Student Online Teaching Advice Notice

The materials and content presented within this session are intended solely for use in a context of teaching and learning at Trinity.

Any session recorded for subsequent review is made available solely for the purpose of enhancing student learning.

Students should not edit or modify the recording in any way, nor disseminate it for use outside of a context of teaching and learning at Trinity.

Please be mindful of your physical environment and conscious of what may be captured by the device camera and microphone during videoconferencing calls.

Recorded materials will be handled in compliance with Trinity's statutory duties under the Universities Act, 1997 and in accordance with the University's policies and procedures.

Further information on data protection and best practice when using videoconferencing software is available at https://www.tcd.ie/info_compliance/data-protection/.

CSU22041: Information Management I

Xpath

… an **art of making information accessible**.

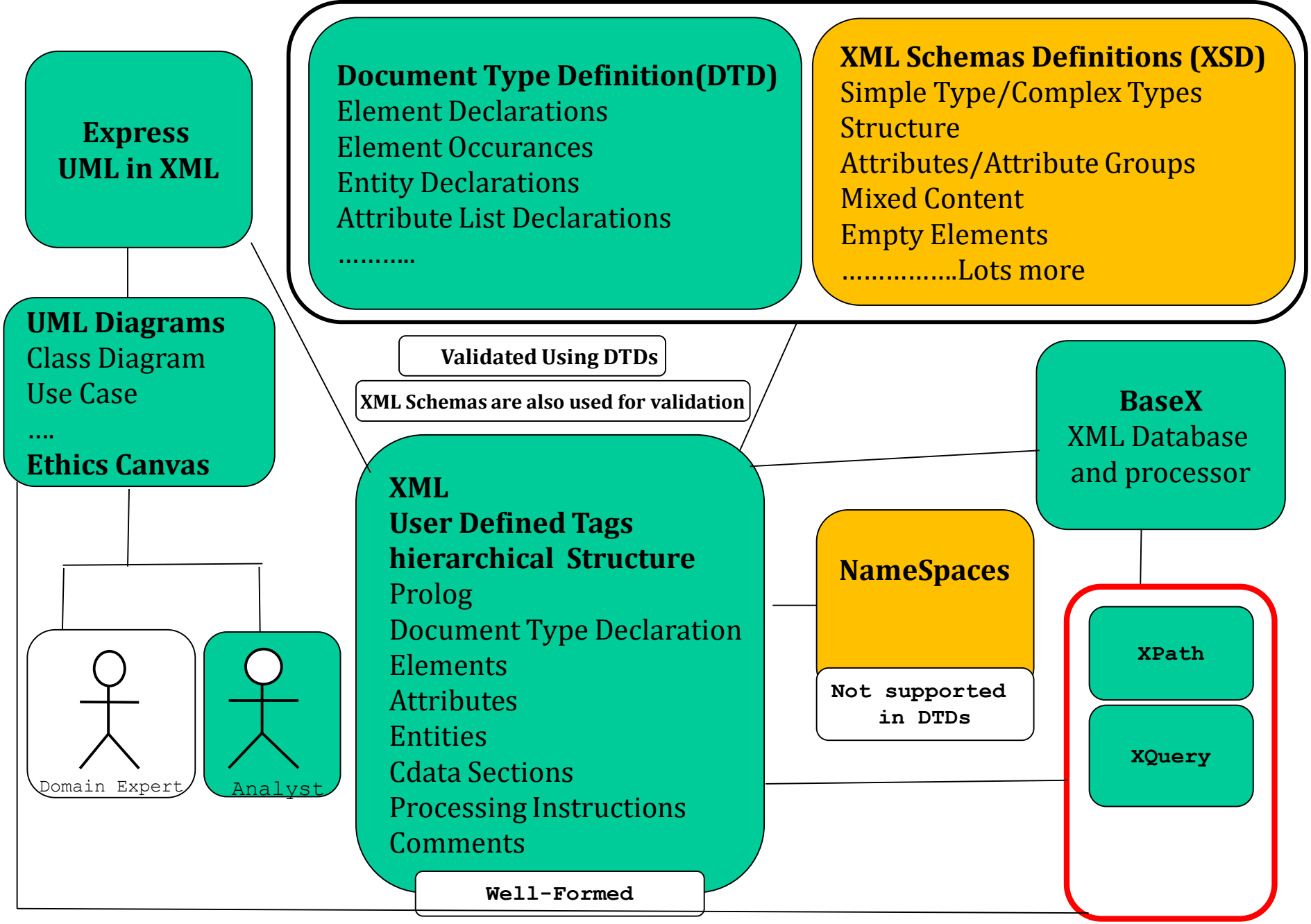
2020-2021

Gaye Stephens gaye.stephens@tcd.ie

I am expecting that …..

