CSC-318 Web Technology

(BSc CSIT, TU)

Ganesh Khatri
kh6ganesh@gmail.com

XSL / XSLT

- XSL (eXtensible Stylesheet Language) is a styling language for XML.
- XSLT stands for XSL Transformations.
- Used to transform XML documents into other formats (like transforming XML into HTML).

XSL(T) Languages

- **XSLT** is a language for transforming XML documents.
- **XPath** is a language for navigating in XML documents.
- **XQuery** is a language for querying XML documents
- XSL stands for E**X**tensible **S**tylesheet **L**anguage.
- The World Wide Web Consortium (W3C) started to develop XSL because there was a need for an XML-based Stylesheet Language

XSL - More Than a Style Sheet Language

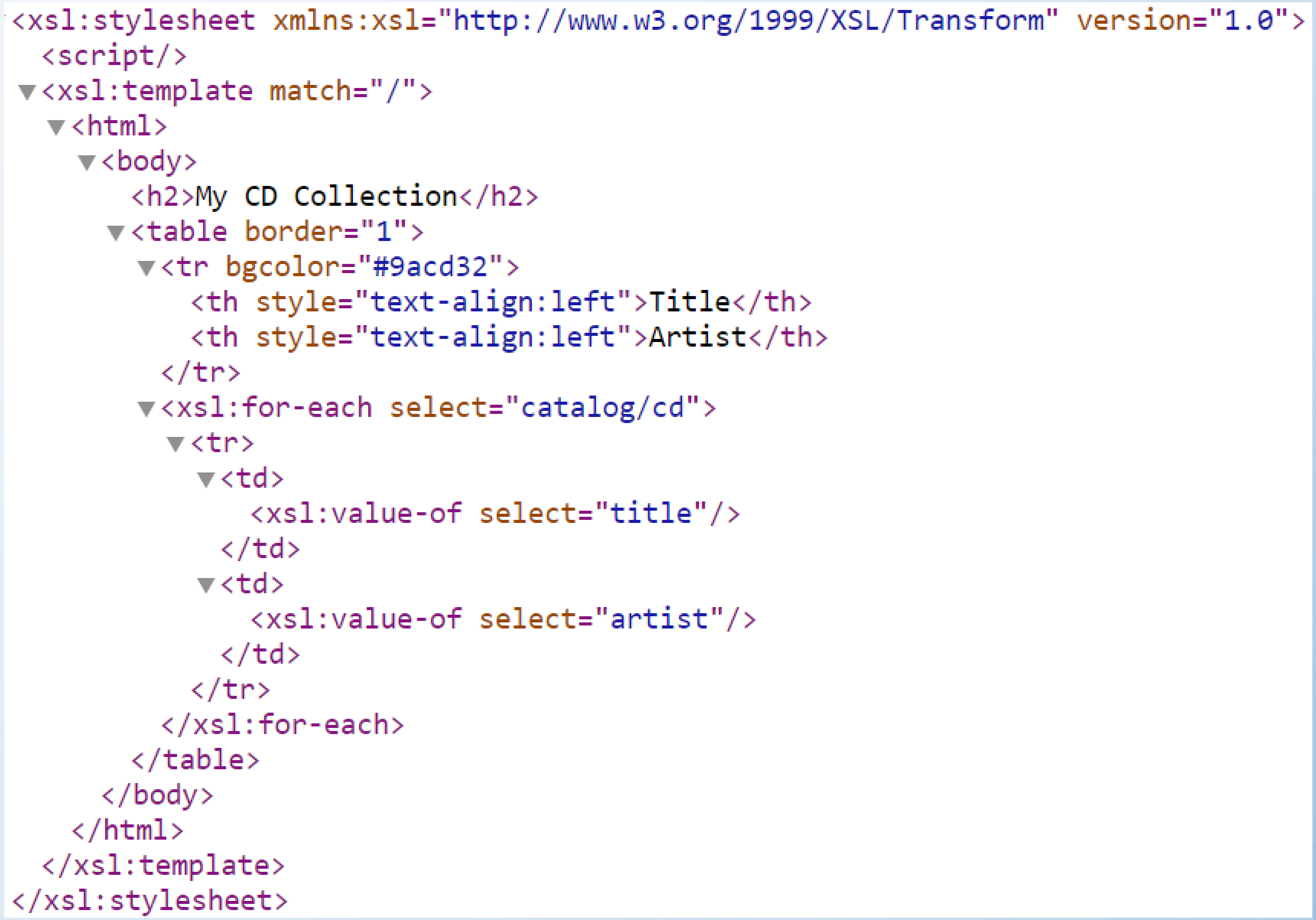
- XSL types
  - XSLT - a language for transforming XML documents
  - XPath - a language for navigating in XML documents
  - XQuery - a language for querying XML documents

XSLT = XSL Transformations

- XSLT is the most important part of XSL.
- XSLT is used to transform an XML document into another XML document, or another type of document that is recognized by a browser, like HTML.
- Normally XSLT does this by transforming each XML element into an HTML element.
- With XSLT, you can add/remove elements and attributes to or from the output file.
- You can also rearrange and sort elements, perform tests and make decisions about which elements to hide and display, and a lot more.
- A common way to describe the transformation process is to say that **XSLT transforms an XML source-tree into an XML result-tree**

