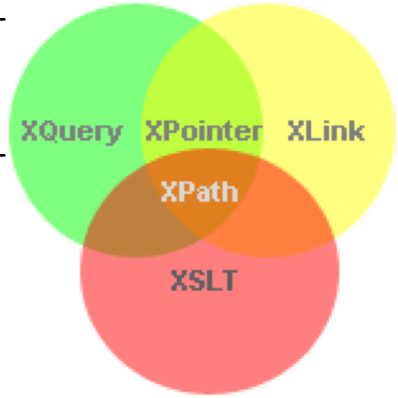
XPath

What is XPath ?



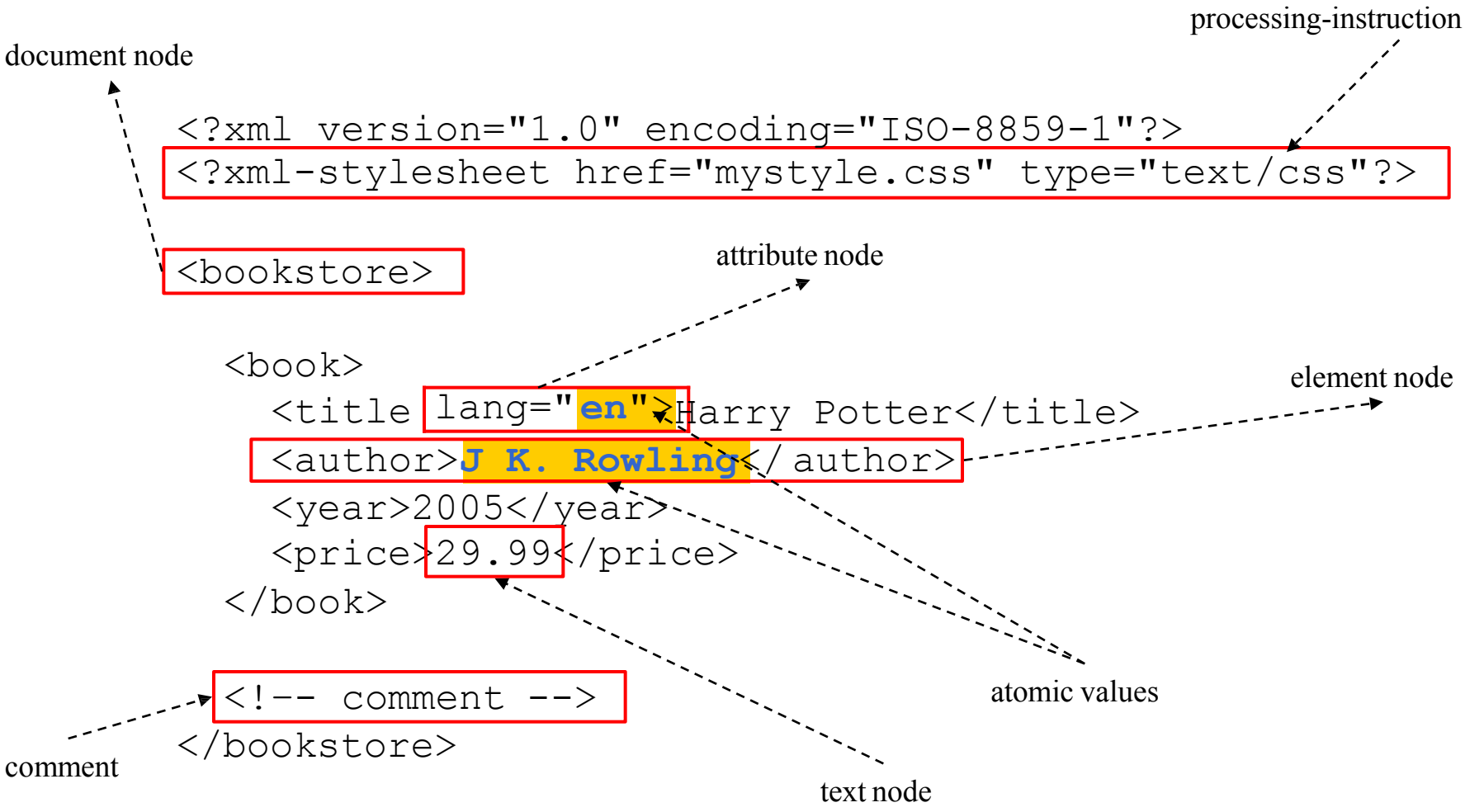
- XPath is an XML query language.

