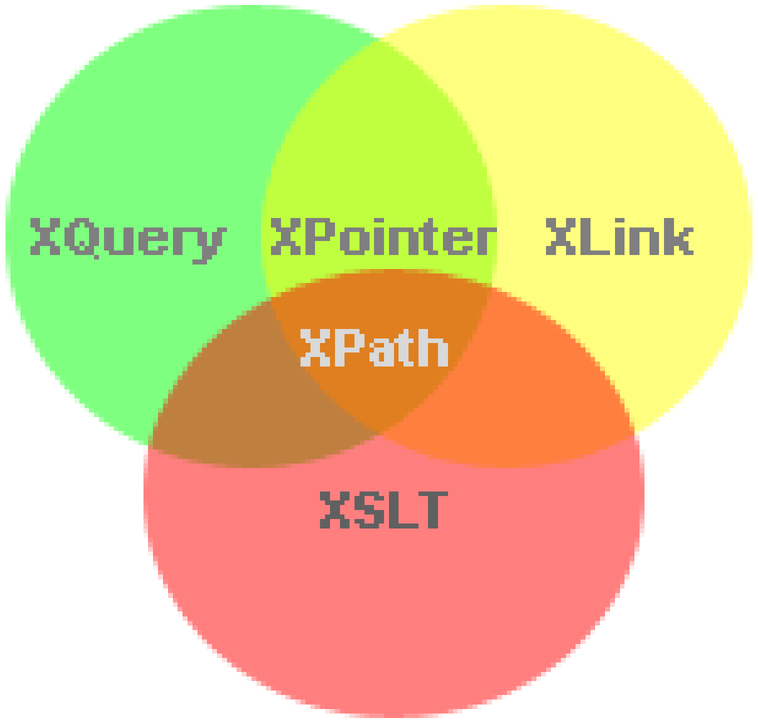
XPath and XQuery

Instructor: Purkayastha, S.

XML XPath

A query language for selecting XML nodes


XQuery
XPointer
XLink
XPath
XSLT

XPath

- XPath stands for *XML Path Language*
- XPath is a language for finding information in an XML document.
- It is used to navigate through elements and attributes in an XML document
- Other XML technologies are based on XPath e.g. XSLT, XQuery and XPointer.
- Understanding XPath is fundamental to a lot of advanced XML usage.
- It use path expressions to select nodes or node-sets in an XML document
- Simliar as path expressions in a traditional computer file system
- XPath contains a library of standard functions
- XPath is a W3C Standard (1999)

Where to use XPath?

