XML

Digital Multimedia, 2nd edition

Nigel Chapman & Jenny Chapman

Chapter 14

XML

- eXtensible Markup Language
  - Simplified version of SGML
  - W3C Recommendation XML 1.0, 1998
- Define your own set of elements for any type of document
  - *DTD* (Document Type Definition) formal definition of set of elements, their attributes and constraints on usage

XML-based Languages

- SMIL (Synchronized Multimedia Integration Language)
- SVG (Scalable Vector Graphics)
- MathML (Math Markup Language)
- XHTML

These and others are all formally defined by XML DTDs

XML Document Syntax

- 'Like XHTML, except that you can make up your own tags and attribute names'
- <tag> … </tag>
  - <empty />
  - Properly nested
- attribute-name = "attribute-value"
- &entity-reference

A document that follows the syntax rules is *well-formed*

Structure Model

- Nested tags ⇒ hierarchical structure, which can be represented as a tree, k/a *structure model*
- Each element and text section represented by a *node*
- Each node corresponding to an element that is not empty has some *child nodes*, corresponding to its content

Validity

- Specification of a set of permitted elements, the attributes each may have and which elements they can contain defines a class of documents with a certain structure (e.g. the class of all XHTML documents)
- A well-formed document which conforms to the rules in a specification attached to it is said to be *valid*

DTDs and Schemas

- DTD
  - Established form of specification
  - Uses special syntax
  - Cannot express all constraints
- Schema
  - Newer form of specification
  - Uses ordinary XML syntax
  - More powerful than DTD

Referencing a DTD

- XML declaration

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

  (Example of a *Processing Instruction*, *PI*)

- DOCTYPE declaration

  e.g. <!DOCTYPE books PUBLIC "-//DMM//BOOK Bibliographics information 1.0//EN" "http://www.digitalmultimedia.org/DTDs/books.dtd">

  - public name, system identifier

Markup Declarations

- <! *keyword information* >
- Elements
  - <!ELEMENT *name content model* >

  e.g. <!ELEMENT price EMPTY>

Content Models

- EMPTY – no content
- (#PCDATA)* text

  e.g. <!ELEMENT author (#PCDATA)*>
- + one or more, ? optional, * zero or more

  e.g. <!ELEMENT books (book+)>

Content Models

- Sequence: elements &c separated by commas

  e.g. <!ELEMENT book (title, author+, price)>

- Choice |

  e.g. <!ELEMENT book (title, (author+ | editor+), price>

  (Note use of brackets)

Markup Declarations

- <! *keyword information* >
- Attribute lists
  - <!ATTLIST *element name*
    attribute name type requirement
    … >

e.g. <!ATTLIST price
sterling CDATA #REQUIRED
euro CDATA #IMPLIED >

Attribute Types

- CDATA – character data, i.e. strings, including numeric strings
- Enumerations

  e.g. (mon|tue|wed|thu|fri|sat|sun)
- ID – unique identifiers

Required/Optional

- #REQUIRED – attribute must be given a value in start tag
- #IMPLIED – attribute is optional
- Default value – if attribute is omitted, takes on specified value

e.g. <!ATTLIST numberinstock

ordered CDATA "0" >

Namespaces

- Two-level naming system resolves potential name clashes
- Element or attribute names preceded by a *prefix*, separated from name by a colon

  e.g. <bbl:title>, <ppl:title>
- Each prefix is associated with a URL
  - Prefixed name is an abbreviation for combination of URL + name, URLs are always unique, so combination is unique

Declaring Namespaces

- Assign namespace URL to an attribute xmlns:*prefix*
- Usually done in start tag of the document element, so prefix is in scope throughout document

  e.g.

  <bbl:books xmlns:bbl="http://www. …" >

  N.B. The document, if any, at the namespace URL has no significance

Default Namespace

- Assign namespace URL to attribute xmlns
- All names without prefixes belong to the namespace identified by the URL

e.g.