Create an XSL Style Sheet

- you create an XSL Style Sheet ("cdcatalog.xsl") with a transformation template



```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <script/>
▼<xsl:template match="/">
  ▼<html>
    ▼<body>
        <h2>My CD Collection</h2>
      ▼<table border="1">
        ▼<tr bgcolor="#9acd32">
            <th style="text-align:left">Title</th>
            <th style="text-align:left">Artist</th>
          </tr>
        ▼<xsl:for-each select="catalog/cd">
          ▼<tr>
            ▼<td>
                <xsl:value-of select="title"/>
              </td>
            ▼<td>
                <xsl:value-of select="artist"/>
              </td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

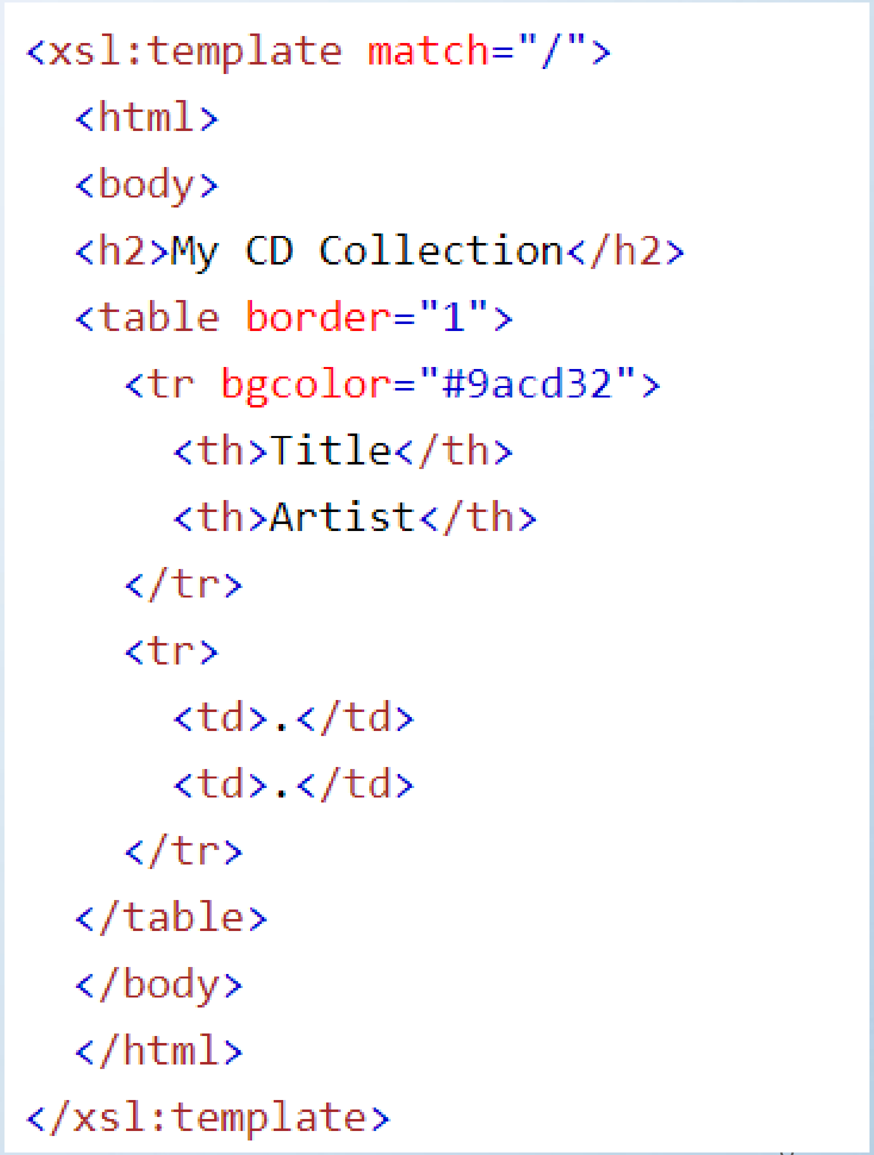
Link the XSL Style Sheet to the XML Document

- Add the XSL style sheet reference to your XML document ("cdcatalog.xml"):

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
.
.
</catalog>
```

XSLT <xsl:template> Element

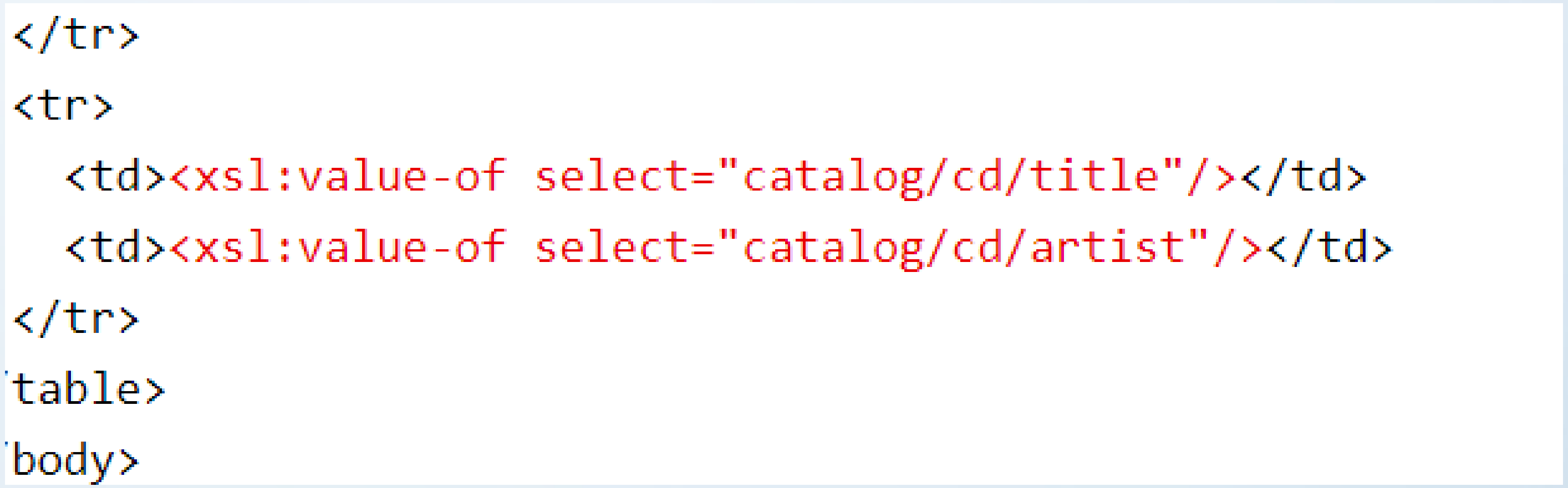
- An XSL style sheet consists of one or more set of rules that are called templates.
- A template contains rules to apply when a specified node is matched
- The <xsl:template> element is used to build templates.
- The match attribute is used to associate a template with an XML element. The match attribute can also be used to define a template for the entire XML document.
- The value of the match attribute is an XPath expression (i.e. match="/" defines the whole document)



```
<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <table border="1">
    <tr bgcolor="#9acd32">
      <th>Title</th>
      <th>Artist</th>
    </tr>
    <tr>
      <td>.</td>
      <td>.</td>
    </tr>
  </table>
  </body>
  </html>
</xsl:template>
```