- XPath is a building block for other XML query languages, such as XSLT and XQuery.
- XPath addresses parts of an XML document by means of path expressions.
- A path expression in XPath is a sequence of location steps separated by “/”.
- The result of a path expression is a set of values.

XPath Terminology

- Nodes
  - element
  - attribute
  - text
  - namespace
  - processing-instruction
  - comment
  - document (root) nodes

Example

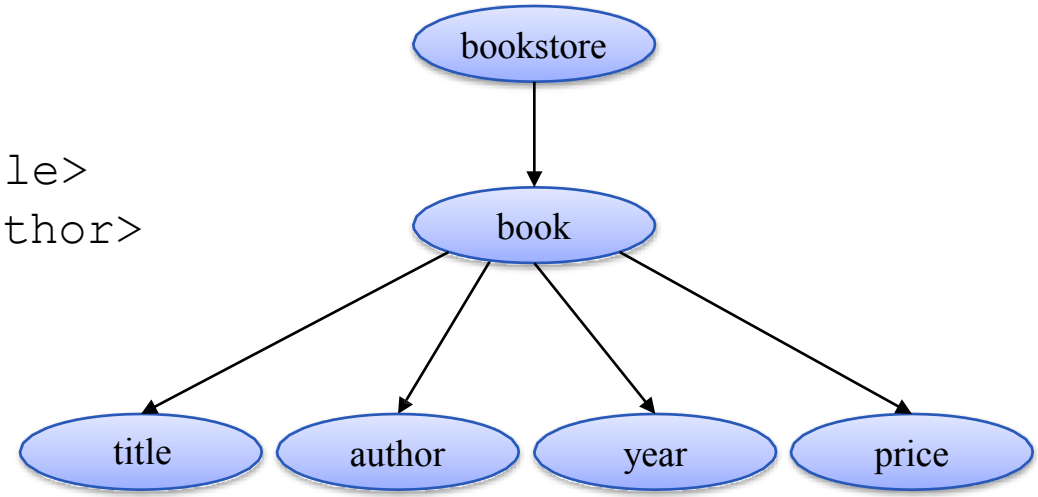

document node
processing-instruction
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="mystyle.css" type="text/css"?>
<bookstore>
attribute node
<book>
<title lang="en">Harry Potter</title>
element node
<author>J K. Rowling</author>
<year>2005</year>
<price>29.99</price>
</book>
<!-- comment -->
</bookstore>
atomic values
comment
text node

Relationship of Nodes

```
</bookstore>

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

</bookstore>
```



Parent | Child | Siblings | Ancestors | Descendants

Slashes

- A path that begins with a / represents an *absolute path*, starting from the top of the document
  - Example: /email/message/header/from
  - Note that even an absolute path can select *more than one* element
  - A slash by itself means "the whole document"
- A path that does *not* begin with a / represents a path starting from the current element
  - Example: header/from
- A path that begins with // can start from *anywhere* in the document
  - Example: //header/from selects every element from that is a child of an element header
  - This can be expensive, since it involves searching the entire document

Brackets and last()

- A number in brackets selects a particular matching child
  - Example: /library/book[1] selects the first book of the library
  - Example: //chapter/section[2] selects the second section of every chapter in the XML document
  - Example: //book/chapter[1]/section[2]
  - Only matching elements are counted; for example, if a book has both sections and exercises, the latter are ignored when counting sections
- The function `last()` in brackets selects the last matching child
  - Example: /library/book/chapter[last()]
- You can even do simple arithmetic
  - Example: /library/book/chapter[last()-1]

Stars

- A star, or asterisk, is a "wild card“ -- it means "all the elements at this level"
  - Example: /library/book/chapter/* selects every child of every chapter of every book in the library
  - Example: //book/* selects every child of every book (chapters, tableOfContents, index, etc.)
  - Example: /*/*/*/paragraph selects every paragraph that has exactly three ancestors
  - Example: //* selects every element in the entire document

Example 1

The basic XPath syntax is similar to filesystem addressing. If the path starts with the slash / , then it represents an absolute path to the required element.