<html xmlns="http://www.w3.org/TR/xhtml1">

All HTML tags can be used without any prefix

Stylesheets and XML

- No layout information at all in XML – purely structure and content
- Use PI to associate a stylesheet with an XML document

e.g.

```
<?xml-stylesheet href="books.css" type="text/css"?>
```

CSS and XML

- display property (no a priori distinction between block and inline elements)
  - values: block, inline, list-item, none, compact
- content property
  - used with :before and :after pseudo-classes
  - attr(X) to insert value of X attribute
- Attribute selectors

XSLT and XSL-FO

- Both XML languages
- XSLT (eXtensible Stylesheet Language for Transformations)
  - Transform structure tree of an XML document into a different tree (e.g. XML → XHTML)
- XML - Formatting Objects
  - Tags that describe how a document should appear when displayed

Linking in XML

- Three components
  - XPointer – language for identifying link destinations
  - XPath – language for addressing nodes in a structure tree, used by XPointer
  - XLink – a set of attributes for constructing elements that serve as links

XPath

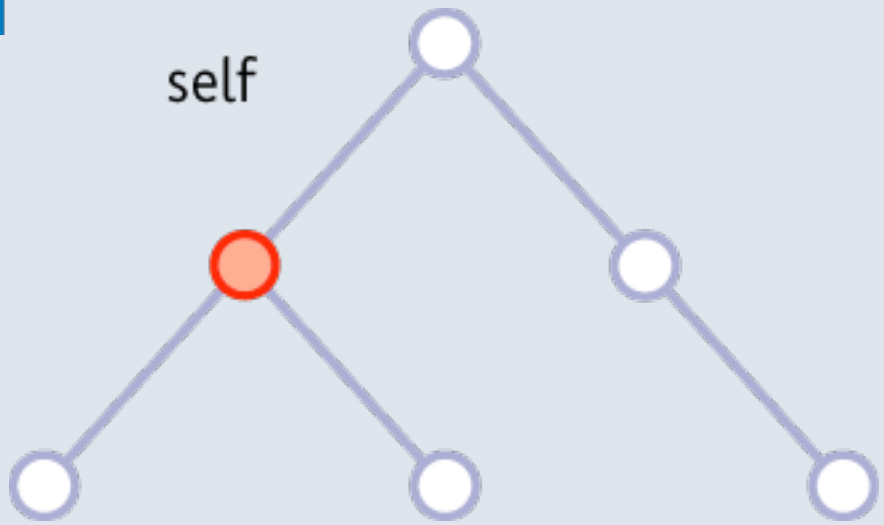
- Special syntax, not XML
- *Location path*
  - Set of instructions (*location steps*), separated by /s, telling you how to reach a specific point in a document
- *Context node*
  - The node reached by following any preceding location steps