- You know what an XML document looks like
- You know what a DTD document looks like
- You have downloaded the BaseX software
- You have validated some XML documents using https://www.xmlvalidation.com
- You are working on the XML part of your project.

- In this video we will….
- Look at an overview of the process you are engaged in for your assignment.
- Introduce Xpath for navigating around XML documents


Express
UML in XML
Document Type Definition(DTD)
Element Declarations
Element Occurances
Entity Declarations
Attribute List Declarations
……..
XML Schemas Definitions (XSD)
Simple Type/Complex Types
Structure
Attributes/Attribute Groups
Mixed Content
Empty Elements
……..Lots more
UML Diagrams
Class Diagram
Use Case
….
Ethics Canvas
Validated Using DTDs
XML Schemas are also used for validation
BaseX
XML Database
and processor
XML
User Defined Tags
hierarchical Structure
Prolog
Document Type Declaration
Elements
Attributes
Entities
Cdata Sections
Processing Instructions
Comments
NameSpaces
Not supported
in DTDs
XPath
XQuery
Domain Expert
Analyst
Well-Formed

Navigating XML documents

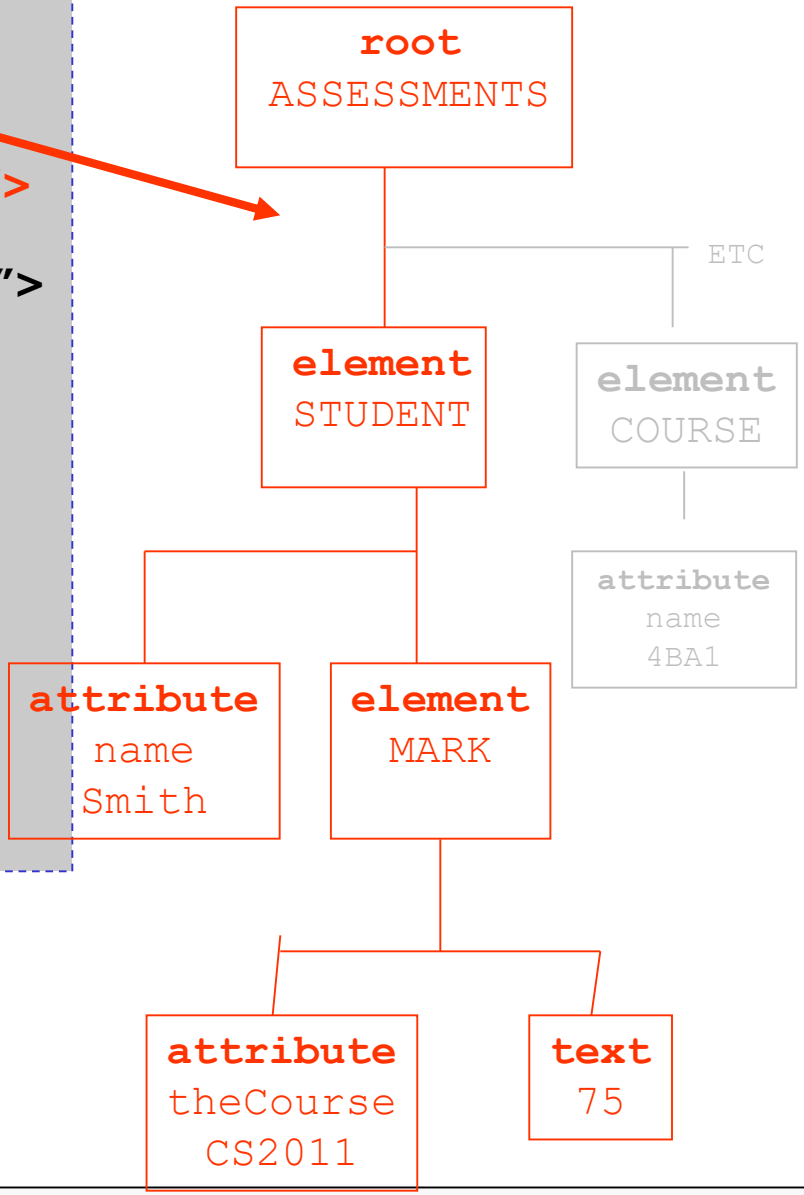
XML as a Tree structure

X Path for navigating the tree

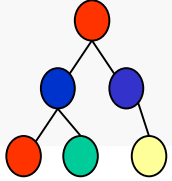
Xpath is used in XQueries

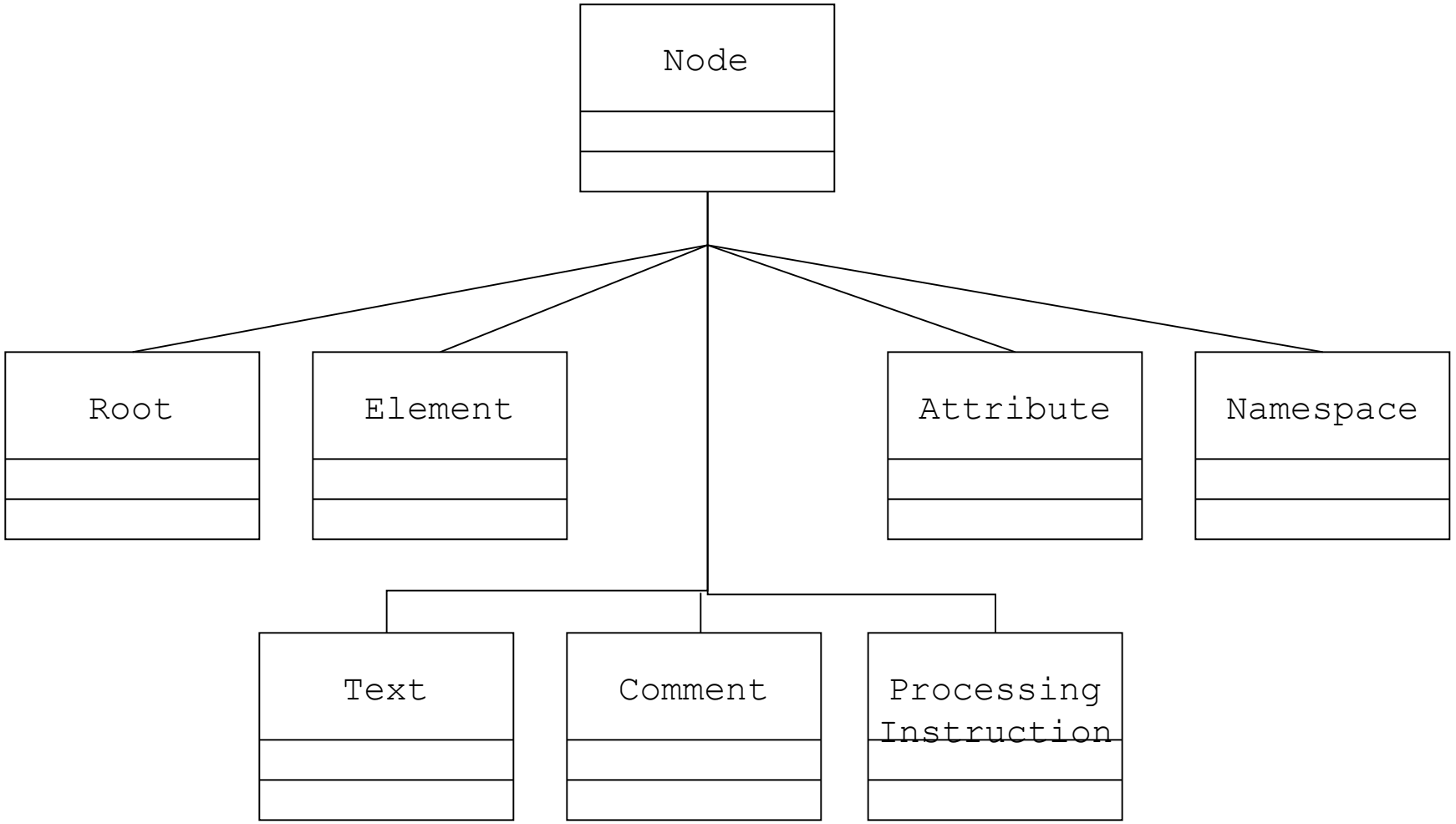
XML as a tree structure

```
<ASSESSMENTS>
        <STUDENT name = "Smith">
            <MARK theCourse = "CS2011">
75 </MARK>
            <MARK theCourse = "CS2012">
99 </MARK>
        </STUDENT> …


        <COURSE name = "CS2011",
takenBy = "Smith, Jones, … ">
         <MARK> 60 </MARK>
        </COURSE> …
</ASSESSMENTS>
```