XSLT <xsl:value-of> Element

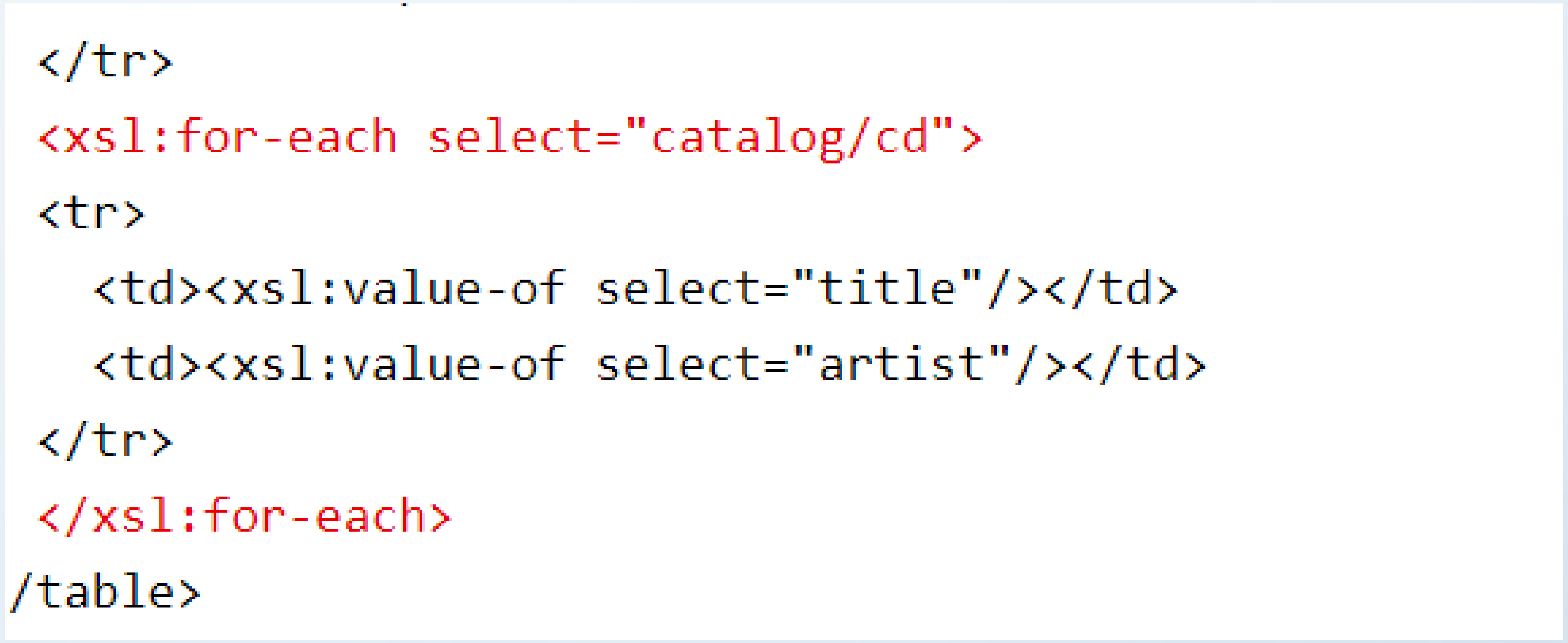
- The <xsl:value-of> element can be used to extract the value of an XML element and add it to the output stream of the transformation



```
</tr>
<tr>
  <td><xsl:value-of select="catalog/cd/title"/></td>
  <td><xsl:value-of select="catalog/cd/artist"/></td>
</tr>
table>
body>
```

XSLT <xsl:for-each> Element

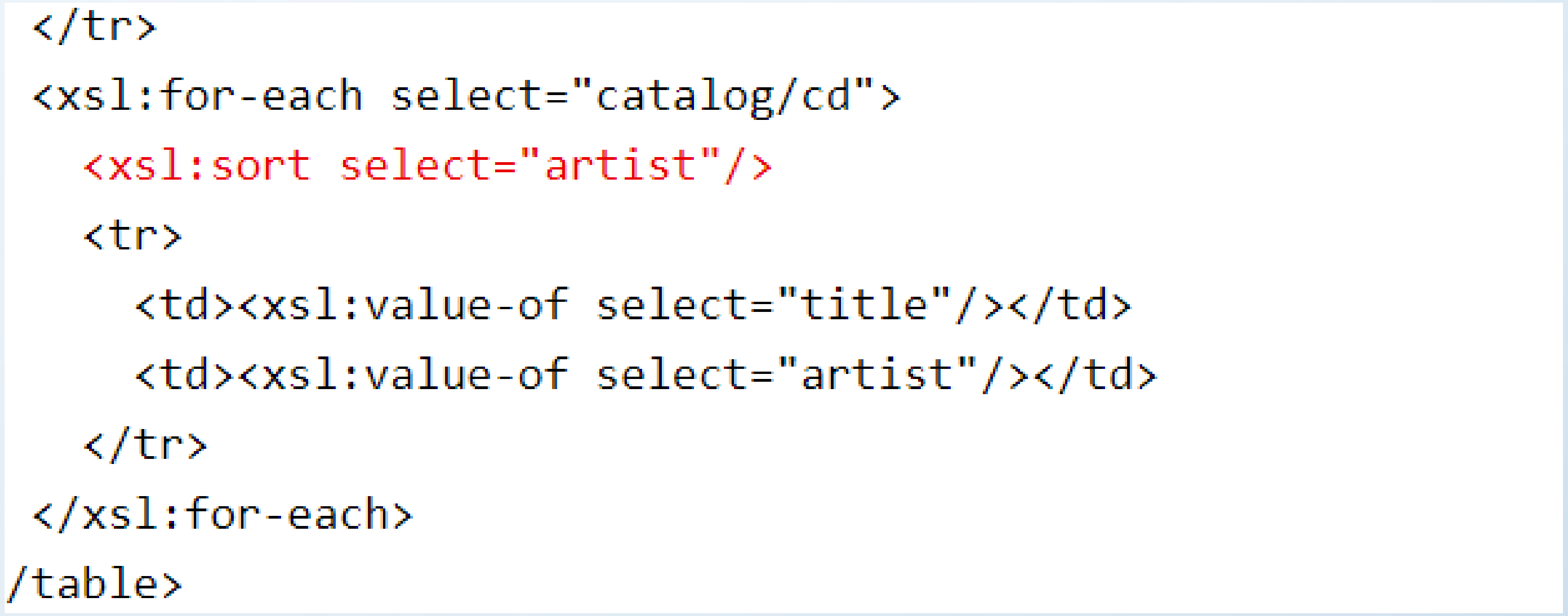
- The <xsl:for-each> element allows you to do looping in XSLT.



```
 </tr>
 <xsl:for-each select="catalog/cd">
 <tr>
   <td><xsl:value-of select="title"/></td>
   <td><xsl:value-of select="artist"/></td>
 </tr>
 </xsl:for-each>
/table>
```

XSLT <xsl:sort> Element

- To sort the output, simply add an <xsl:sort> element inside the <xsl:for-each> element in the XSL file



```
 </tr>
 <xsl:for-each select="catalog/cd">
   <xsl:sort select="artist"/>
   <tr>
     <td><xsl:value-of select="title"/></td>
     <td><xsl:value-of select="artist"/></td>
   </tr>
 </xsl:for-each>
/table>
```

