Lecture Notes

Course Number:	CSC 513
Instructor:	Dr. Singh
Lecture Number:	22

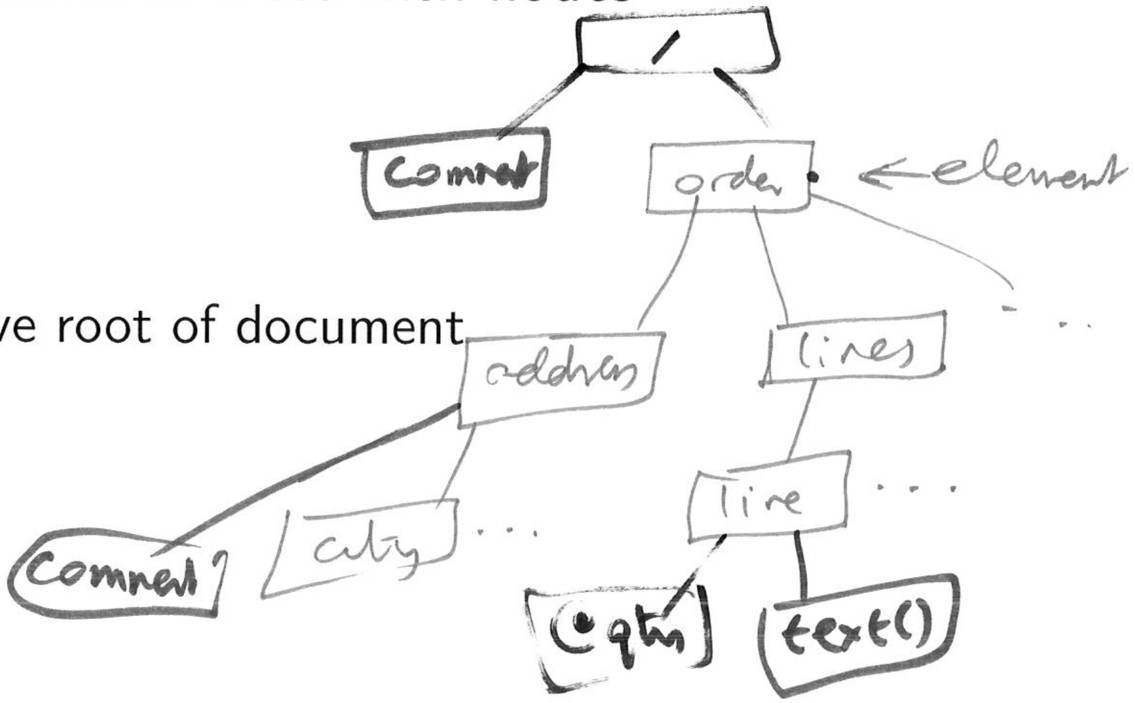
XPath

/element() : e-children

// entire subtree

Used as part of XPointer, SQL/XML, XQuery, and XSLT

- Models XML documents as trees with nodes
  - Elements
  - Attributes
  - Text (PCDATA)
  - Comments
  - Root node: above root of document



Achtung!

XPath: originally was
$\subseteq$ XSLT

quick

- Parent in XPath is like parent as traditionally in computer science
- Child in XPath is confusing:
  - An attribute is not a child of its parent
  - Makes a difference for recursion (e.g., in XSLT **apply-templates**)
- Our terminology follows computer science:
  - e-children, a-children, t-children
  - Sets via et-, ta-, and so on

(Kifer)

XPath Location Paths: 1

Ways to walk a tree

- Relative or absolute
- Reminiscent of file system paths, but *much* more subtle
  - Name of an element to walk down ~~a~~ sub dir
  - Leading /: root
  - /: indicates walking down a tree ← separator
  - .: currently matched (*context*) node
  - ..: parent node

XPath Location Paths: 2

- @attr: to check existence or access value of the given attribute
- text(): extract all the text nodes — t-children within the current or "context" node
- comment(): extract all the comment nodes
- [ ]: generalized array accessors
- Variety of *axes*, discussed below

line[3]

line[@qty = 2]

line[@qty] exists

attribute() : a-children
element() : e-children
node() : all children

XPath Navigation

line[3]

- Select children according to position, e.g., [j], where j could be 1 . . . last()
- Descendant-or-self operator, //
  - .//elem finds all elems under the current node
  - //elem finds all elems in the document
- Wildcard, *:
  - collects e-children (subelements) of the node where it is applied, but omits the t-children
  - @*: finds all attribute values

lines/*

./*

//line/*

//*

order//*

XPath Queries (Selection Conditions)

(Almost Enough for this course exams)