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<bookstore>
<book category=“HISTORY“ year=“1998”>
  <title lang=“en">300</title>
  <author>Frank Miller</author>
  <author>Lynn Varley</author>
  <price>17.90</price>
</book>
<book category=“SF“ year=“1982”>
  <title lang="en">Foundation and Empire</title>
  <author>Isaac Asimov</author>
  <price>34.27</price>
</book>
<book category="CHILDREN“ year=“2005”>
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <price>29.99</price>
</book>
<book category="COOKING“ year=“2005”>
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <price>30.00</price>
</book>
</bookstore>
```

Select particular parts of this document, some possible queries:

- **Give me all books from 2005**
- **All titles**
- **All books having the price lower than 30**

etc.

All this is expressible in XPath!

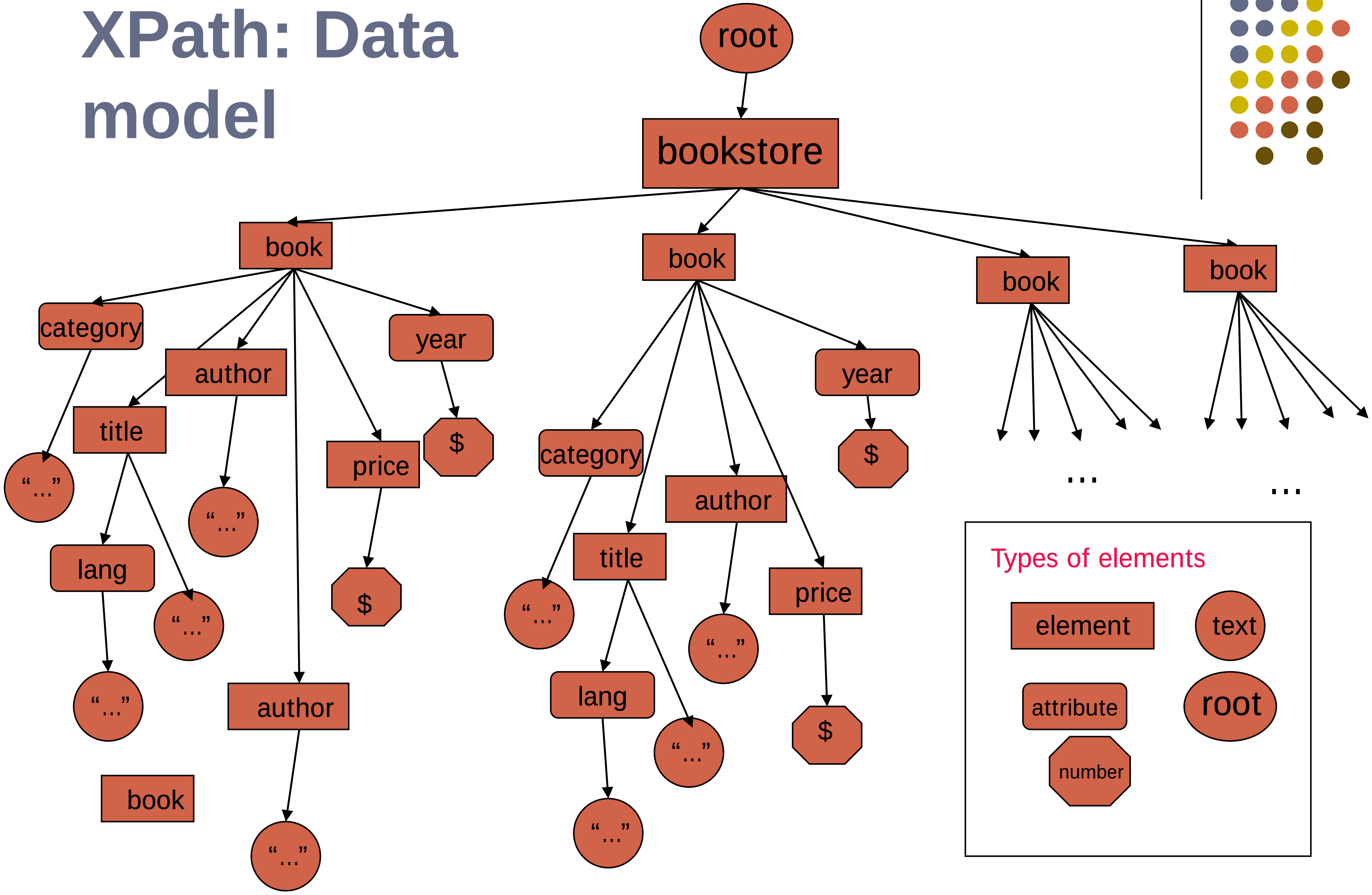
XPath overview

- **Data Model**
- Path Expression
- Examples

XPath: Data model

- XML documents are treated as tree of nodes
- Nodes can be: elements, attributes, text, namespace, comment, and document (root) node
- An order relation is defined over all nodes except attribute and namespace nodes
  - Root node is the *first*
  - Element node is the parent of associated set of attributes/namespaces
  - Relation between nodes: parent, children, siblings, ancestors, descendants

XPath: Data model



Look at this slide and also look at slide 5

An example node

```
<?xml version=“1.0” encoding=“ISO-8859-1” ?>

<bookstore>

<book>
  <title lang=“en“>Harry Potter</title>
       <!-- here “lang“ is attribute node -->
  <author>J. K. Rowling</author> <!-- element node -->
  <year>2005</year>
  <price>29.99</price>
</book>

</bookstore>  <!--  document node (root node) -->
```

Relationship of nodes

- Parent
  - Each element and attribute has one parent
- Children
  - Element nodes may have zero, one or more children
- Siblings
  - Nodes that have the same parent
- Ancestors
  - A node‘s parent, grand-parent, grand-grand-parent, etc.
- Descendants
  - A node‘s children, grand-children, grand-grand-children, etc.

Examples

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <author>J. K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
</bookstore>
```

- Book element is the parent of the title, author, year and price;
- The title, author, year and price are the children of book element;
- The title, author, year and price are all siblings;
- Bookstore element and book element are the ancestors of the title element;
- Descendants of the bookstore element are the book, title, author, year and price elements.