XSLT <xsl:if> Element

- To put a conditional if test against the content of the XML file, add an <xsl:if> element to the XSL document.

```
<xsl:if test="expression">
  ...some output if the expression is true...
</xsl:if>
```

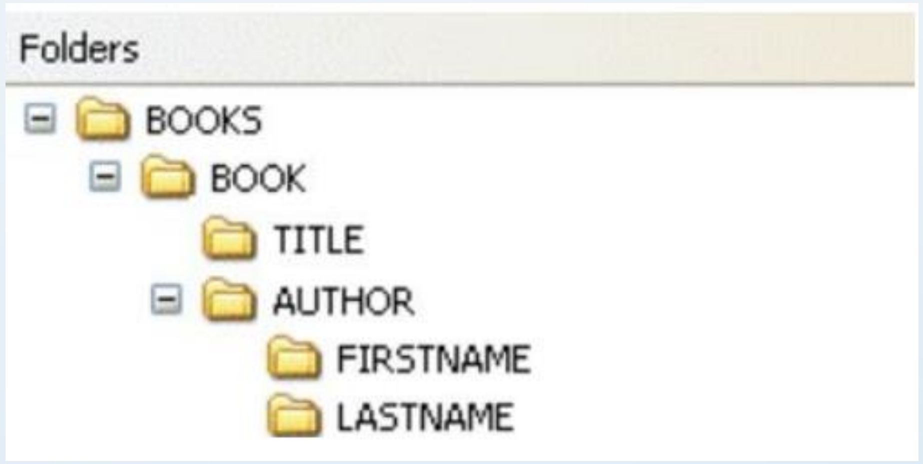
```
<xsl:for-each select="catalog/cd">
  <xsl:if test="price &gt; 10">
    <tr>
      <td><xsl:value-of select="title"/></td>
      <td><xsl:value-of select="artist"/></td>
      <td><xsl:value-of select="price"/></td>
    </tr>
  </xsl:if>
</xsl:for-each>
```

XPath

- XPath is a major element in the XSLT standard.
- XPath can be used to navigate through elements and attributes in an XML document.
- XPath stands for XML Path Language
- XPath uses "path like" syntax to identify and navigate nodes in an XML document
- XPath contains over 200 built-in functions
- XPath is a major element in the XSLT standard
- XPath is a W3C recommendation

XPath Path Expressions

- XPath uses path expressions to select nodes or node-sets in an XML document.
- These path expressions look very much like the path expressions you use with traditional computer file systems:



XPath Standard Functions

- XPath includes over 200 built-in functions.
- There are functions for string values, numeric values, booleans, date and time comparison, node manipulation, sequence manipulation, and much more.
- Today XPath expressions can also be used in JavaScript, Java, XML Schema, PHP, Python, C and C++, and lots of other languages

XPath is Used in XSLT

- XPath is a major element in the XSLT standard.
- With XPath knowledge you will be able to take great advantage of your XSLT knowledge

XPath Terminology

- Nodes
  - In XPath, there are seven kinds of nodes: element, attribute, text, namespace, processing-instruction, comment, and document nodes.
  - XML documents are treated as trees of nodes. The topmost element of the tree is called the root element.
  - Example of nodes in the XML document above: ===➔

```xml
<?xml version="1.0" encoding="UTF-8"?>

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

```
<bookstore> (root element node)

<author>J K. Rowling</author> (element node)

lang="en" (attribute node)
```

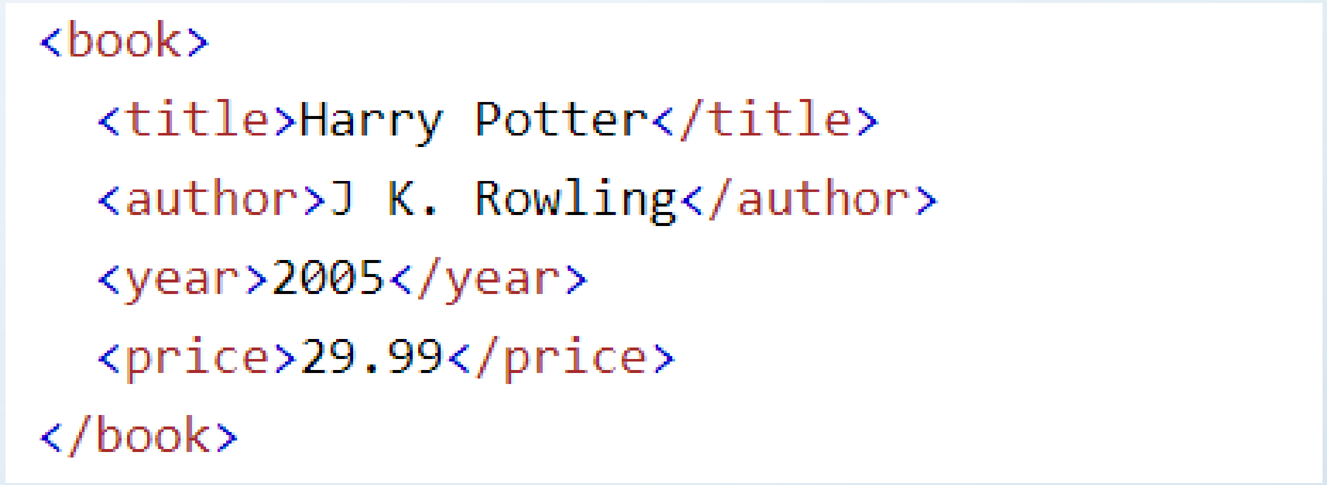
XPath Terminology

- Atomic values
  - Atomic values are nodes with no children or parent.
  - Example of atomic values:

```
J K. Rowling


"en"
```

Relationship of Nodes

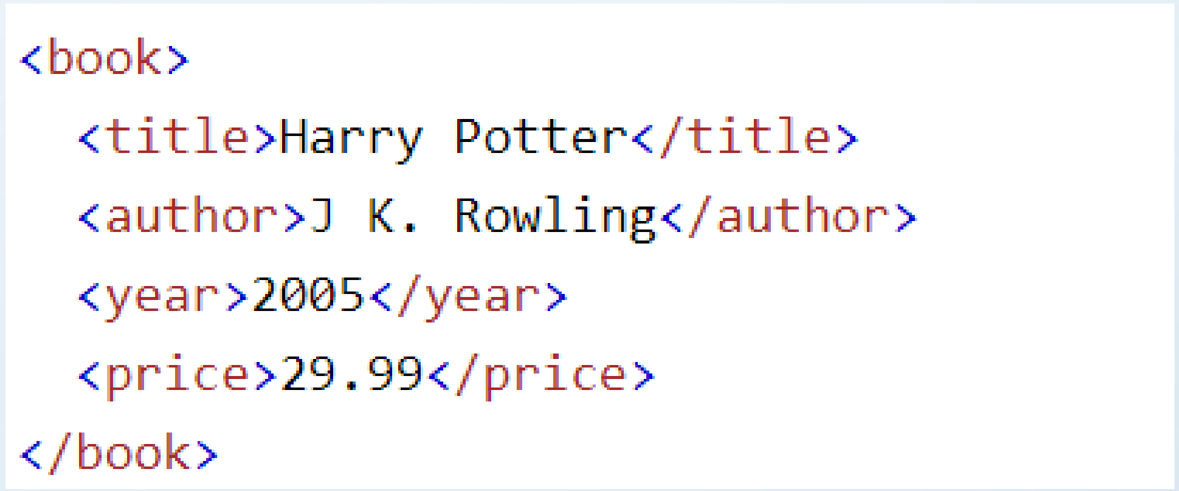
- Parent
  - Each element and attribute has one parent.
  - In the following example; the book element is the parent of the title, author, year, and price:



```xml
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

Relationship of Nodes

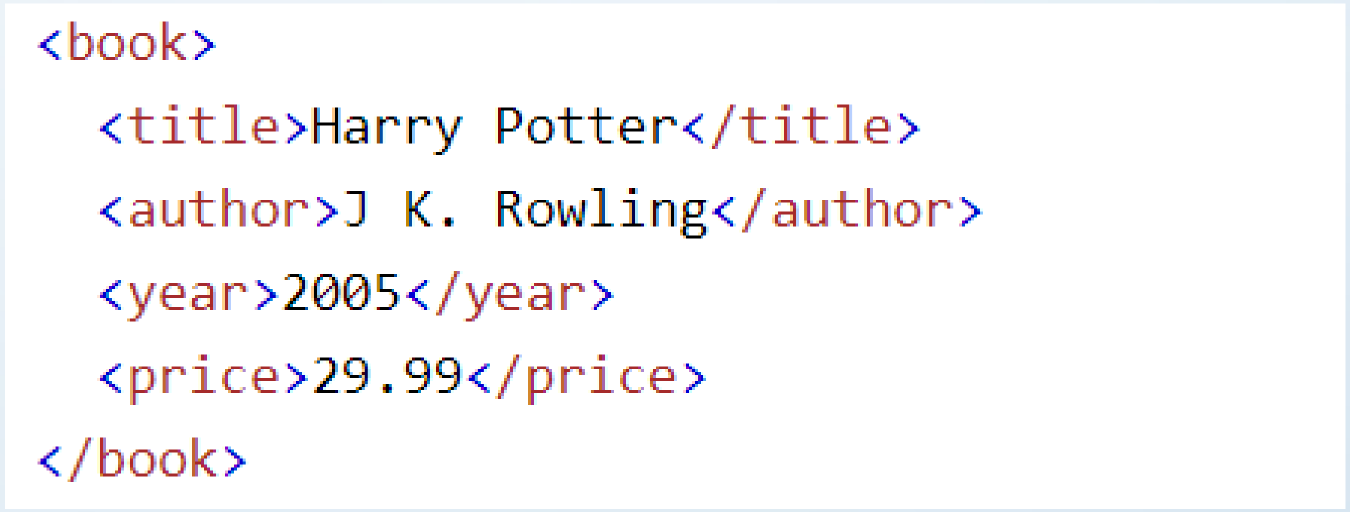
- Children
  - Element nodes may have zero, one or more children.
  - In the following example; the title, author, year, and price elements are all children of the book element:



```
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

Relationship of Nodes

- Siblings
  - Nodes that have the same parent.
  - In the following example; the title, author, year, and price elements are all siblings:



```xml
<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

Relationship of Nodes

- Ancestors
  - A node's parent, parent's parent, etc.
  - In the following example; the ancestors of the title element are the book element and the bookstore element:

```
<bookstore>


<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>


</bookstore>
```

Relationship of Nodes

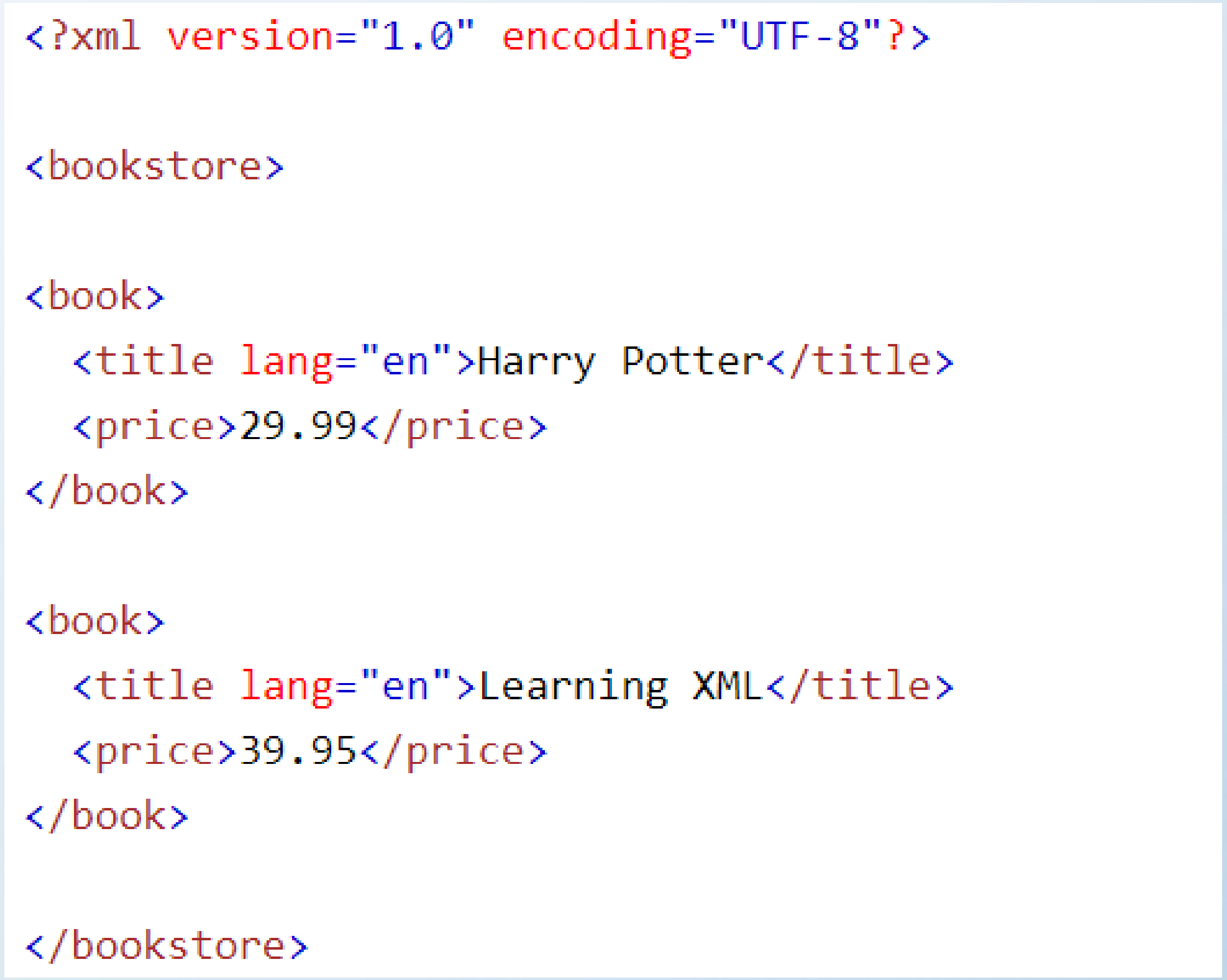
- Descendants
  - A node's children, children's children, etc.
  - In the following example; descendants of the bookstore element are the book, title, author, year, and price elements:

```
<bookstore>


<book>
  <title>Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>


</bookstore>
```