/AAA	/AAA/CCC	/AAA/DDD/BBB
Select the root element AAA	Select all elements CCC which are children of the root element AAA	Select all elements BBB which are children of DDD which are children of the root element AAA

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <BBB/>
  <DDD>
     <BBB/>
  </DDD>
  <CCC/>
</AAA>
```

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <BBB/>
  <DDD>
     <BBB/>
  </DDD>
  <CCC/>
</AAA>
```

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <BBB/>
  <DDD>
     <BBB/>
  </DDD>
  <CCC/>
</AAA>
```

Example 2

If the path starts with // then all elements in the document which fulfill following criteria are selected.

//BBB
Select all elements BBB

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
    </DDD>
  </CCC>
</AAA>
```

//DDD/BBB
Select all elements BBB which are children of DDD

```
<AAA>
  <BBB/>
  <CCC/>
  <BBB/>
  <DDD>
    <BBB/>
  </DDD>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
    </DDD>
  </CCC>
</AAA>
```

Example 3

The star * selects all elements located by preceeding path

/AAA/CCC/DDD/*

Select all elements enclosed by elements /AAA/CCC/DDD

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

/*/*/*/BBB

Select all elements BBB which have 3 ancestors

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

//*

Select all elements

```
<AAA>
  <XXX>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </XXX>
  <CCC>
    <DDD>
      <BBB/>
      <BBB/>
      <EEE/>
      <FFF/>
    </DDD>
  </CCC>
  <CCC>
    <BBB>
      <BBB>
        <BBB/>
      </BBB>
    </BBB>
  </CCC>
</AAA>
```

Example 4

Expresion in square brackets can further specify an element. A number in the brackets gives the position of the element in the selected set. The function last() selects the last element in the selection.

/AAA/BBB[1]
Select the first BBB child of element AAA
<AAA>   <BBB/>   <BBB/>   <BBB/>   <BBB/> </AAA>

/AAA/BBB[last()]
Select the last BBB child of element AAA
<AAA>   <BBB/>   <BBB/>   <BBB/>   <BBB/> </AAA>

Example 5

Attributes are specified by @ prefix.

//@id

Select all attributes @id

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

//BBB[@id]

Select BBB elements which have attribute id

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

//BBB[@name]

Select BBB elements which have attribute name

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

//BBB[@*]

Select BBB elements which have any attribute

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

//BBB[not(@*)]

Select BBB elements without an attribute

```
<AAA>
  <BBB id = "b1"/>
  <BBB id = "b2"/>
  <BBB name = "bbb"/>
  <BBB/>
</AAA>
```

Example 6

Values of attributes can be used as selection criteria. Function normalize-space removes leading and trailing spaces and replaces sequences of whitespace characters by a single space.

//BBB[@id='b1']

Select BBB elements which have attribute id with value b1

```
<AAA>
  <BBB id = "b1"/>
  <BBB name = " bbb "/>
  <BBB name = "bbb"/>
</AAA>
```

//BBB[@name='bbb']

Select BBB elements which have attribute name with value 'bbb'

```
<AAA>
  <BBB id = "b1"/>
  <BBB name = " bbb "/>
  <BBB name = "bbb"/>
</AAA>
```

//BBB[normalize-space(@name)='bbb']

Select BBB elements which have attribute name with value bbb, leading and trailing spaces are removed before comparison

```
<AAA>
  <BBB id = "b1"/>
  <BBB name = " bbb "/>
  <BBB name = "bbb"/>
</AAA>
```

Example 7

Function count() counts the number of selected elements

//*[count(BBB)=2]	//*[count(*)=2]	//*[count(*)=3]
Select elements which have two children BBB	Select elements which have 2 children	Select elements which have 3 children
<AAA> <CCC> <BBB/> <BBB/> <BBB/> </CCC> <DDD> <BBB/> <BBB/> </DDD> <EEE> <CCC/> <DDD/> </EEE> </AAA>	<AAA> <CCC> <BBB/> <BBB/> <BBB/> </CCC> <DDD> <BBB/> <BBB/> </DDD> <EEE> <CCC/> <DDD/> </EEE> </AAA>	<AAA> <CCC> <BBB/> <BBB/> <BBB/> </CCC> <DDD> <BBB/> <BBB/> </DDD> <EEE> <CCC/> <DDD/> </EEE> </AAA>

Example 8

Function name() returns name of the element, the starts-with function returns true if the first argument string starts with the second argument string, and the contains function returns true if the first argument string contains the second argument string.

//*[name()='BBB']

Select all elements with name BBB, equivalent with //BBB

```
<AAA>
  <BCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </BCC>
  <DDB>
    <BBB/>
    <BBB/>
  </DDB>
  <BEC>
    <CCC/>
    <DBD/>
  </BEC>
</AAA>
```

//*[starts-with(name(),'B')]

Select all elements name of which starts with letter B

```
<AAA>
  <BCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </BCC>
  <DDB>
    <BBB/>
    <BBB/>
  </DDB>
  <BEC>
    <CCC/>
    <DBD/>
  </BEC>
</AAA>
```

//*[contains(name(),'C')]

Select all elements name of which contain letter C

```
<AAA>
  <BCC>
    <BBB/>
    <BBB/>
    <BBB/>
  </BCC>
  <DDB>
    <BBB/>
    <BBB/>
  </DDB>
  <BEC>
    <CCC/>
    <DBD/>
  </BEC>
</AAA>
```

Example 9

The string-length function returns the number of characters in the string. You must use < as a substitute for < and > as a substitute for >.