XPath Axes

- An axis defines a node-set relative to the current node

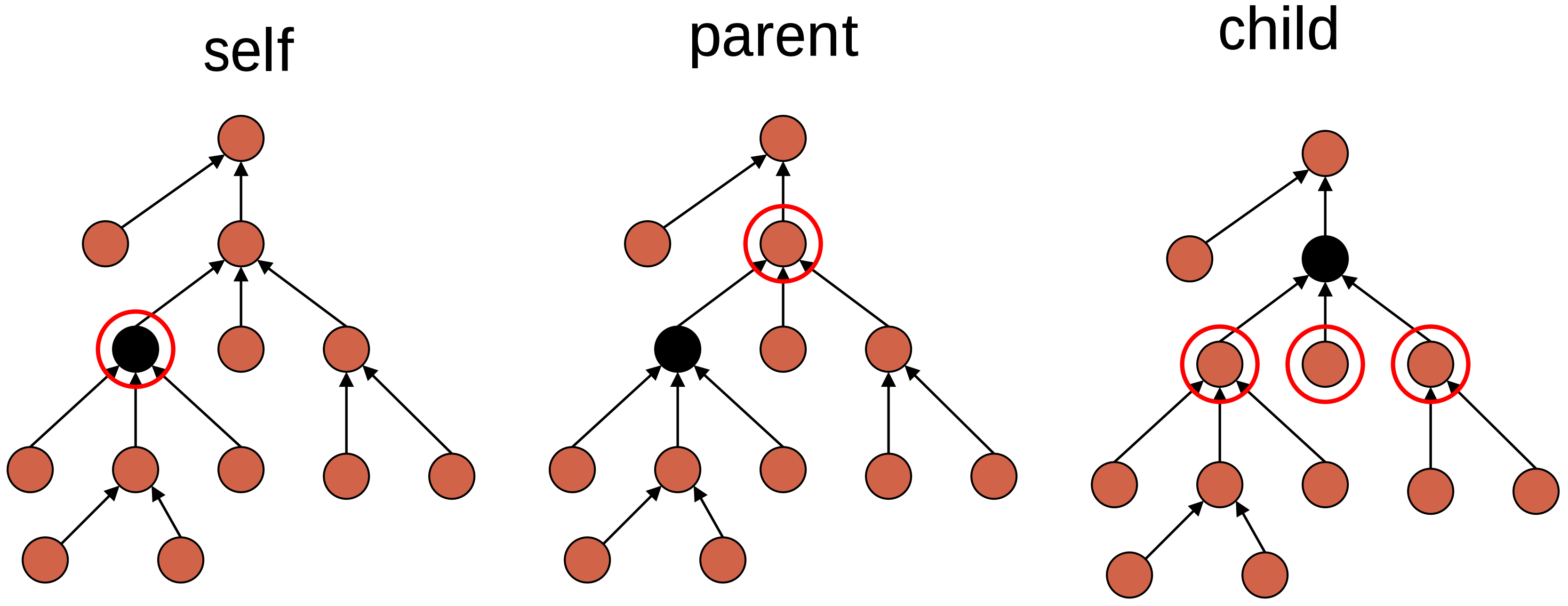
XPath Axes (I)

Axis Name	Result
ancestor	Selects all ancestors (parent, grandparent, etc.) of the current node
ancestor-or-self	Selects all ancestors (parent, grandparent, etc.) of the current node and the current node itself
attribute	Selects all attributes of the current node
child	Selects all children of the current node
descendant	Selects all descendants (children, grandchildren, etc.) of the current node
descendant-or-self	Selects all descendants (children, grandchildren, etc.) of the current node and the current node itself

XPath Axes (II)

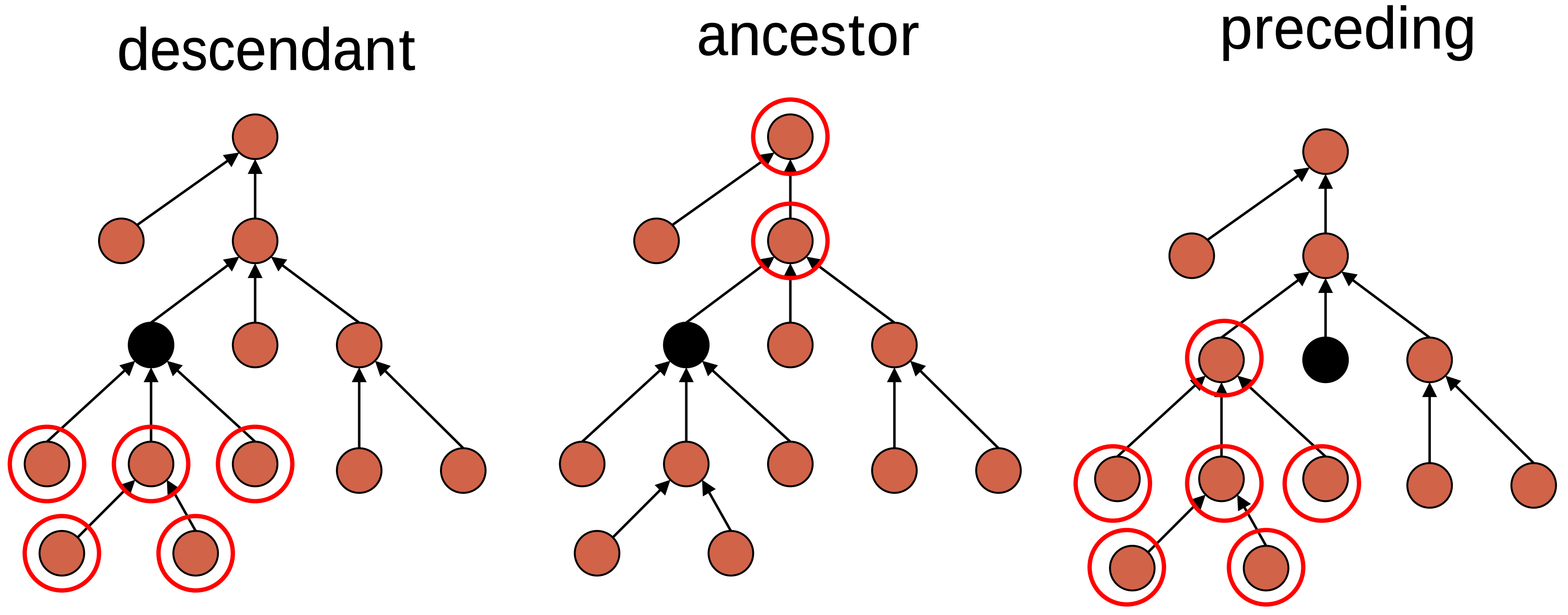
Axis Name	**Result**
following	Selects everything in the document after the closing tag of the current node
following-sibling	Selects all siblings after the current node
namespace	Selects all namespace nodes of the current node
parent	Selects the parent of the current node
preceding	Selects everything in the document that is before the start tag of the current node
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