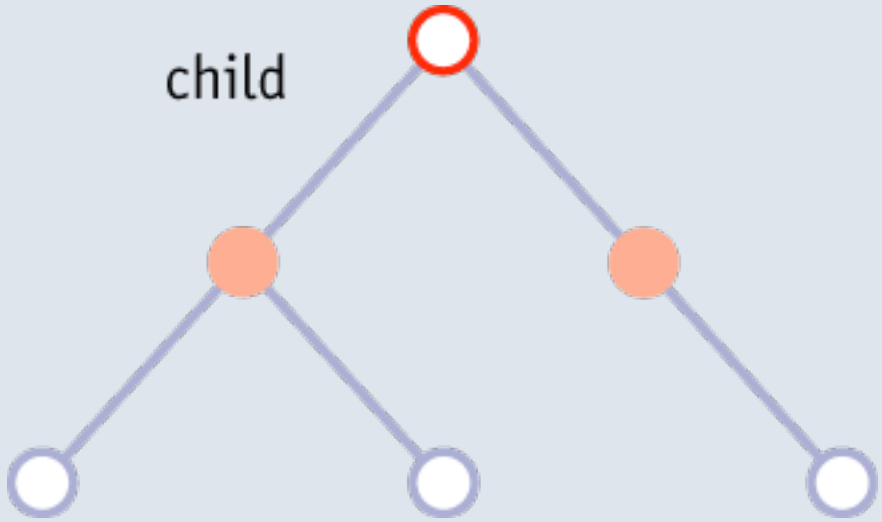
Location Steps

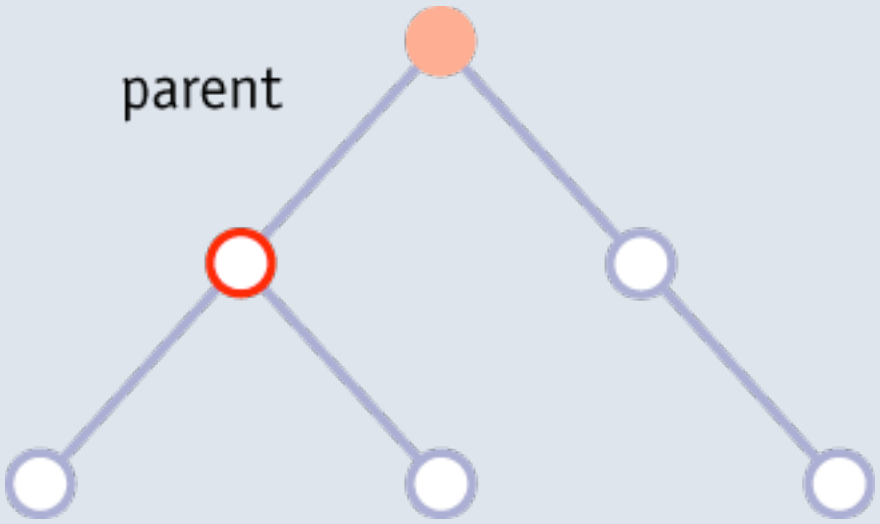
- / – root node
  - Location paths beginning with / are *absolute*
- Element name
  - Set of nodes of that type among the children of the context node
- Element name [number], e.g. book[2]
  - Particular child node of that element type, e.g. 2nd book child

Axes

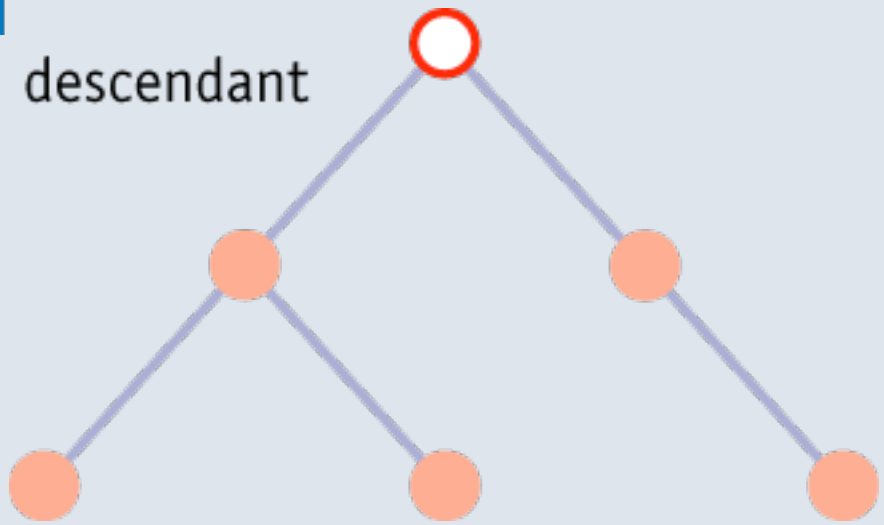
- *axis*::*element name*
  - selects all elements of that type in that axis
  - e.g. following::book – all the book nodes in the following axis
- *name* is shorthand for child::*name*



