x="4cm" y="1cm" width="3cm"
  height="6cm" />
</svg>
```

Where Namespace URIs Point to?

- Purely formal
- Can point somewhere but do not have to


XSLT namespace - Netscape
File Edit View Go Communicator Help
Bookmarks Location: http://www.w3.org/1999/XSL/Transform
Document: Done

W3C®

This is an XML namespace defined in the XSL Transformations (XSLT) Version 1.0 specification.

See also:

- The Extensible Markup Language (XML) 1.0
- Namespaces in XML

Webmaster

Parsers Compare Namespace URIs Character by Character

- These are three different namespaces:

```
http://www.w3.org/1999/XSL/Transform
http://www.w3.org/1999/XSL/Transform/
http://www.w3.org/1999/XSL/Transform/index.html
```

Namespaces and DTDs

- DTDs must declare the qualified names

```
<!ELEMENT svg:text (#PCDATA)>
```

- If the prefix changes, the DTD needs to change too.

Parameter Entity References for Namespace Prefixes

- Must use double indirection

```
<!ENTITY % NS.prefixed "IGNORE" >
<!ENTITY % MATHML.prefixed "%NS.prefixed;" >
<!ENTITY % MATHML.xmlns "http://www.w3.org/1998/Math/MathML" >
<!ENTITY % MATHML.prefix "m" >
<![%MATHML.prefixed;[ <!ENTITY % MATHML.xmlns.extra.attrib ""
  > ]]>
<!ENTITY % MATHML.xmlns.extra.attrib "" >
<![%MATHML.prefixed;[ <!ENTITY % MATHML.pfx "%MATHML.prefix;:"
  >
<!ENTITY % MATHML.xmlns.attrib "xmlns:%MATHML.prefix; CDATA
  #FIXED '%MATHML.xmlns;' %MATHML.xmlns.extra.attrib;" > ]]>
<!ENTITY % MATHML.pfx "" >
<!ENTITY % MATHML.xmlns.attrib "xmlns CDATA #FIXED
  '%MATHML.xmlns;' %MATHML.xmlns.extra.attrib;" >
<![%NS.prefixed;[ <!ENTITY % XHTML.xmlns.extra.attrib
  "%MATHML.xmlns.attrib;" > ]]>
<!ENTITY % XHTML.xmlns.extra.attrib "" > <!-- Section B:
  MathML Qualified Names :::::::::::::::::::::::::::::: --> <!-
```

Overriding the Prefix in the Internal DTD Subset

```
<?xml version="1.0"?>
<!DOCTYPE math:math PUBLIC "-//W3C//DTD MathML
2.0//EN" "mathml2.dtd" [ <!ENTITY % NS.prefixed
"INCLUDE" > <!ENTITY % MATHML.prefix "math" > ]>
<math:math xmlns="http://www.w3.org/1998/Math/MathML">
<math:apply> <math:eq/> <math:cn> 1 </math:cn>
<math:apply> <math:plus/> <math:apply> <math:divide/>
<math:apply> <math:power/> <math:ci> x </math:ci>
<math:cn> 2 </math:cn> </math:apply>
<math:apply> <math:power/> <math:ci> a </math:ci>
<math:cn> 2 </math:cn>
  </math:apply>
   </math:apply>
   <math:apply> <math:divide/>
    <math:apply> <math:power/> <math:ci> y </math:ci>
      <math:cn> 2 </math:cn>
    </math:apply>
    <math:apply> <math:power/> <math:ci> b </math:ci>
      <math:cn> 2 </math:cn>
    </math:apply>
```

Namespaces and Schemas

- URI matters; prefix doesn't
- Do not have to declare xmlns and xmlns:*prefix* attributes
- Namespaces are used on elements and attribute values, but not attribute names

How Parsers Handle Namespaces

- Namespaces were added to XML 1.0 after the fact,
  but care was taken to ensure backwards compatibility.
- An XML 1.0 parser that does not know about namespaces will most likely not have any troubles reading a document that uses namespaces.
- A namespace aware parser also checks to see that all prefixes are mapped to URIs. Otherwise it behaves almost exactly like a non-namespace aware parser.
- Other software that sits on top of the raw XML parser, an XSLT engine for example, may treat elements differently depending on what namespace they belong to. However, the XML parser itself mostly doesn't care as long as all well-formedness and namespace constraints are met.
- A possible exception occurs in the unlikely event that elements with different prefixes belong to the same namespace or elements with the same prefix belong to different namespaces
- Many parsers have the option of whether to report namespace violations so that you can turn namespace processing on or off as you see fit.