XPath Axes



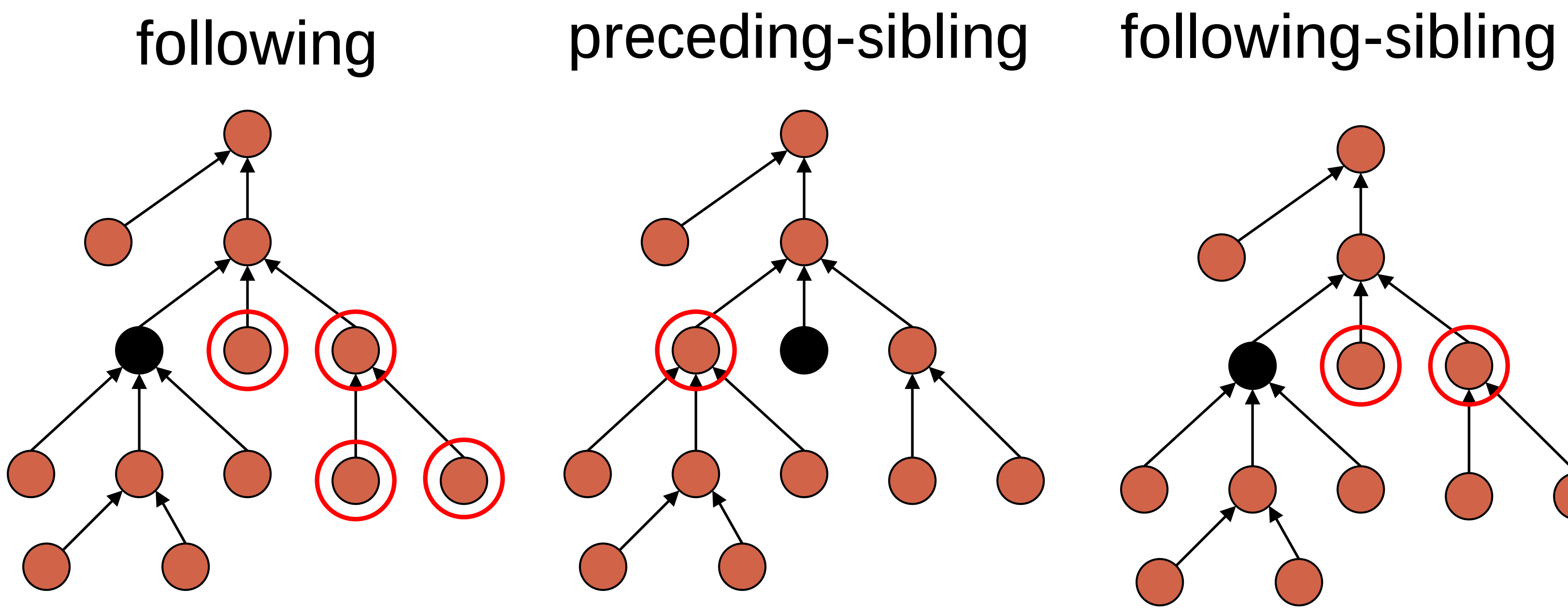
* See the red circles around nodes

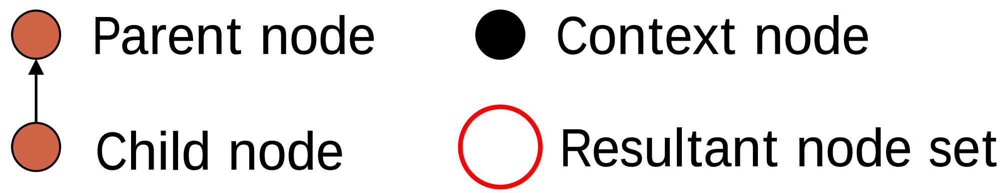
XPath Axes



*** See the red circles around nodes**

XPath Axes





*** See the red circles around nodes**

XPath overview

- Data Model
- **Path Expressions**
- Examples

XPath

- It uses path expressions to select nodes or node-sets in an XML document.
- The node is selected by following a path or steps
- They can be absolute or relative path.
- A path is constructed from steps

