XPath Tutorial

DAN OLTEANU

INSTITUT FÜR INFORMATIK, LMU MÜNCHEN

20.06.2001

XPath 1.0 - W3C Recommendation

- language for **addressing** and **matching** parts of an XML document
- provides also basic facilities for strings, numbers and booleans **manipulation**
- uses a compact, **non-XML** format to facilitate use within URIs and attribute values
- supports namespaces
- designed to be used by **XPointer**, **XSLT**, **XQuery**

XPath: The Data Model

- ➜ can be derived from **XML InfoSet**
- ➜ XML document is viewed as a **tree**
- ➜ types of nodes: root, element, text, attribute, namespace, processing instruction, comment nodes
- ➜ imposes a document **order** defined on all nodes except attribute and namespace nodes (order of occurence of elements start-tag)
- ➜ the root node is the **first** node
- ➜ the namespace nodes are defined to occur **before** the attribute nodes
- ➜ root and element nodes have an **ordered list of child** nodes
- ➜ an element node is the **parent** of the associated **set** of attribute/namespace nodes

  the attributes/namespaces **are not children** of the associated element node

XPath Expressions: Syntactic Building Blocks

- **primary** expressions: location paths, predicates, function calls, variable reference, strings, numbers, booleans
- **complex** expressions: unions, filters, relational expressions
- basic **types** for an expression: node-set, string, number, boolean
- expression **evaluation** occurs wrt a context, made from:
  - a node (the context **node**)
  - a pair of positive integers (the context **position** and the context **size**)
  - a set of variable bindings
  - a function library
  - a set of namespace declarations in expression scope
- examples
  - ($x + $y) · 2 > 10.7
  - //lecture(@name = 'Markup' and contains(author,'Bry'))
  - 'Markup-Sprachen und semi-strukturierte Daten'

Location Paths: The MOST Important Constructs

A path is constructed from steps, which have:

- ➜ an **axis**, which specifies the tree relationship between the nodes
- ➜ a **node** test, which specifies the node type of the selected nodes
- ➜ **predicates**, which refine the set of nodes selected by the location step

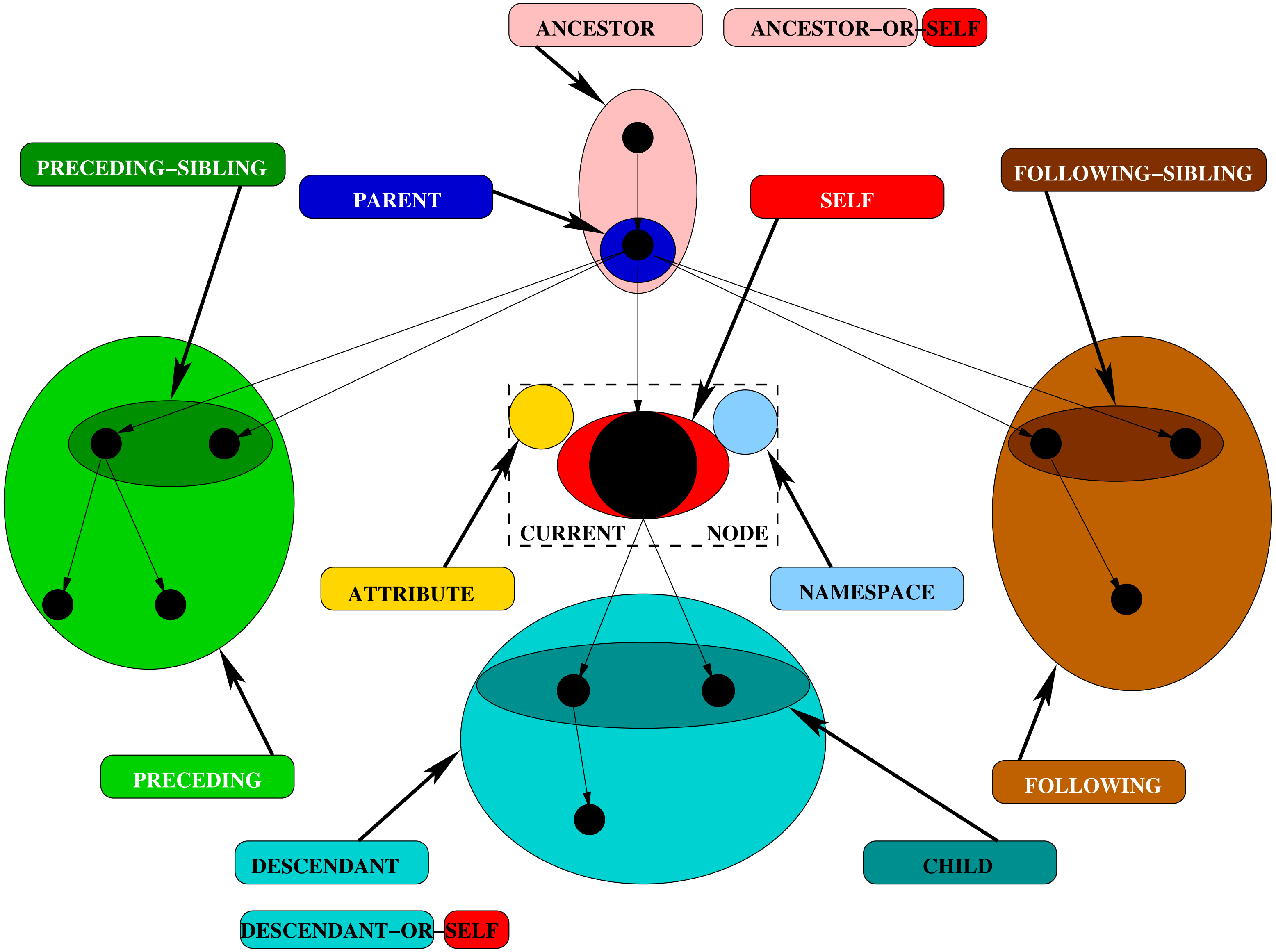
Examples:

- ➜ **child**::para
- ➜ **child**::para[**attribute**::type="warning"]
- ➜ **child**::*
- ➜ **child**::*[**self**::chapter or **self**::appendix][position()=last()]

Abbreviated syntax:

- ➜ **@name** instead of *attribute::name*
- ➜ **para[1]** instead of *child::para[position()=1]*
- ➜ **.//para** instead of *self::node()/descendant-or-self::node()/child::para*
- ➜ **.** instead of *self::node()*
- ➜ **../para** instead of *parent::node()/child::para*

AXES IN XPATH 1.0



CORE FUNCTION LIBRARY: NODE SET FUNCTIONS

- *number* **last()**
  returns the context size from the expression evaluation context
- *number* **position()**
  returns the context position from the expression evaluation context
- *number* **count(*node-set*)**
  returns the number of nodes in the argument node-set
- *node-set* **id(*object*)**
  selects elements by their unique ID, as declared in DTD
- *string* **name(*node-set*?)** , *string* **local-name(*node-set*?)**, *string* **namespace-uri(*node-set*?)**
  returns the expanded/local name/namespace URI of the node in the argument node-set that is first in document order

Core Function Library: String Functions

- *string* **string(*object*?)**
- *string* **concat(*string*,*string*, *string**)**
- *boolean* **start-with(*string*, *string*)**
- *boolean* **contains(*string*, *string*)**
- *string* **substring-before(*string*, *string*)**
- *string* **substring-after(*string*, *string*)**
- *string* **substring(*string*, *number*, *number*?)**
- *number* **string-length(*string*?)**
- *string* **normalize-space(*string*?)**
- *string* **translate(*string*, *string*, *string*)**

Core Function Library: Boolean and Number Functions

Boolean Functions

- ➜ *boolean* **boolean(*object*)**
- ➜ *boolean* **not(*boolean*)**
- ➜ *boolean* **true()**
- ➜ *boolean* **false()**

Number Functions

- ➜ *number* **number(*object*?)**
- ➜ *number* **sum(*node-set*?)**
- ➜ *number* **floor(*number*)**
- ➜ *number* **ceiling(*number*)**
- ➜ *number* **round(*number*)**

Some examples (1)

- Find the *chapter* elements without attributes
  //chapter(not(@*))

- Find *chapter*s having three *para*graphs
  //chapter(count(.//para) = 3)

- Find the first *para*graph of each *chapter*
  //chapter//para(1)

- Find the first *para*graph from the *chapter*s
  (//chapter//para)(1)

- Find the *figure*s with *caption* 'Strawberry' in the second *chapter*
  //chapter(2)//figure(caption = 'Strawberry')

- Find all the *figure*s in *chapter*s 2 through 5
  //chapter(position() >= 2 and position() <=5)//figure

- Find *caption*s of *figure*s that are referenced by *figref* elements in the *chapter* with title 'Strawberry'
  id(//chapter(title = "Strawberry")//figref/@refid)(self::fig)/caption

Some Examples (2)

- Find *chapters* in which 'Strawberry' is mentioned in at least one *paragraph*
  //chapter(.//para(contains(.,'Strawberry')))

- Find *chapters* in which 'Strawberry' is mentioned in every *paragraph*
  //chapter(count(.//para) = count(.//para(contains(.,'Strawberry'))) and .//para)
  OR
  //chapter(not(.//para(not(contains(.,'Strawberry')))) and .//para)

- List the *names* of the second-level *managers* of all *employees* whose *rating* is 'Good'
  id(id(/emp(rating = "Good")/@mgr)(self::emp)/@mgr)(self::emp)/name

- List all distinct *employees* from a *company*
  (//company//employee)(not(.=preceding::employee))
- Prepare a critical sequence report consisting of all elements that occur between the first and second *incision*
  (//incision(2)/preceding::*) (count(. | (//incision(1)/following::*)) = count (//incision(1)/following::*))

WHAT'S COMING? XPATH 2.0 REQUIREMENTS

- explicit "**For Any**" and "**For All**" comparison and equality semantics
- extended set of **aggregation** functions (e.g. min, max, avg ..)
- node-set **intersection** and **difference** functions
- **conditional** expressions (if-then-else)
- string matching using **regular expressions**
- support for **XML Schema** primitive datatypes

USEFUL LINKS

- W3C XPath Recommendation
  http://www.w3.org/TR/xpath

- XPath Tutorial from Zvon
  http://www.zvon.org/xxl/XPathTutorial/General/examples.html

- XPath Tester from 5Sight
  http://www.fivesight.com/downloads/xpathtester.asp

- XPath Tester from PhPXML
  http://www.phpxml.org/scripts/testsuite/

- XPath Tutorial from Resin
  http://www.caucho.com/products/resin/ref/xpath.xtp

- XPath Implementation from Xalan
  http://xml.apache.org/xalan-j/index.html