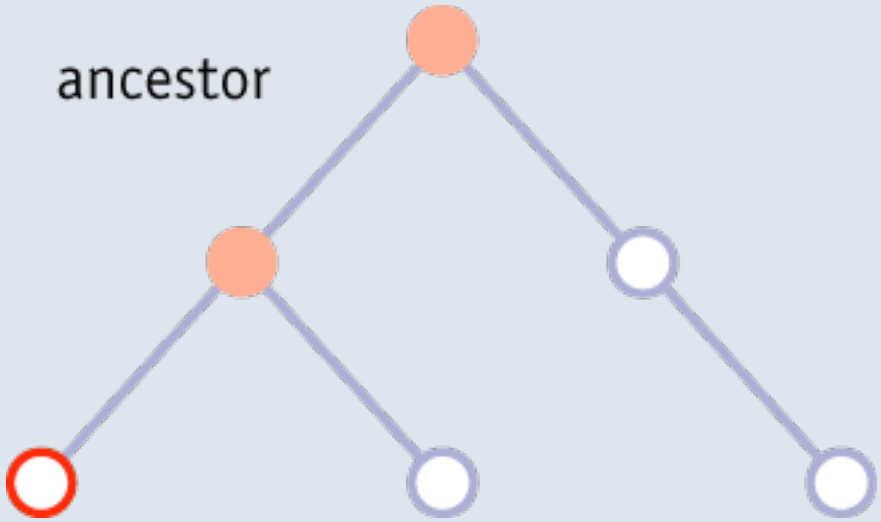




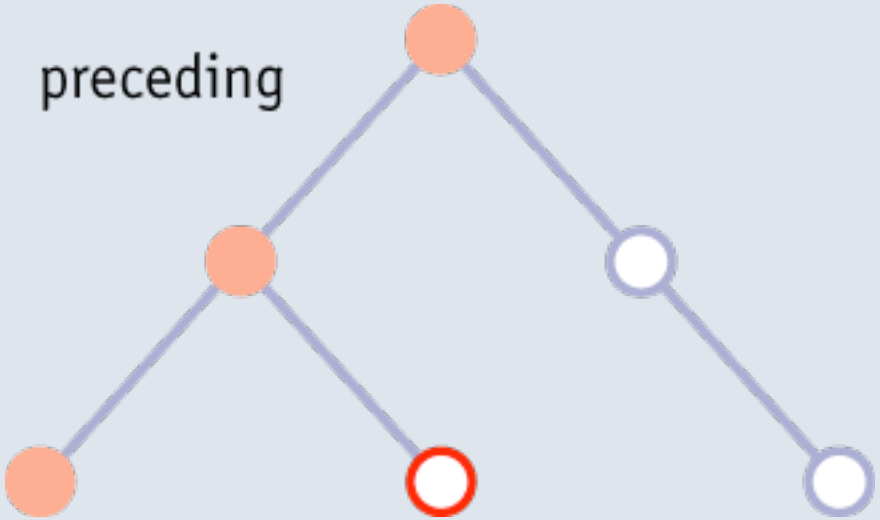
descendant



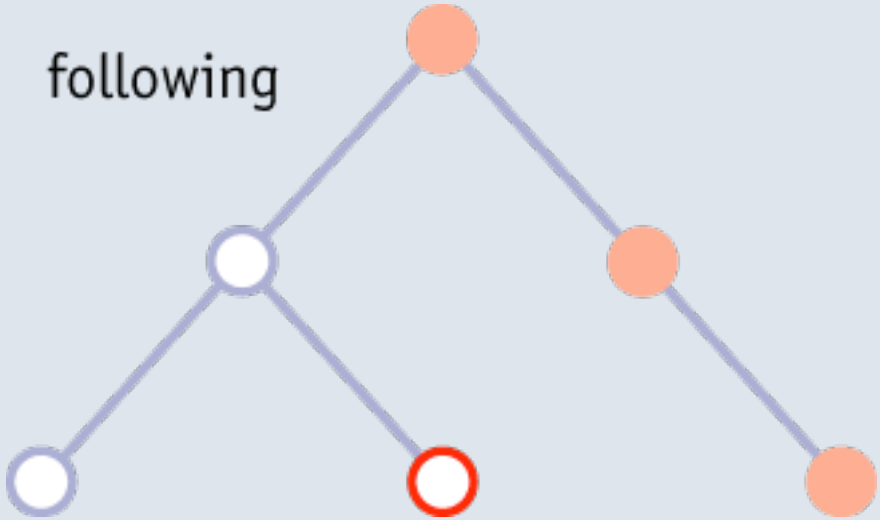
ancestor



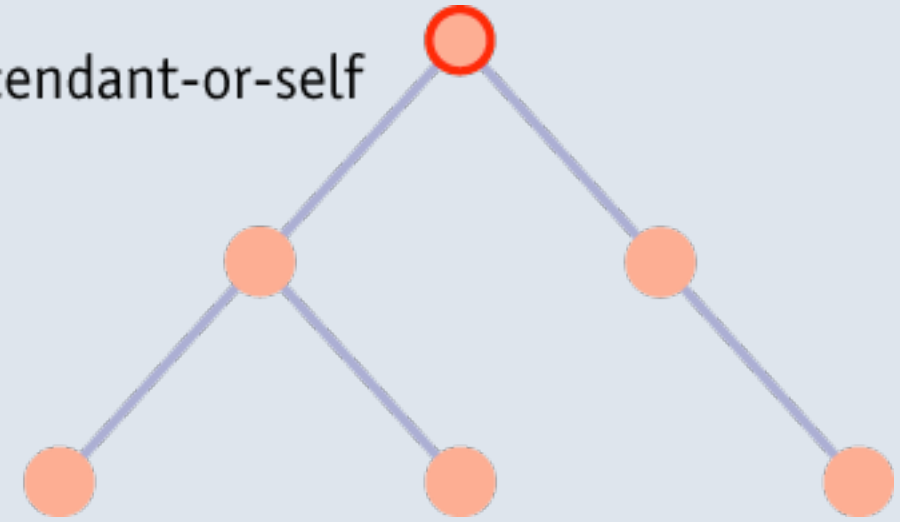
preceding



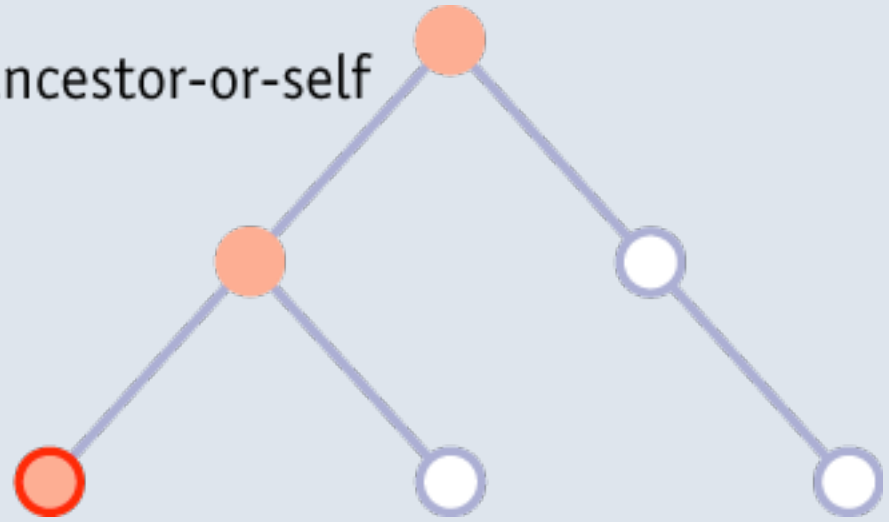
following



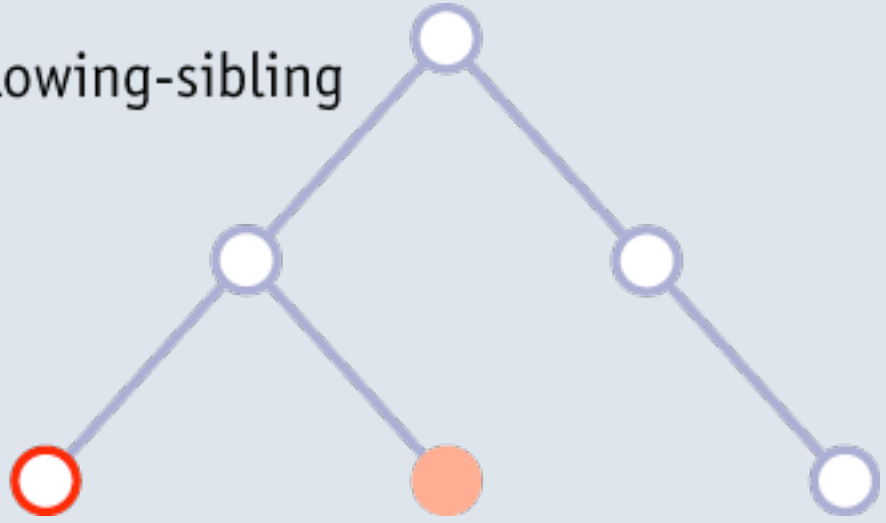
descendant-or-self



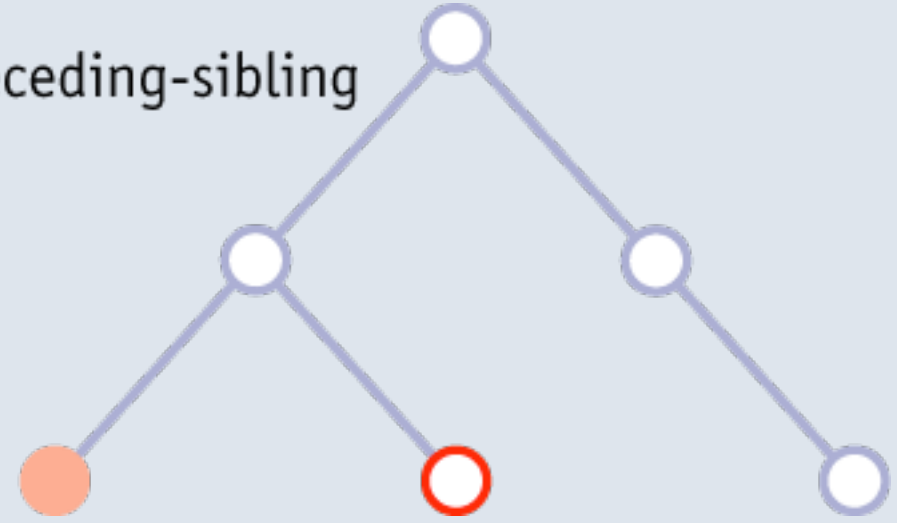
ancestor-or-self



following-sibling



preceding-sibling



XPointer

- Define fragment identifiers
  - Uses XPath to identify parts of a document in terms of structure, where necessary
- *Shorthand pointer: #id*, matches element with attribute of type ID, value id (equivalent to named anchor)
- *Scheme-based pointer: #scheme name*(*data*)
  - #xpointer(*XPath expression*)
  - #element(/n1/n2/...)

XLink

- Namespace containing a collection of *attributes*

  xmlns:xlink="http://www.w3.org/1999/xlink"

- Any element with XLink attribute can be treated as a linking element

- xlink:type determines type of link

  - If xlink:type = *X*, we have an *X-type link*

  e.g. xlink:type="simple" ⇒ simple-type link

Simple-type Links

- xlink:href – locator attribute (like <a> in HTML)
- xlink:show like target
  = "replace" – display in same window
  = "new" – display in new window
  = "embed" – display as part of document (like <img>)
- xlink:actuate – when is link followed
  = "onLoad" – when page loads
  = "onRequest" – when an event occurs (mouse click)

Extended-type Links

- Generalize simple-type links to bi-directional links and multi-links
- Use locator-type elements to point to resources, combine them into relationships as content of extended-type elements
- Add arc-type links to specify traversals
- Store extended-type links in *linkbases*