Absolute/Relative Path

- Absolute or relative path
  - An absolute location path starts with a slash (/)
  - A relative location path does not start with a slash (/)
  - Both location path consists of one or more steps, each separated by a slash (/)

```
An absolute location path:
/step/step/…

A relative location path:
step/step/…
```

XPath expression table

Expression	**Description**
nodename	Selects all child nodes of the named node
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Selecting node examples

Expression	Description
bookstore	Selects all child nodes of the bookstore element
/bookstore	Selects the root node bookstore Note: if the path starts with a slash (/), it always represents an absolute path to an element
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

Predicates

- Used to find a specific node or a node that contains a specific value,
- Predicates are always embeded in square brackets

Predicate examples

Path Expression	Result
/bookstore/book[1]	Selects the first book element that is the child of the bookstore element Note: IE5 and later has implemented that [0] is the first node, but according to W3C, it should be [1]
/bookstore/book[last()]	Selects the last book element that is the child of the bookstore element
/bookstore/book[last()-1]	Selects the last but one book element that is the child of the bookstore element
/bookstore/book[position()<3]	Selects the first two book elements that are children of the bookstore element
//title[@lang]	Selects all the title elements that have an attribute named lang
//title[@lang=‘eng‘]	Selects all title elements that have an attribute named lang with a value of ‘eng‘
/bookstore/book[price>35.00]	Selects all the book elements of the bookstore element that have a price element with a value greater than 35.00
/bookstore/book[price>35.00] /title	Selects all the title elements of the book elements of the bookstore elements that have a price with a value greater than 35.00

Selecting unknown nodes

Wildcard	Description
*	Matches any element node
@*	Matches any attribute node
node()	Matches any node of any kind

Examples

Path Expression	**Result**
/bookstore/*	Selects all the child nodes of the bookstore element
//*	Selects all elements in the document
//title[@*]	Selects all title elements which have any attribute

Selecting several paths

- Using the | operator in an XPath expression to select several paths

Path Expression	Result
//book/title | //book/price	Selects all the title AND price elements of all book elements
//title | //price	Selects all title AND price elements in the document
/bookstore/book/title | //price	Selects all the title elements of the book elements of the bookstore elements AND all the price elements in the document

XPath Operators

- An XPath expression returns either a node-set, a string, a boolean, or a number

XPath operators (I)

Operator	Description	Example	Return value
|	Computes two node-sets	//book | //cd	a node-set with all book and cd elements
+	addition	6+4	10
-	subtraction	6-4	2
*	multiplication	6*4	24
div	division	8 div 4	2
mod	Modulus (division remainder)	5 mod 2	1
=	equal	price=9.80	True if price is 9.80 False if price is not 9.80

XPath operators (II)

Operator	Description	Example	Return value
!=	Not equal	price!=9.80	True if price is not 9.80 False if price is 9.80
<	Less than	price<9.80	True if price is 9.00 False if price is 9.90
<=	Less than or equal to	price<=9.80	True if price is 9.00 False if price is 9.90
>	Greater than	price>9.80	True if price is 9.90 False if price is 9.00
>=	Greater than or equal to	price>=9.80	True if price is 9.90 False if price is 9.00
or	or	price=9.80 or price=9.70	True if price is 9.80 False if price is 9.00
and	and	price>9.00 and price<9.90	True if price is 9.80 False if price is 8.00

XPath overview

- Data Model
- Path Expression
- **Examples**

XPath examples

- books.xml
- Different browsers deal with XML and XPath in different ways
- Our examples should work with most major browsers

Select nodes

- Select all book title nodes:
  - /bookstore/book/title
- Select the first book title node:
  - /bookstore/book[0]/title

Select nodes

- Select the prices
  - /bookstore/book/price
- Select title nodes with price>35
  - /bookstore/book[price>35]/title
- Select price nodes with price>35
  - /bookstore/book[price>35]/price

XPath: Summary

- Simple but powerful query & navigation language for XML trees
- Allows to almost arbitrarily select parts of the XML Tree.
- Many useful built-in functions!