Nodes in a Tree Model





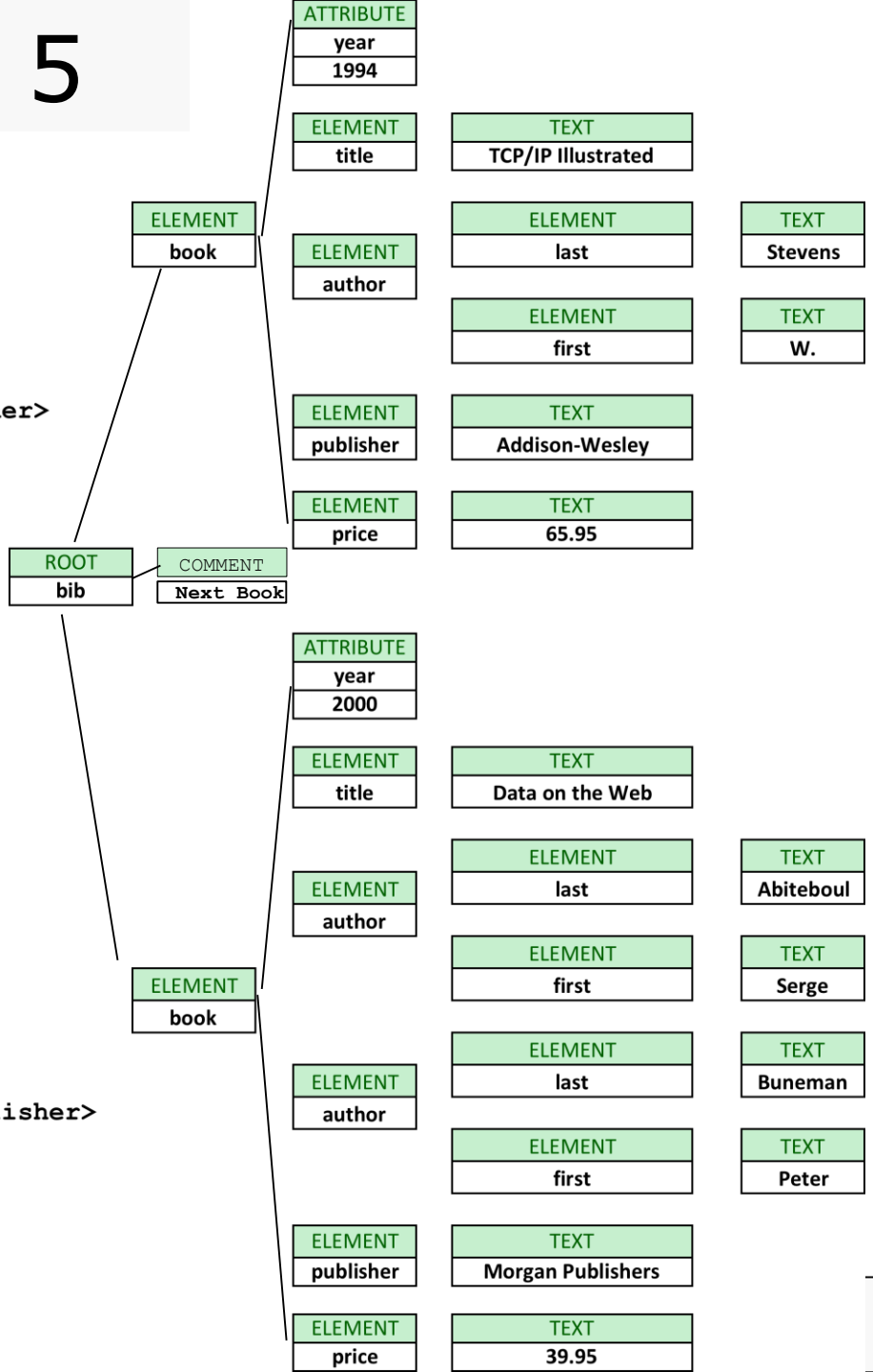
Exercise 5

- Create a XML Tree representation for the snippet of XML

```
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
    <author>
      <last>Stevens</last>
      <first>W.</first>
    </author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
</book>
<!-- Next Book --!>
  <book year="2000">
    <title>Data on the Web</title>
    <author>
      <last>Abiteboul</last>
      <first>Serge</first>
    </author>
    <author>
      <last>Buneman</last>
      <first>Peter</first>
    </author>
<publisher>Morgan Publishers</publisher>
    <price>39.95</price>
  </book>

</bib>
```

Solution Exercise 5

```
<bib>
  <book year="1994">
    <title>TCP/IP Illustrated</title>
    <author>
      <last>Stevens</last>
      <first>W.</first>
    </author>
    <publisher>Addison-Wesley</publisher>
    <price>65.95</price>
  </book>



   <!-- Next Book --!>



  <book year="2000">
    <title>Data on the Web</title>
    <author>
      <last>Abiteboul</last>
      <first>Serge</first>
    </author>
    <author>
      <last>Buneman</last>
      <first>Peter</first>
    </author>
    <publisher>Morgan Publishers</publisher>
    <price>39.95</price>
  </book>

</bib>
```



XPath Expression

- "Xpath, essentially specification of path for walking the XML tree"

- Simple path expression is a sequence of *steps* to walk the tree. The sequence of steps are separated by slashes (/)

Example X Path expression: /ASSESSMENTS

```
<ASSESSMENTS>
        <STUDENT name = "Smith">
                <MARK theCourse = "4BA1"> 75 </MARK>
                <MARK theCourse = "4BA5"> 99 </MARK>
        </STUDENT> …

        <COURSE name = "4BA1", takenBy = "Smith, Jones, … ">
        </COURSE> …
</ASSESSMENTS>
```