//*[string-length(name()) = 3]

Select elements with three-letter name

```
<AAA>
  <Q/>
  <SSSS/>
  <BB/>
  <CCC/>
  <DDDDDDDD/>
  <EEEE/>
</AAA>
```

//*[string-length(name()) < 3]

Select elements name of which has one or two characters

```
<AAA>
  <Q/>
  <SSSS/>
  <BB/>
  <CCC/>
  <DDDDDDDD/>
  <EEEE/>
</AAA>
```

//*[string-length(name()) > 3]

Select elements with name longer than three characters

```
<AAA>
  <Q/>
  <SSSS/>
  <BB/>
  <CCC/>
  <DDDDDDDD/>
  <EEEE/>
</AAA>
```

Example 10

Several paths can be combined with | separator.

//CCC | //BBB

Select all elements CCC and BBB

```
<AAA>
  <BBB/>
  <CCC/>
  <DDD>
    <CCC/>
  </DDD>
  <EEE/>
</AAA>
```

/AAA/EEE | //BBB

Select all elements BBB and elements EEE which are children of root element AAA

```
<AAA>
  <BBB/>
  <CCC/>
  <DDD>
    <CCC/>
  </DDD>
  <EEE/>
</AAA>
```

/AAA/EEE | //DDD/CCC | /AAA | //BBB

Number of combinations is not restricted

```
<AAA>
  <BBB/>
  <CCC/>
  <DDD>
    <CCC/>
  </DDD>
  <EEE/>
</AAA>
```

Example 11

The child axis contains the children of the context node. The child axis is the default axis and it can be omitted.

/AAA
Equivalent of /child::AAA
<AAA>   <BBB/>   <CCC/> </AAA>

/child::AAA
Equivalent of /AAA
<AAA>   <BBB/>   <CCC/> </AAA>

/AAA/BBB
Equivalent of /child::AAA/child::BBB
<AAA>   <BBB/>   <CCC/> </AAA>

/child::AAA/child::BBB
Equivalent of /AAA/BBB
<AAA>   <BBB/>   <CCC/> </AAA>

/child::AAA/BBB
Both possibilities can be combined
<AAA>   <BBB/>   <CCC/> </AAA>

Example 12

The descendant axis contains the descendants of the context node; a descendant is a child or a child of a child and so on; thus the descendant axis never contains attribute or namespace nodes

/descendant::*

Select all descendants of document root and therefore all elements

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

/AAA/BBB/descendant::*

Select all descendants of /AAA/BBB

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

//CCC/descendant::*

Select all elements which have CCC among its ancestors

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

//CCC/descendant::DDD

Select elements DDD which have CCC among its ancestors

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

Example 13

The parent axis contains the parent of the context node, if there is one.

//DDD/parent::*

Select all parents of DDD element

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

Example 14

The ancestor axis contains the ancestors of the context node; the ancestors of the context node consist of the parent of context node and the parent's parent and so on; thus, the ancestor axis will always include the root node, unless the context node is the root node.

/AAA/BBB/DDD/CCC/EEE/ancestor::*

Select all elements given in this absolute path

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

//FFF/ancestor::*

Select ancestors of FFF element

```
<AAA>
  <BBB>
    <DDD>
      <CCC>
        <DDD/>
        <EEE/>
      </CCC>
    </DDD>
  </BBB>
  <CCC>
    <DDD>
      <EEE>
        <DDD>
          <FFF/>
        </DDD>
      </EEE>
    </DDD>
  </CCC>
</AAA>
```

Example 15

The following-sibling axis contains all the following siblings of the context node.

/AAA/BBB/following-sibling::*

```
<AAA>
  <BBB>
    <CCC/>
    <DDD/>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

//CCC/following-sibling::*

```
<AAA>
  <BBB>
    <CCC/>
    <DDD/>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

Example 16

The preceding-sibling axis contains all the preceding siblings of the context node.

/AAA/XXX/preceding-sibling::*

```
<AAA>
  <BBB>
    <CCC/>
    <DDD/>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

//CCC/preceding-sibling::*

```
<AAA>
  <BBB>
    <CCC/>
    <DDD/>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

Example 17

The following axis contains all nodes in the same document as the context node that are after the context node in document order, excluding any descendants and excluding attribute nodes and namespace nodes.