XPath Syntax

- XPath uses path expressions to select nodes or node-sets in an XML document. The node is selected by following a path or steps.

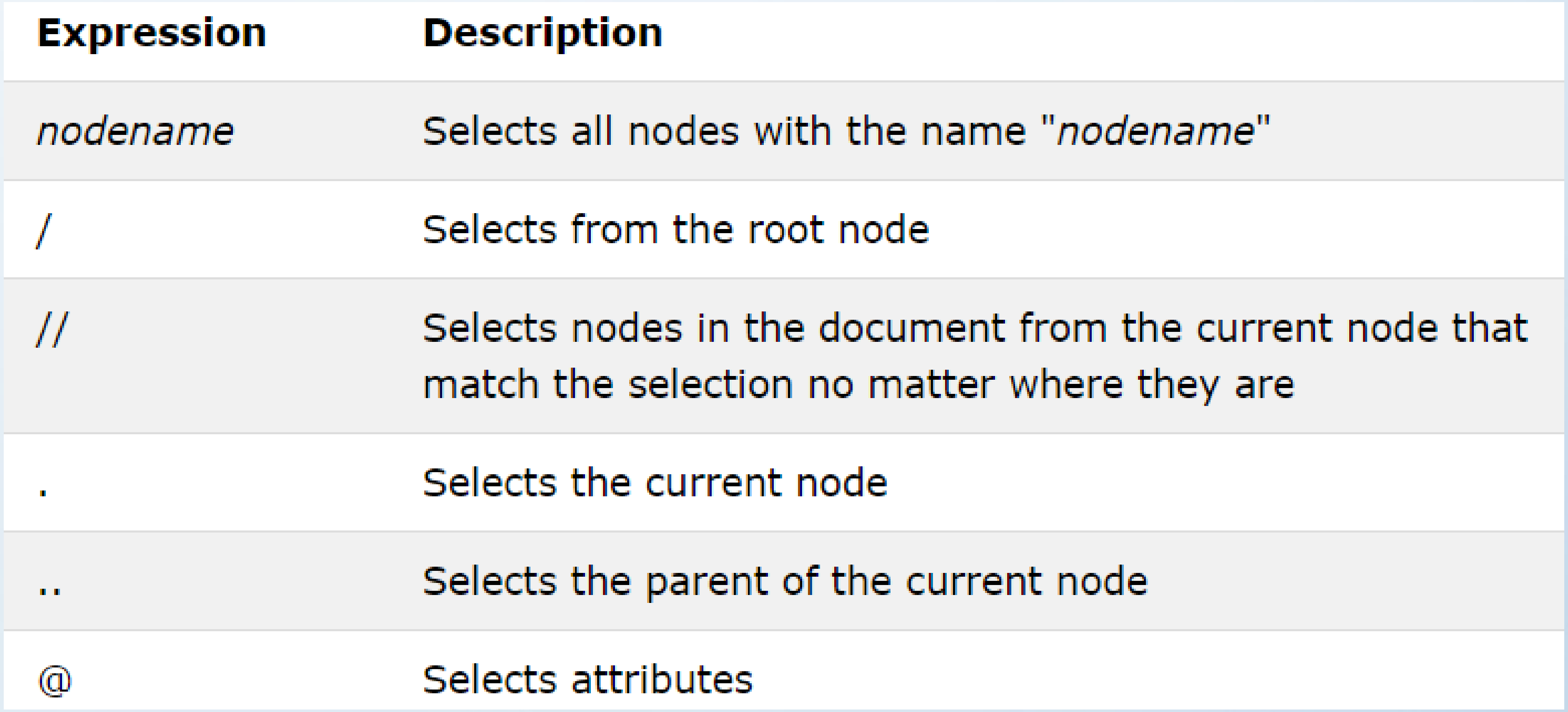
- XML File ===➔



```
<?xml version="1.0" encoding="UTF-8"?>

<bookstore>

<book>
  <title lang="en">Harry Potter</title>
  <price>29.99</price>
</book>

<book>
  <title lang="en">Learning XML</title>
  <price>39.95</price>
</book>

</bookstore>
```