XML XQuery

A query language like SQL for XML

XQuery

- XQuery is the language for querying XML data
- XQuery for XML is like SQL for databases
- XQuery is built on XPath expressions
- XQuery is supported by all the major database engines (IBM, Oracle, Microsoft, etc.)
- XQuery is a W3C Recommendation (Jan 2007)

Querying XML data

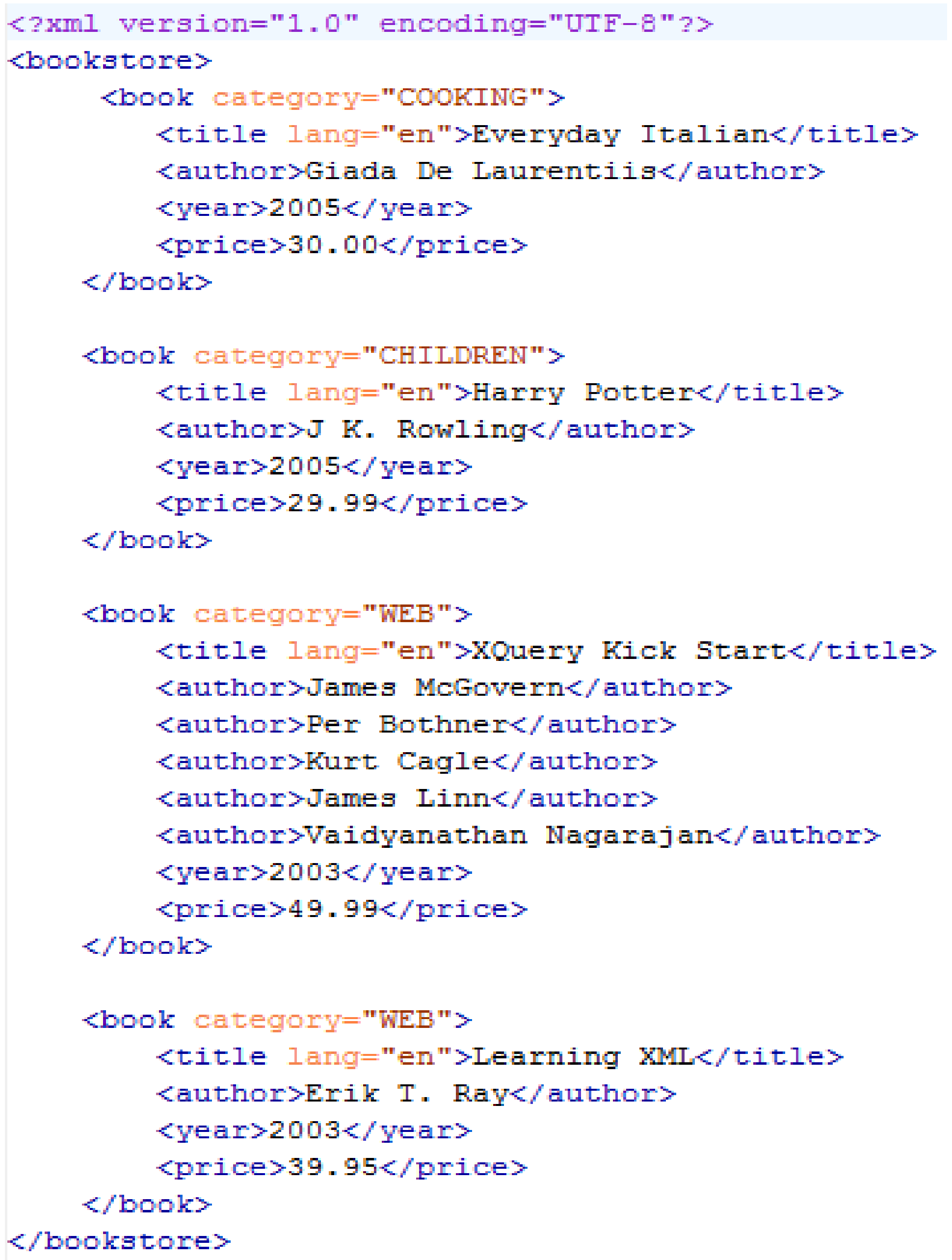
- XQuery provides ways to find and extract elements and attributes from XML documents
  - Select all CD records with a price less than $10 from the CD collection stored in the XML document called cd_catalog.xml

- XQuery can be used for:
  - Extracting information from a web service
  - Generating summary reports
  - Transform XML data to HTML
  - Search web documents for relevant information

Querying XML data

- XQuery provides ways to find and extract elements and attributes from XML documents
  - Select all CD records with a price less than $10 from the CD collection stored in the XML document called cd_catalog.xml
- XQuery 1.0 and XPath 2.0 share the same data model and support the same functions and operators
- XQuery can used for:
  - Extracting information from a web service
  - Generating summary reports
  - Transform XML data to XHTML
  - Search web documents for relevant information