Example: /ASSESSMENTS/STUDENT

```
<ASSESSMENTS>
        <STUDENT name = "Smith">
                <MARK theCourse = "4BA1"> 75 </MARK>
                <MARK theCourse = "4BA5"> 99 </MARK>
        </STUDENT> …

        <COURSE name = "4BA1", takenBy = "Smith, Jones, … ">
        </COURSE> …
</ASSESSMENTS>
```

Example: /ASSESSMENTS/STUDENT/MARK

```
<ASSESSMENTS>
        <STUDENT name = "Smith">
                <MARK theCourse = "4BA1"> 75 </MARK>
                <MARK theCourse = "4BA5"> 99 </MARK>
        </STUDENT> ...

        <COURSE name = "4BA1", takenBy = "Smith, Jones, ... ">
        </COURSE> ...
</ASSESSMENTS>
```

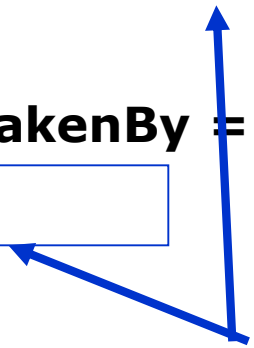
Describes the set with these two MARK element nodes as well as any other MARK elements nodes for any other STUDENT

If Xpath expression begins with //

- Selects nodes in the document from the current node that match the selection no matter where they are i.e. trying to match any descendent nodes in the set of nodes

Example: //MARK

```
<ASSESSMENTS>
   <STUDENT name = "Smith">
        <MARK theCourse = "4BA1"> 75 </MARK>
        <MARK theCourse = "4BA5"> 99 </MARK>
   </STUDENT> ...

   <COURSE name = "4BA1", takenBy = "Smith, Jones, ... ">
        <MARK> 60 </MARK>
   </COURSE> ...
</ASSESSMENTS>
```



Still returns set of nodes from the document with an element node named "MARK" but this time not just those noted in student assessment statements e.g. a mark allocated to a course by an external examiner

Example: //MARK/string()

```
<ASSESSMENTS>
        <STUDENT name = "Smith">
                <MARK theCourse = "4BA1"> 75 </MARK>
                <MARK theCourse = "4BA5"> 99 </MARK>
        </STUDENT> …

        <COURSE name = "4BA1", takenBy = "Smith, Jones,
… ">
                <MARK> 60 </MARK>
        </COURSE> …
</ASSESSMENTS>
```

Getting just the text from any "mark" elements
Using the string() function

Attribute @

- Attributes are referred to by putting an “at” symbol (@) before the name
- Appear in the path as if nested within the tag

Example:
/ASSESSMENTS/STUDENT/string(@name)

```
<ASSESSMENTS>
        <STUDENT name = "Smith">
                <MARK theCourse = "4BA1"> 75 </MARK>
                <MARK theCourse = "4BA5"> 99 </MARK>
        </STUDENT> ...

        <COURSE name = "4BA1", takenBy = "Smith, Jones,
... ">
                 <MARK> 60 </MARK>
        </COURSE> ...
</ASSESSMENTS>
```

Getting at an attribute value, string() function

Predicate Filters []

- A part of the path that allows for expression of a condition.
- [..] will ensure that only nodes that satisfy the condition are included in the resultant set

Example:

/ASSESSMENTS/STUDENT[MARK > 80]

```
<ASSESSMENTS>
        <STUDENT name = "Smith">
                <MARK theCourse = "4BA1"> 75 </MARK>
                <MARK theCourse = "4BA5"> 99 </MARK>
        </STUDENT> …

        <COURSE name = "4BA1", takenBy = "Smith, Jones,
… ">
                 <MARK> 60 </MARK>
        </COURSE> …
</ASSESSMENTS>
```

Example:

/ASSESSMENTS/STUDENT[MARK > 80]

```
<ASSESSMENTS>
        <STUDENT name = "Smith">
                <MARK theCourse = "4BA1"> 75 </MARK>
                <MARK theCourse = "4BA5"> 99 </MARK>
        </STUDENT> ...

        <COURSE name = "4BA1", takenBy = "Smith, Jones,
... ">
                <MARK> 60 </MARK>
        </COURSE> ...
</ASSESSMENTS>
```

This set of nodes is returned as it satisfies the condition

Example Attribute in the filter:

/ASSESSMENTS/STUDENT/MARK[@theCourse = "4BA1"]

```
<ASSESSMENTS>
        <STUDENT name = "Smith">
                <MARK theCourse = "4BA1"> 75 </MARK>
                <MARK theCourse = "4BA5"> 99 </MARK>
        </STUDENT> ...

        <COURSE name = "4BA1", takenBy = "Smith, Jones,
... ">
                <MARK> 60 </MARK>
        </COURSE> ...
</ASSESSMENTS>
```