/AAA/XXX/following::*

```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
      <DDD>
        <EEE/>
      </DDD>
    </ZZZ>
    <FFF>
      <GGG/>
    </FFF>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

//ZZZ/following::*

```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
      <DDD>
        <EEE/>
      </DDD>
    </ZZZ>
    <FFF>
      <GGG/>
    </FFF>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

Example 18

The preceding axis contains all nodes in the same document as the context node that are before the context node in document order, excluding any ancestors and excluding attribute nodes and namespace nodes

/AAA/XXX/preceding::*

```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
    </ZZZ>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

//GGG/preceding::*

```
<AAA>
  <BBB>
    <CCC/>
    <ZZZ>
      <DDD/>
    </ZZZ>
  </BBB>
  <XXX>
    <DDD>
      <EEE/>
      <DDD/>
      <CCC/>
      <FFF/>
      <FFF>
        <GGG/>
      </FFF>
    </DDD>
  </XXX>
  <CCC>
    <DDD/>
  </CCC>
</AAA>
```

What is XPointer?

an extension of XPath suited for linking

specifies connection between XPath expressions and URIs

Related to, but much more powerful than location specification of HTML.

Example of HTML location Ref

<A NAME='3.2.4'> or <Image name = "3.2.4">

<A HREF='#3.2.4'> Reference to locationTag

Problems:

- An anchor must be placed at every link
- The link definition must be at the same location as the link source
- Only individual nodes can be linked to

Xpointer examples

The Tag # Introduction locates an ID.

```
<elem id= “Introduction”>
        …….
<elem>
```

* #xpointer(id("foo"))
* xpointer(/chapter[3]/elem[@name="foo"])

```
Match:   <chapter>
             <elem name=“foo”> …<elem>
         </chapter>
```

Absolute location

Root() root of the document tree.
Origin() Where the traversal started.
Id() You just saw this.
Html() existing compatibility.

Relative Location (from Xpath)

Child(): 1 level down.
descendant() depth first down.
ancestor()
following() appears after
psibling() Previous sibling.
fsibling()   following.

Syntax

child(2, section).(1,subsection) is the same as:

child(2, section).child(1, subsection)

If key word is omitted, it is treated as equivalent to the immediately preceding keyword.

First keyword must not be omitted.

Child(-1, section)

Returns the last section tag of the xml document.

Attributes

#pi (processing instruction)
#comment(XML comment)
#text (CDATA) text region inside elem & CDATA
#cdata (Same as above)
#all (same as #element)

Another example

```
<!DOCTYPE SPEECH [
<!ELEMENT SPEECH (#PCDATA|SPEAKER|DIRECTION)*>
<!ATTLIST SPEECH
ID ID #IMPLIED>
<!ELEMENT SPEAKER (#PCDATA)>
<!ELEMENT DIRECTION (#PCDATA)>
]>
```

```
<SPEECH ID="a27"><SPEAKER>Polonius</SPEAKER>
<DIRECTION>crossing downstage</DIRECTION>Fare you well,
my lord. <DIRECTION>To Ros.</DIRECTION>
You go to seek Lord Hamlet? There he is.</SPEECH>
```

<SPEECH ID="a27">

<SPEAKER>Polonius</SPEAKER>

<DIRECTION>crossing downstage</DIRECTION>

Fare you well, my lord.

<DIRECTION>To Ros.</DIRECTION>

You go to seek Lord Hamlet? There he is.

</SPEECH>

id(a27).child(2,DIRECTION)

Selects the second "DIRECTION" element (whose content is " To Ros.").

id(a27).child(2,#element)

Selects the second child element (that is, the first direction, whose content is "crossing downstage").

id(a27).child(2,#text)

Selects the second text region , "Fare you well, my lord." (The line break between the SPEAKER and DIRECTION elements is the first text region.)

child(1,#element,TARGET,*)

child(1,#element,N,2).(1,#element,N,1)

child(1,FS,RESP,#IMPLIED)
first child, FS element, RESP attribute that is unspecified

html(Sec3.2)

root().descendant(1,A,NAME,"Sec3.2")

Spanning Term

The span keyword locates a sub-resource starting at the beginning of the data selected by its first argument and continuing through to the end of the data selected by its second argument.

Example: id(a23).span(child(1),child(3))

Attribute-match Term

The attr keyword takes only an attribute name as a selector and returns the attribute's value.

Example: id(a23).attr(N)

String Location Terms

Selects one or more strings or positions between strings in the location source.

InstanceOrAll SkipLit Position Length

```
StringTerm ::= 'string(' InstanceOrAll ',
' SkipLit (',' Position (',' Length')?)?)'
```

Example: root().string(3,"Thomas Pynchon",8)