XML Example



```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
     <book category="COOKING">
        <title lang="en">Everyday Italian</title>
        <author>Giada De Laurentiis</author>
        <year>2005</year>
        <price>30.00</price>
    </book>

    <book category="CHILDREN">
        <title lang="en">Harry Potter</title>
        <author>J K. Rowling</author>
        <year>2005</year>
        <price>29.99</price>
    </book>

    <book category="WEB">
        <title lang="en">XQuery Kick Start</title>
        <author>James McGovern</author>
        <author>Per Bothner</author>
        <author>Kurt Cagle</author>
        <author>James Linn</author>
        <author>Vaidyanathan Nagarajan</author>
        <year>2003</year>
        <price>49.99</price>
    </book>

    <book category="WEB">
        <title lang="en">Learning XML</title>
        <author>Erik T. Ray</author>
        <year>2003</year>
        <price>39.95</price>
    </book>
</bookstore>
```

Selecting nodes

- XQuery uses functions to extract data from XML documents
  - doc() function is used to open one xml file

```
doc("books.xml")
```

- XQuery uses path expressions to navigate through elements in an XML document
  - Select all the title elements in the books.xml file

```
doc("books.xml")/bookstore/book/title
```

```
<title lang="en">Everyday Italian</title>
<title lang="en">Harry Potter</title>
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

title.xq

Predicates

- XQuery uses predicates to limit the extracted data from XML documents
  - Select all the book that have a price less than 30

```
doc(“books.xml”)/bookstore/book[price<30]
```

```
<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

price30.xq

Beyond XPath

- We can select and filter elements with either a path expression or with a FLWOR expression.

FLWOR expressions

- FLWOR is acronym for For, Let, Where, Order by, Return
  - for
    - Optional
    - Binds a variable to each item returned by the “in” expression
  - let
    - Optional
    - define variable
  - where
    - Optional
    - Specifies a criteria
  - order by
    - Optional
    - Sorts the results before return
  - return
    - Result results

FLWOR Expression

```
for $x in doc(“books.xml”)/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

titleprice30.xq

The “for” clause

- The for clause binds a variable to each item returned by the “in” expression
- It results in iteration
- To use “at” keyword to count the iteration

For clause Examples

```
for $x in (1 to 5)
return <test>{$x}</test>
```

```
<test>1</test>
<test>2</test>
<test>3</test>
<test>4</test>
<test>5</test>
```

for.xq

For clause Examples

```
for $x at $i in doc(“books.xml”)/bookstore/book/title
return <book>{$i}.{data($x)}</book>
```

```
<book>1. Everyday Italian</book>
<book>2. Harry Potter</book>
<book>3. XQuery Kick Start</book>
<book>4. Learning XML</book>
```

for1.xq

For clause Examples

```
for $x in (10,20), $y in (100,200)
return <test>x={$x} and y={$y}</test>
```

```
<test>x=10 and y=100</test>
<test>x=10 and y=200</test>
<test>x=20 and y=100</test>
<test>x=20 and y=200</test>
```

for2.xq

The let clause

- It allows variable assignments
- It avoids repeating the same expression many times
- It does NOT result in iteration

```
let     $a := avg(doc("books.xml")/bookstore/book/price)
return  $a
```

```
37.5
```

```
let $x := (1 to 5)
return <test>{$x}</test>
```

let.xq

```
<test>12345</test>
```

The where clause

- It is used to specify one or more criteria for the result

```
for $x in doc(“books.xml”)/bookstore/book
where $x/price>30 and $x/price<100
return $x/title
```

where.xq

```
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

The order by clause

- It is used to specify the sort order of the result

```
for $x in doc(“books.xml”)/bookstore/book
order by $x/@category, $x/title
return $x/title
```

order.xq

```
<title lang=“en”>Harry Potter</title>
<title lang=“en”>Everyday Italian</title>
<title lang=“en”>Learning XML</title>
<title lang=“en”>XQuery Kick Start</title>
```

The return clause

- It specifies what is to be returned

```
for $x in doc(“books.xml”)/bookstore/book
return $x/title
```

return.xq

```
<title lang=“en”>Everyday Italian</title>
<title lang=“en”>Harry Potter</title>
<title lang=“en”>XQuery Kick Start</title>
<title lang=“en”>Learning XML</title>
```

XPath and FLWOR Examples

- Example

```
In XPath
doc(“books.xml”)/bookstore/book[price>30]/title

In FLWOR
for $x in doc(“books.xml”)/bookstore/book
where $x/price>30
return $x/title
```

```
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

FLWOR Examples

- Example

```
for     $b in doc(“books.xml“)/bookstore/book
let     $t := $b/title
where   $b/price > 30
order by   $t
return  ($b/author, $t)
```

```
<author>Erik T. Ray</author>
<title lang="en">Learning XML</title>
<author>James McGovern</author>
<author>Per Bothner</author>
<author>Kurt Cagle</author>
<author>James Linn</author>
<author>Vaidyanathan Nagarajan</author>
<title lang="en">XQuery Kick Start</title>
```

flwor1.xq

FLWOR Examples

- In XQuery and XPath, expressions within square brackets [] are subqueries, those expressions are not used to retrieve elements themselves, but to qualify the elements that are retrieved. For example:
  - /bookstore/book/price – retrieve price elements
  - /bookstore/book[price] – retrieve book elements that have a price element as a child.
  - //book[@category="WEB"] – retrieve book element which its category is WEB
  - for $x in //book
    return data($x/@category) – retrieve the value of the category of different book elements

FLWOR Examples

- Example