This set of nodes is returned as well as any other student MARK subtree nodes for 4BA1 elsewhere

Wildcard *

- An asterix (*) Can be used as a wildcard
- Example /*/*/MARK will return any MARK Element appearing at the third level of nesting in the document

Consider what part of the tree (set of nodes) the following Xpath expressions will return

```
<database>
<person age='34'>
    <name>
        <title> Mr </title>
        <firstname> John </firstname>
        <firstname> Paul </firstname>
        <surname> Murphy </surname>
    </name>
    <hobby> Football </hobby>
    <hobby> Racing </hobby>
</person>

<person >
    <name>
        <firstname> Mary </firstname>
        <surname> Donnelly </surname>
    </name>
</person>
</database>
```

1. /database
2. //surname
3. /*/person/@age
4. /*/person/string(@age)

```
<database>
<person age='34'>
    <name>
        <title> Mr </title>
        <firstname> John </firstname>
        <firstname> Paul </firstname>
        <surname> Murphy </surname>
    </name>
    <hobby> Football </hobby>
    <hobby> Racing </hobby>
</person>

<person >
    <name>
        <firstname> Mary </firstname>
        <surname> Donnelly </surname>
    </name>
</person>
</database>
```

1. /database
2. //surname
3. /*/person[@age]
4. /*/person/string(@age)

```
<database>
<person age='34'>
    <name>
        <title> Mr </title>
        <firstname> John </firstname>
        <firstname> Paul </firstname>
        <surname> Murphy </surname>
    </name>
    <hobby> Football </hobby>
    <hobby> Racing </hobby>
</person>

<person >
    <name>
        <firstname> Mary </firstname>
        <surname> Donnelly </surname>
    </name>
</person>
</database>
```

1. /database
2. //surname
3. /*/person[@age]
4. /*/person/string(@age)

```
<database>
<person age='34'>
    <name>
        <title> Mr </title>
        <firstname> John </firstname>
        <firstname> Paul </firstname>
        <surname> Murphy </surname>
    </name>
    <hobby> Football </hobby>
    <hobby> Racing </hobby>
</person>

<person >
    <name>
        <firstname> Mary </firstname>
        <surname> Donnelly </surname>
    </name>
</person>
</database>
```

1. /database
2. //surname
3. /*/person[@age]
4. /*/person/string(@age)

```
<database>
<person age='34'>
    <name>
        <title> Mr </title>
        <firstname> John </firstname>
        <firstname> Paul </firstname>
        <surname> Murphy </surname>
    </name>
    <hobby> Football </hobby>
    <hobby> Racing </hobby>
</person>

<person >
    <name>
        <firstname> Mary </firstname>
        <surname> Donnelly </surname>
    </name>
</person>
</database>
```

1. /database
2. //surname
3. /*/person[@age]
4. /*/person/string(@age)

Selecting Several Paths

- By using the | operator in an XPath expression you can select several paths.
- //book/title | //book/price
  - Selects all the title together with price elements of all book elements
- //title | //price
  - Selects all the title together with price elements in the document
- //book/title | //price
  - Selects all the title elements of the book element together with all the price elements in the document

Summary XPath 3 examples

```
<doc type="book" isbn="1-56592-796-9">
  <title>A Guide to XML</title>
  <author>Norman Walsh</author>
  <chapter>[...]</chapter>
  <chapter>
    <title>What Do XML Documents Look
      Like?</title>
    <paragraph>If you are [...]</paragraph>
    <paragraph>A few things [...]</paragraph>
    <ol>
      <item><paragraph>The document begins
         [...]</paragraph></item>
      <item><paragraph type="warning">There's
         no document [...]</paragraph></item>
      <item><paragraph>Empty elements have
         [...]</paragraph>
         <paragraph>In a very
           [...]</paragraph></item>
    </ol>
    <paragraph>XML documents are
      [...]</paragraph>
    <section>[...]</section>
    [...]
  </chapter>
</doc>
```

//paragraph

```
<paragraph>If you are [...]</paragraph>
<paragraph>A few things[...]</paragraph>
<paragraph>The document begins
  [...]</paragraph>
<paragraph type="warning">There's
   no document [...]</paragraph>
<paragraph>Empty elements have
   [...]</paragraph>
<paragraph>In a very [...]</paragraph>
<paragraph>XML documents are
   [...]</paragraph>
```

//ol//paragraph[@type="warning"]

```
<paragraph type="warning">
  There's no document [...]
</paragraph>
```

/doc/chapter[2]/ol/item[position()=last()]