Selecting Nodes

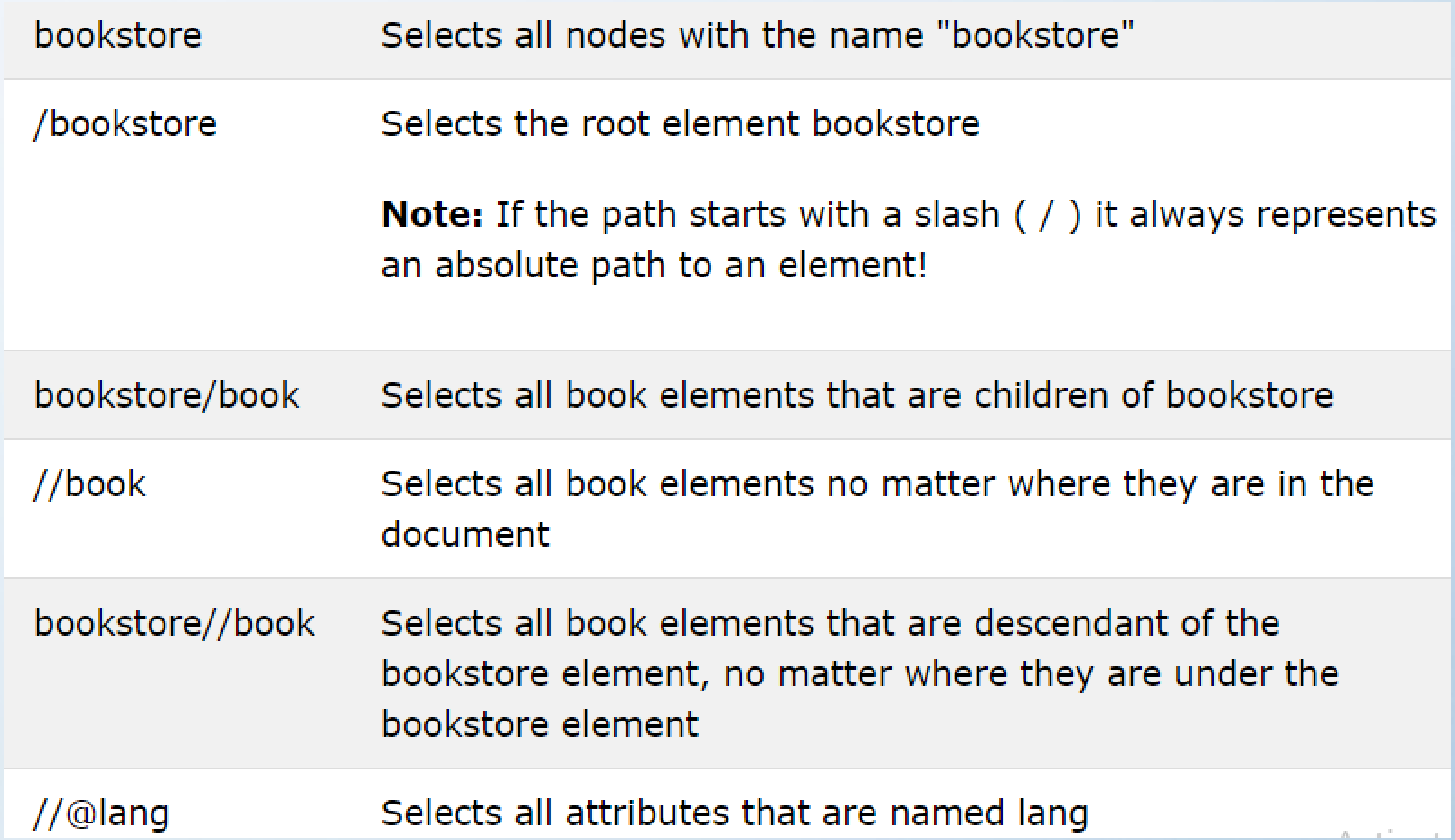
- XPath uses path expressions to select nodes in an XML document. The node is selected by following a path or steps.
- The most useful path expressions are listed below:



Expression	**Description**
nodename	Selects all nodes with the name "*nodename*"
/	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 Nodes

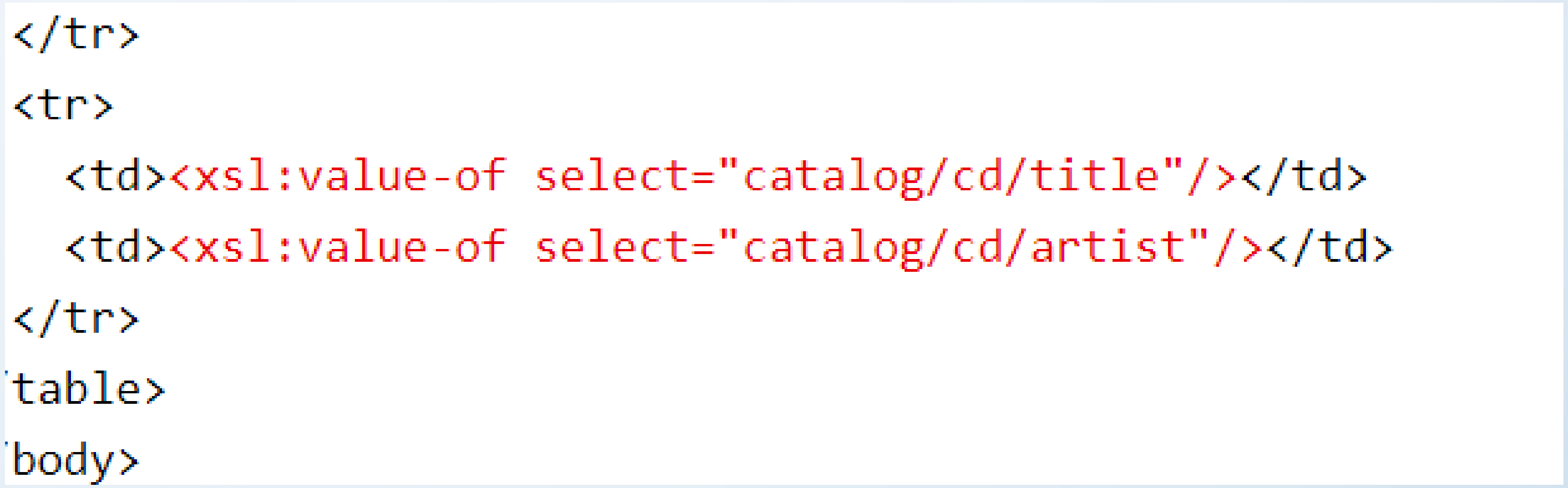
- In the table below, there are some path expressions and the result of the expressions:



bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element 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