```
for    $v in //book[year=2005]
Return $v/title
```

```
<title lang="en">Everyday Italian</title>
<title lang="en">Harry Potter</title>
```

FLWOR + HTML

- Present the result in an HTML list

```
for $x in doc(“books.xml”)/bookstore/book/title
order by $x
return $x
```

```
<ul>
{
for $x in doc(“books.xml”)/bookstore/book/title
order by $x
return <li>{$x}</li>
}
</ul>
```

```
<ul>
<li><title lang=“en”>Everyday Italian</title></li>
<li><title lang=“en”>Harry Potter</title></li>
<li><title lang=“en”>Learning XML</title></li>
<li><title lang=“en”>XQuery Kick Start</title></li>
</ul>
```

html.xq

FLWOR + HTML

- If only want data inside the title element

```
for $x in doc(“books.xml”)/bookstore/book/title
order by $x
return $x
```

```
<ul>
{
for $x in doc(“books.xml”)/bookstore/book/title
order by $x
return <li>{data($x)}</li>
}
</ul>
```

```
<ul>
<li>Everyday Italian</li>
<li>Harry Potter</li>
<li>Learning XML</li>
<li>XQuery Kick Start</li>
</ul>
```

XQuery Syntax

- XQuery is case-sensitive
- XQuery elements, attributes, and variables must be vaild XML names
- An XQuery string value can be in single or double quotes
- An XQuery variable is defined with a $ followed by a name
  - e.g. $x
- XQuery comments are delimited by (: and :)
  - e.g. (: XQuery Comment :)

XQuery conditional expressions

- “if-then-else” in XQuery

```
for $x in doc(“books.xml”)/bookstore/book
return  if ($x/@category=“CHILDREN”)
        then <child>{data($x/title)}</child>
        else <adult>{data($x/title)}</adult>
```

```
<adult>Everyday Italian</adult>
<child>Harry Potter</child>
<adult>Learning XML</adult>
<adult>XQuery Kick Start</adult>
```

ifthen.xq

XQuery comparisons

- There are two ways of comparing values:
  - General comparisons:
    - =, !=, <, <=, >, >=
  - Value comparisons:
    - eq, ne, lt, le, gt, ge

```
$bookstore//book/@q > 10

The expression returns true if any q attributes have
values greater than 10

$bookstore//book/@q gt 10

The expression returns true if only one q attribute is
greater than 10. If more than one q is greater than 10,
then an error occurs
```

XQuery comparisons

- Example

```
let     $a := avg(doc("books.xml")/bookstore/book/price)
for     $b in doc("books.xml")/bookstore/book
where   $b/price > $a
return  $b/title
```

```
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

avg.xq

Adding elements and attributes

- Adding HTML elements and text

```
<html>
<body>
<h1>Bookstore</h1>
<ul>
{
for $x in doc(“books.xml”)/bookstore/book
order by $x/title
return <li>{data($x/title)}- Category:
{data($x/@category)}</li>
}
</ul>
</body>
</html>
```

bookstore.xq

Adding elements and attributes

- Adding HTML elements and text: Result

```
<html>
<body>
<h1>Bookstore</h1>
<ul>
<li>Every Italian. Category: COOKING</li>
<li>Harry Potter. Category: CHILDREN</li>
<li>Learning XML. Category: WEB</li>
<li>XQuery Kick Start. Category: WEB</li>
</ul>
</body>
</html>
```

Adding elements and attributes

- Adding attributes to HTML elements

```
<html>
<body>
<h1>Bookstore</h1>
<ul>
{
for $x in doc(“books.xml”)/bookstore/book
order by $x/title
return <li
class=“{data($x/@category)}”>{data($x/title)}</li>
}
</ul>
</body>
</html>
```

Adding elements and attributes

- Adding attributes to HTML elements: Result

```
<html>
<body>
<h1>Bookstore</h1>
<ul>
<li class="COOKING">Every Italian</li>
<li class="CHILDREN">Harry Potter</li>
<li class="WEB">Learning XML</li>
<li class="WEB">XQuery Kick Start</li>
</ul>
</body>
</html>
```

XQuery summary

- XQuery was designed to query anything that can appear as XML, including databases.
- XQuery is based on XPath for node set retrieval
- XQuery based on FLWOR expressions
- Many built-in operators and functions to facilitate data retrieval, modification

- Limitations: XQuery is "Read Only" - no capability for data manipulation operations such as SQL INSERT, UPDATE, DELETE