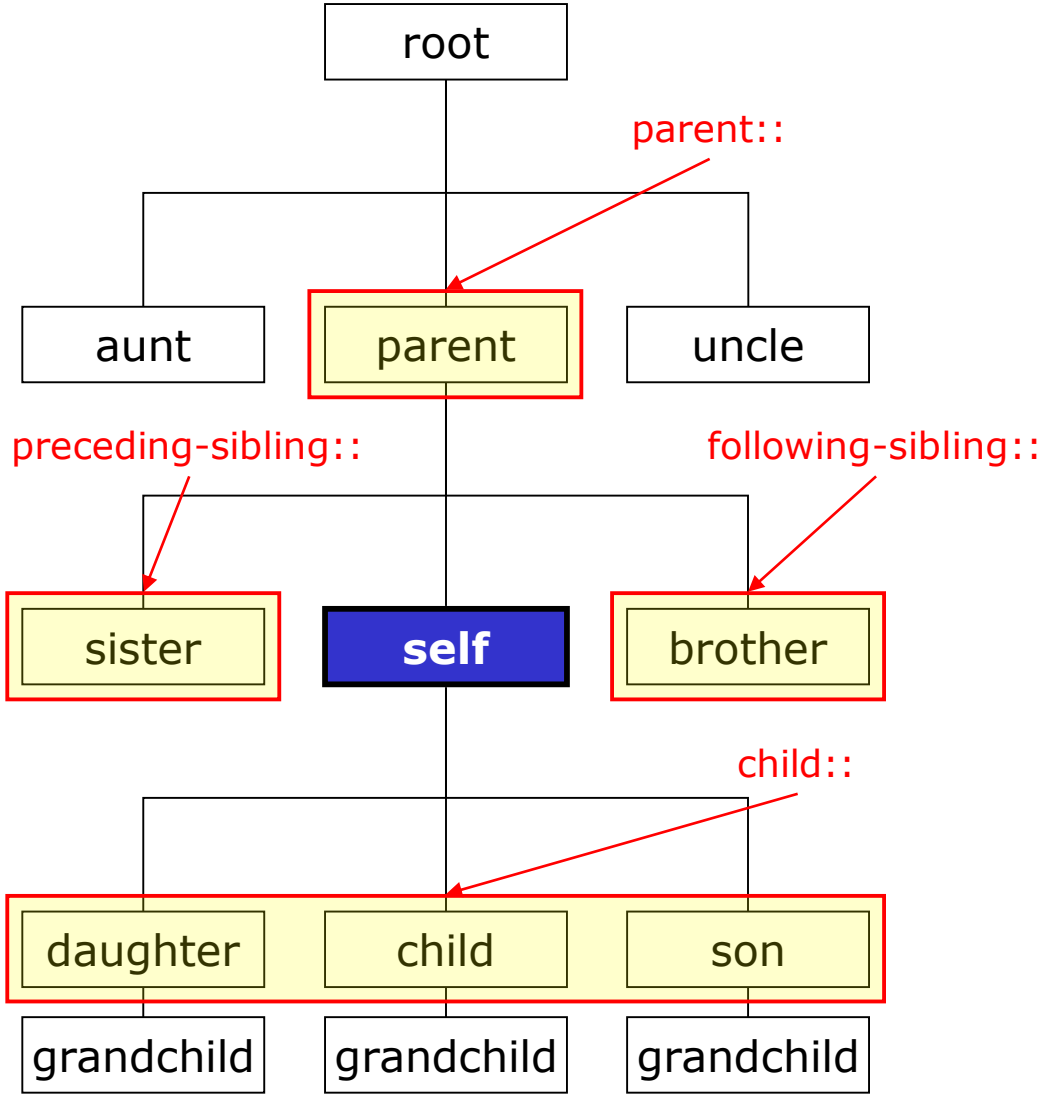
```
<item><paragraph>Empty elements have
  [...]</paragraph>
  <paragraph>In a very [...]</paragraph>
</item>
```

What is XPath?