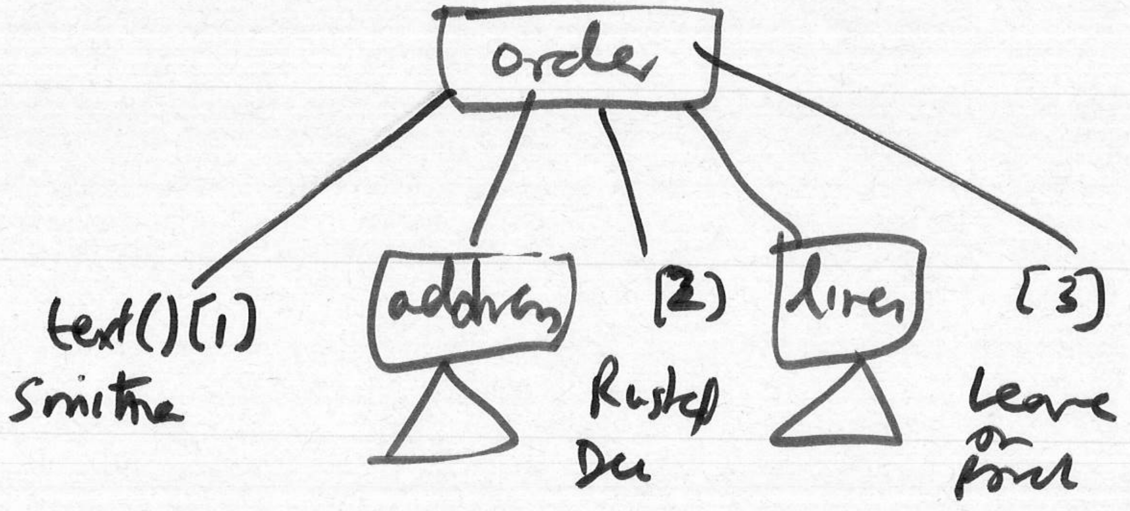
<group> Led Z </group>

- Attributes: //Song[@genre="jazz"]
- Text: //Song[starts-with(.//group, "Led")] → implicit access to the text() node
- Existence of attribute: //Song[@genre]
- Existence of subelement: //Song[group]
- Boolean operators: and, not, or
- Set operator: union (~~+~~), analogous to choice → |
- Arithmetic operators: $\geq$, $\leq$, ...
- String functions: contains(), concat(), length(), starts-with(), ends-with()
- distinct-values()
- Aggregates: sum(), count()

text()

```
<Order>
    Smitha
    <address> ... </address>
    Rushed delivery
    <lines> ... </lines>
    Leave on porch
</order>
```