XPath Example

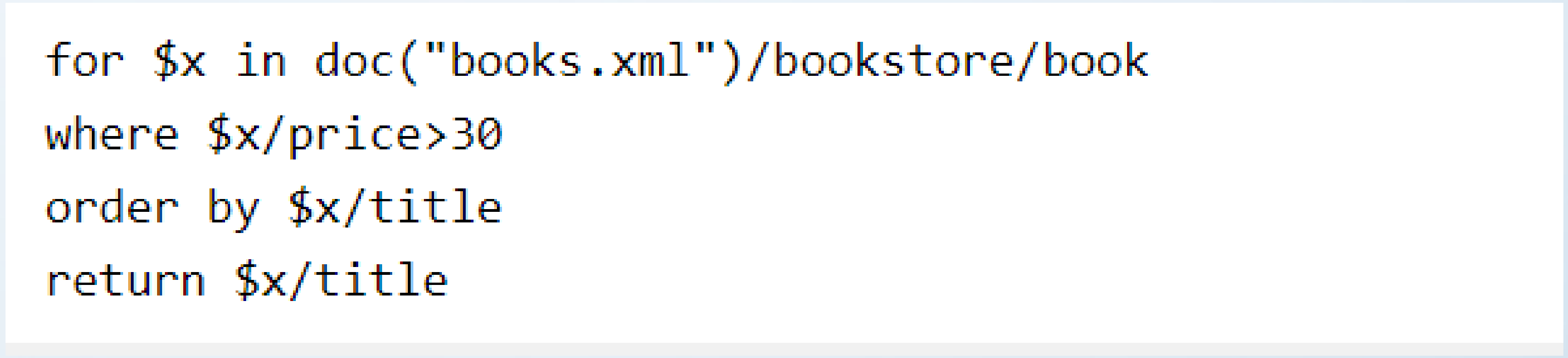
- Example from XSLT



```
</tr>
<tr>
  <td><xsl:value-of select="catalog/cd/title"/></td>
  <td><xsl:value-of select="catalog/cd/artist"/></td>
</tr>
'table>
'body>
```

XQuery

- XQuery is to XML what SQL is to databases.
- XQuery is designed to query XML data



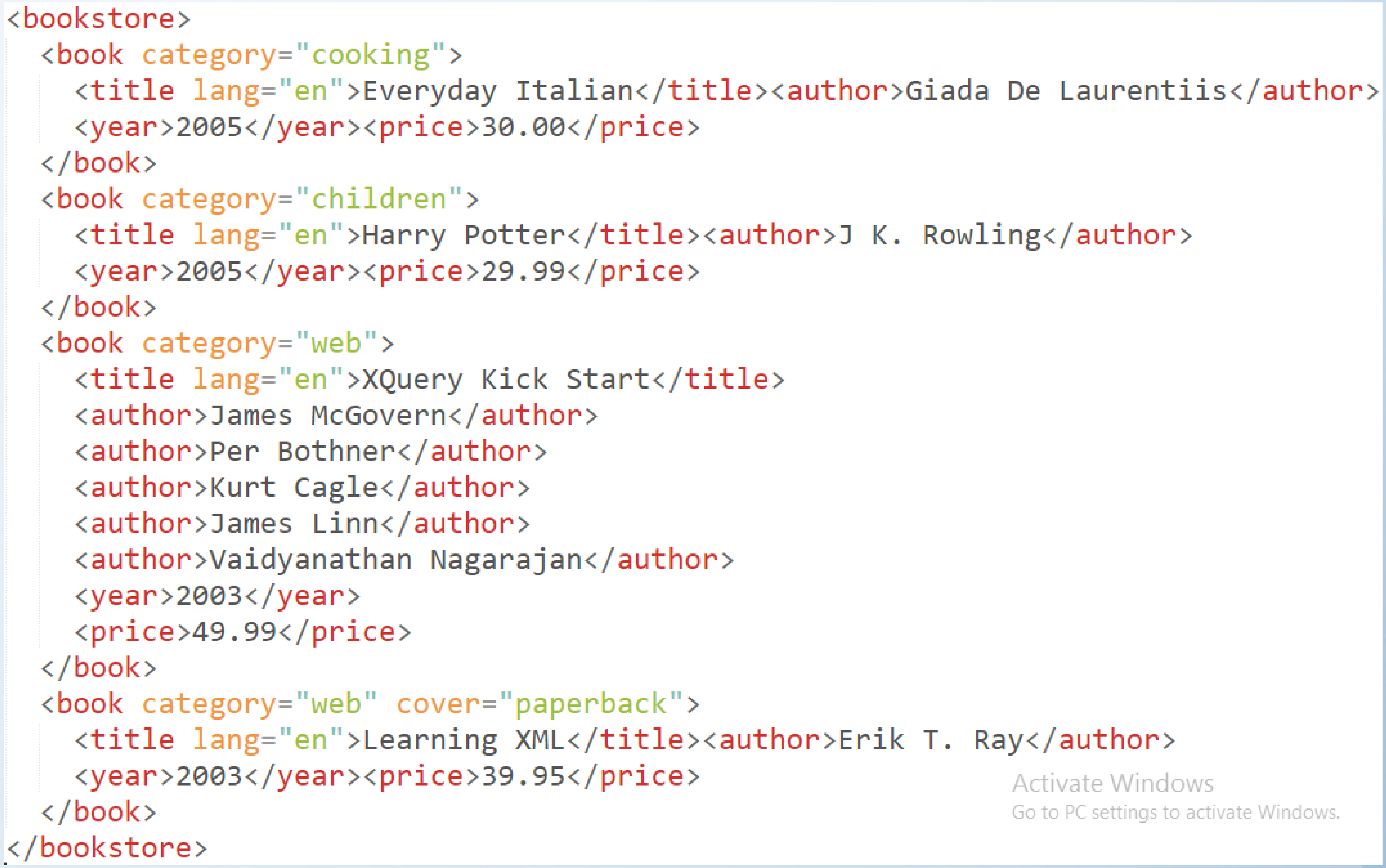
```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

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 major databases
- XQuery is a W3C Recommendation
- XQuery can be used to:
  - Extract information to use in a Web Service
  - Generate summary reports
  - Transform XML data to XHTML
  - Search Web documents for relevant information

XQuery Example

- We will use following XML file



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

XQuery Example

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

- /bookstore selects the bookstore element, /book selects all the book elements under the bookstore element, and /title selects all the title elements under each book element)
- The XQuery above will extract the following:

```
<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>
```

Predicates

- XQuery uses predicates to limit the extracted data from XML documents.
- The following predicate is used to select all the book elements under the bookstore element that have a price element with a value that is less than 30:

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

- The XQuery above will extract the following:

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

XQuery FLWOR Expressions

- **For** - selects a sequence of nodes
- **Let** - binds a sequence to a variable
- **Where** - filters the nodes
- **Order by** - sorts the nodes
- **Return** - what to return (gets evaluated once for every node)

XQuery FLWOR Expressions

- Look at the following path expression:

```
doc("books.xml")/bookstore/book[price>30]/title
```

- The expression above will select all the title elements under the book elements that are under the bookstore element that have a price element with a value that is higher than 30

- The following FLWOR expression will select exactly the same as the path expression above:

```
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>
```

XQuery FLWOR Expressions

- With FLWOR you can sort the result:

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

- Result :

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