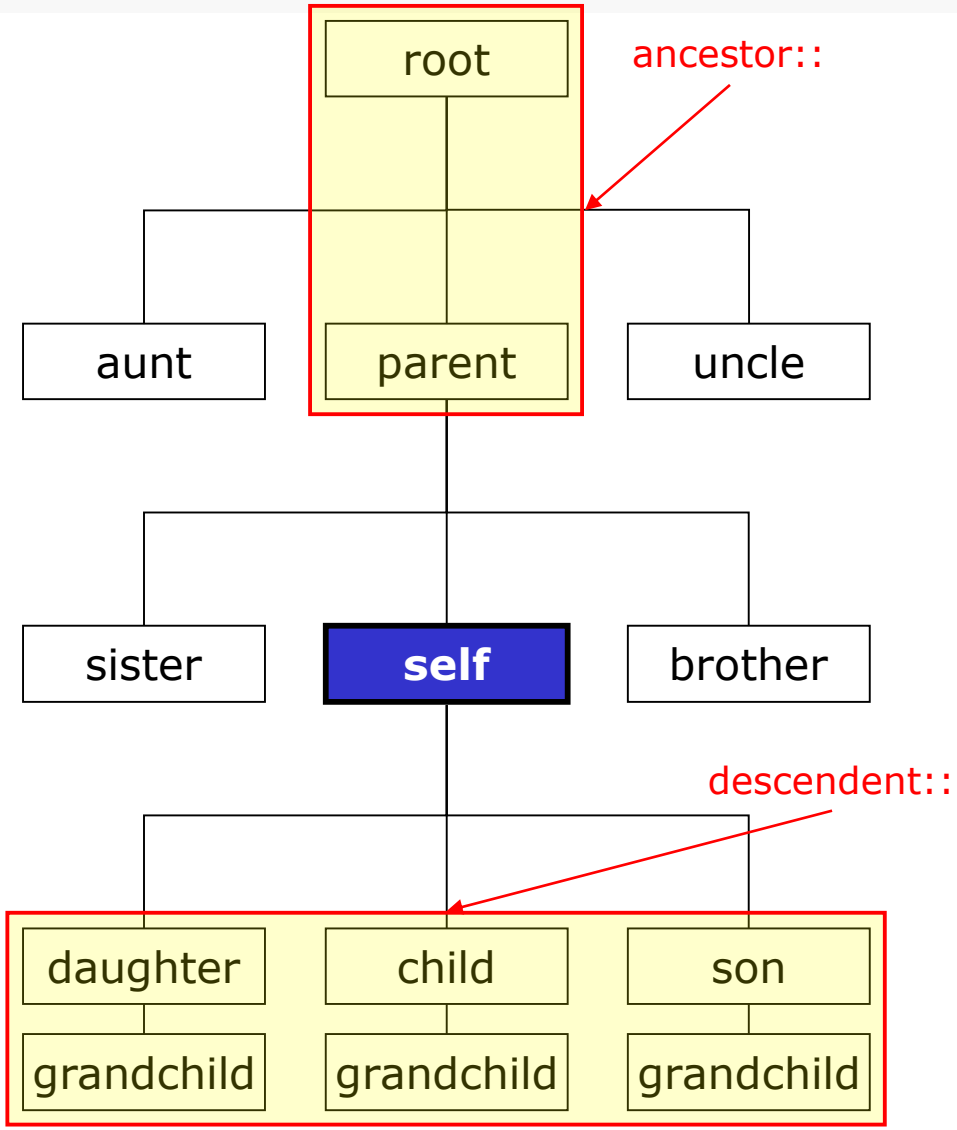
- Expression language
- Used to navigate an XML document
- W3C Recommendation
- Provides basic facilities for manipulation of strings, numbers and booleans
- Compact, non XML syntax
- Operates on the abstract, logical structure of the XML document

Axes spec (1)

There are several directions/axes we can traverse from a node



Axes spec (2)


root
ancestor::
aunt
parent
uncle
sister
self
brother
descendent::
daughter
child
son
grandchild
grandchild
grandchild

Axes

AxisName	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
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 all nodes that appear before the current node in the document, except ancestors, attribute nodes and namespace nodes
preceding-sibling	Selects all siblings before the current node
self	Selects the current node

https://www.w3schools.com/xml/xpath_axes.asp

Example Axes

Example	Result
child::book	Selects all book nodes that are children of the current node
attribute::lang	Selects the lang attribute of the current node
child::*	Selects all element children of the current node
attribute::*	Selects all attributes of the current node
child::text()	Selects all text node children of the current node
child::node()	Selects all children of the current node
descendant::book	Selects all book descendants of the current node
ancestor::book	Selects all book ancestors of the current node
ancestor-or-self::book	Selects all book ancestors of the current node - and the current as well if it is a book node
child::*/child::price	Selects all price grandchildren of the current node

https://www.w3schools.com/xml/xpath_axes.asp

XPath Operators

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

Operator	Description	Example	Return value
|	Computes two node-sets	//book | //cd	Returns 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
=	Equal	price=9.80	true if price is 9.80 false if price is 9.90
!=	Not equal	price!=9.80	true if price is 9.90 false if price is 9.80
<	Less than	price<9.80	true if price is 9.00 false if price is 9.80
<=	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 9.80
>=	Greater than or equal to	price>=9.80	true if price is 9.90 false if price is 9.70
or	or	price=9.80 or price=9.70	true if price is 9.80 false if price is 9.50
and	and	price>9.00 and price<9.90	true if price is 9.80 false if price is 8.50
mod	Modulus (division remainder)	5 mod 2	1

That's All Folks Thank You for Listening