Each largest contiguous block of characters is a text node



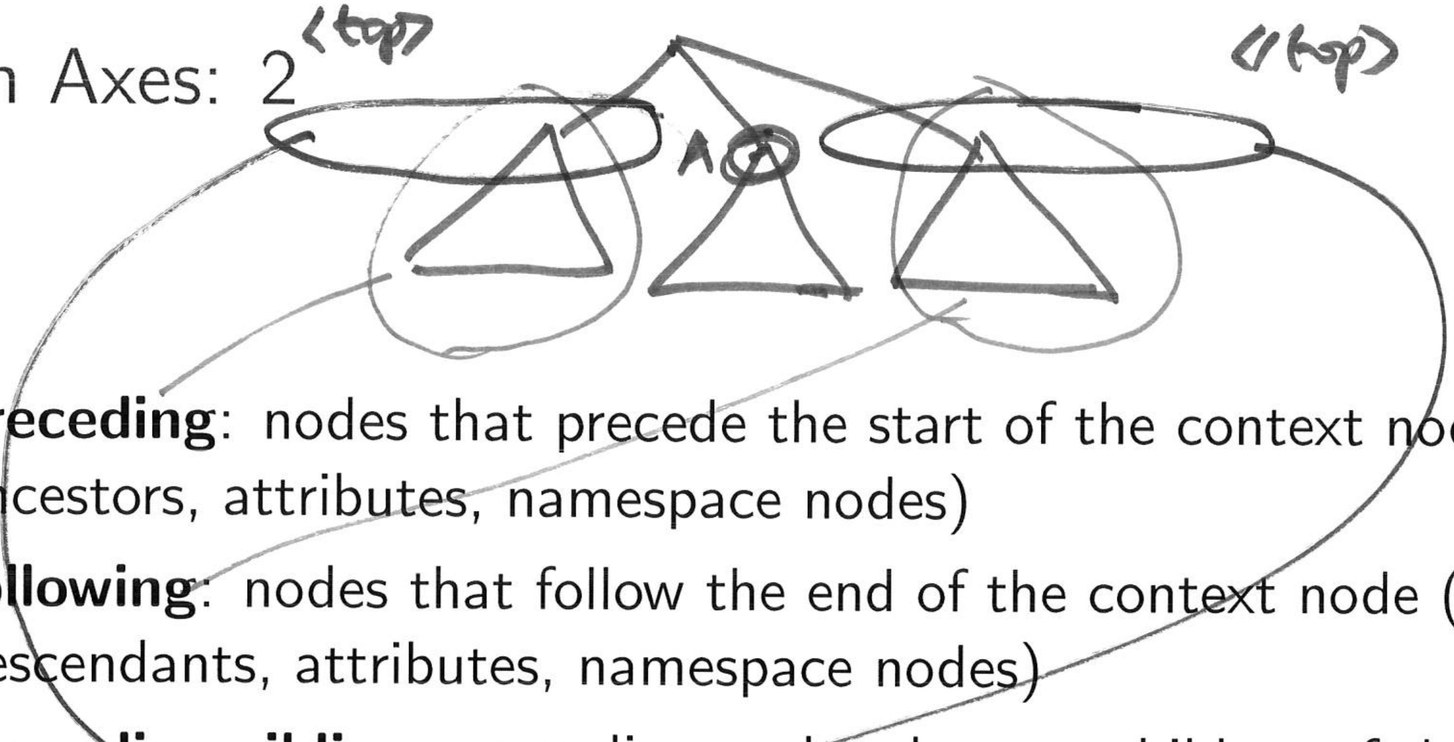
XPath Axes: 1

Axes are addressable node sets based on the document tree and the current node

- Axes facilitate navigation of a tree
- Several are defined
- Mostly straightforward but some of them order the nodes as the reverse of others
- Some captured via special notation
  - **current**, **child**, **parent**, **attribute**, . . .

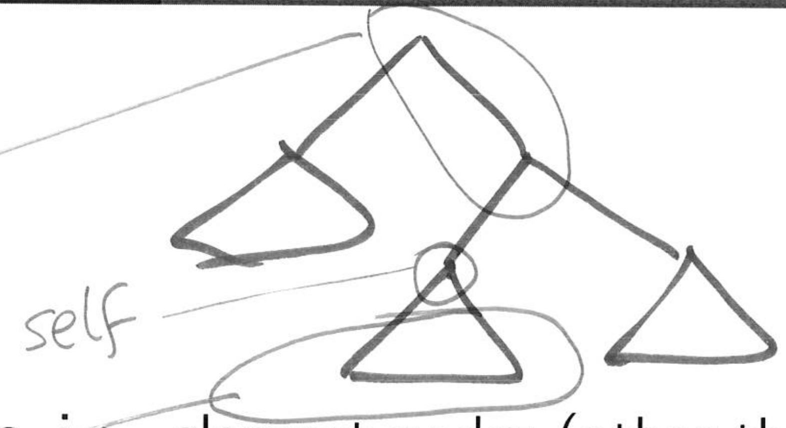
  . /Sgr /.. /@lg

XPath Axes: 2



- **preceding**: nodes that precede the start of the context node (not ancestors, attributes, namespace nodes)
- **following**: nodes that follow the end of the context node (not descendants, attributes, namespace nodes)
- **preceding-sibling**: preceding nodes that are children of the same parent, in reverse document order
- **following-sibling**: following nodes that are children of the same parent

XPath Axes: 3



- **ancestor**: proper ancestors, i.e., element nodes (other than the context node) that contain the context node, in reverse document order
- **descendant**: proper descendants
- **ancestor-or-self**: ancestors, including self (if it matches the next condition)
- **descendant-or-self**: descendants, including self (if it matches the next condition)

XPath Axes: 4

axis:: rest...

- Longer syntax: **child::Song** /Song
- Some captured via special notation
  - **self::***: $\odot$
  - **child::node()**: **node()** matches all nodes that are children of the context node
  - **preceding::***
  - **descendant::text()**
  - **ancestor::Song**
  - **descendant-or-self::node()**, which abbreviates to $\underline{//}$
  - Compare **/descendant-or-self::Song[1]** (first descendant Song) and **//Song[1]** (first Songs (children of their parents))

odd

XPath Axes: 5

- Each axis has a *principal node kind*
  - **attribute**: attribute
  - **namespace**: namespace
  - All other axes: element
- * matches whatever is the principal node kind of the current axis
- **node()** matches all nodes

attribute :: * ≡ attribute :: attribute()

~~self~~ child :: * ≡ child :: element()

XPointer

Enables pointing to specific parts of documents

- Combines XPath with URLs
- URL to get to a document; XPath to walk down the document
- Can be used to formulate queries, e.g.,
  - Song-URL#xpointer(//Song[@genre="jazz"])
  - The part after # is a *fragment identifier*
- Fine-grained addressability enhances the Web architecture

High-level "conceptual" identification of node sets