

Understanding XML

Hans C. Arents
senior IT market analyst

I.T. Works
"Guiding the IT Professional"

Innovation Center, Technologiepark 3, B-9052 Gent (Belgium), Tel: +32 (0)9 241 56 21 - Fax: +32 (0)9 241 56 56

E-mail: hca@itworks.be - Site: <http://www.itworks.be/> - Home: <http://www.arents.be/>

Understanding XML

- n Part 1: XML intro 13u30 – 15u00
coffee break
- n Part 2: **XML standards** 15u30 – 17u30
questions & answers

XML standards

Hans C. Arents
senior IT market analyst

I.T. Works
"Guiding the IT Professional"

Innovation Center, Technologiepark 3, B-9052 Gent (Belgium), Tel: +32 (0)9 241 56 21 - Fax: +32 (0)9 241 56 56
E-mail: hca@itworks.be - Site: <http://www.itworks.be/> - Home: <http://www.arents.be/>

XML standards

n How does the standards approval process work?

n Standards for handling XML

- XML namespaces
- XML Schemas

n Standards for building XML solutions

- XQL: **E**xtensible **Q**uery **L**anguage
 - XPath
- XSL: **E**xtensible **S**tylesheet **L**anguage
 - XSLT (using XPath) and XSLFO
- XLL: **E**xtensible **L**inking **L**anguage
 - XLink and XPointer (using XPath)
- DOM: **D**ocument **O**bject **M**odel

The W3C standards approval process

n Recommendation track:



- **Notes**

non-official documents, used for initiating Working Group discussions

- **Working Drafts (Last Call Working Draft)**

official documents, representing work in progress without consensus

- **Candidate Recommendation**

an official document, a stable Working Draft being proposed for implementation experience and feedback

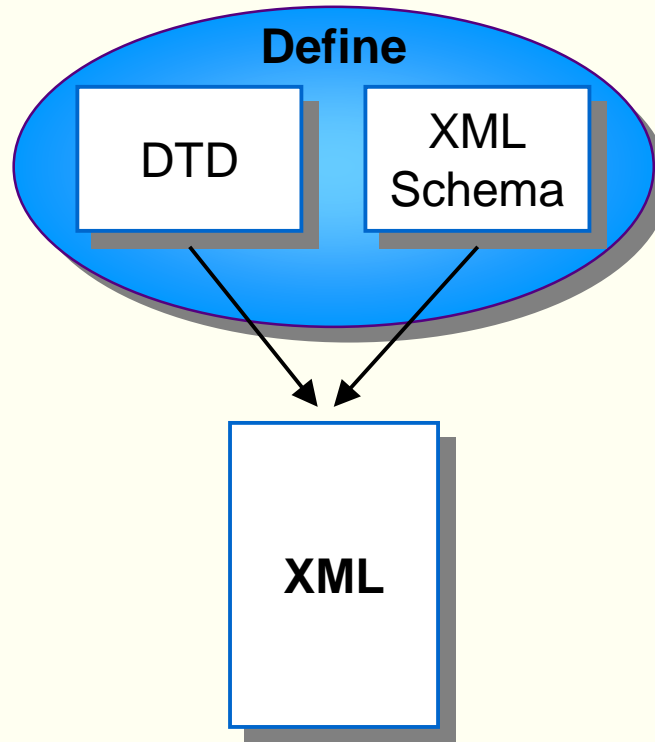
- **Proposed Recommendation**

an official document, a Candidate Recommendation which is sent to the W3C Advisory Committee for final review

- **Recommendation**

an official document, a Proposed Recommendation on which a consensus has been reached and which becomes a W3C standard

Standards for handling XML

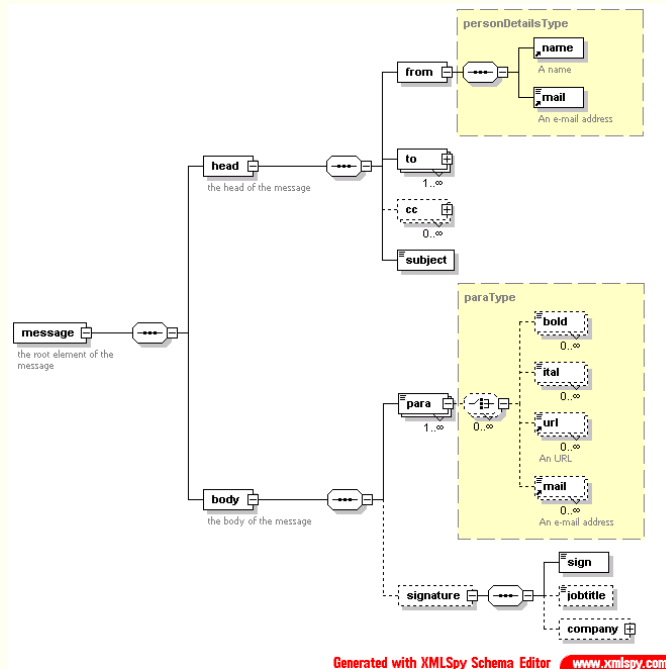


Standards for handling XML

“how to work with the XML syntax”

XML namespaces: *how to mix XML tags*

DTD/XML Schema: *how to define XML vocabularies*



XML namespaces

n Problem:

- same tag may mean different things
 - e.g. `<title>` = document title *or* formal title of author
 - e.g. `<quote>` = stock quote *or* quote in a person's speech

n Solution:

- create tags and attributes that are unique in the context of their specific use
- by adding a prefix to the tag or attribute to make clear that it belongs to a specific namespace
 - e.g. `<easdaq:quote>...</easdaq:quote>`

qualified name

n Note:

- not a way of making modular DTDs
- just a syntactic way of giving to elements and attributes programmer-friendly names that are unique across the whole Internet

XML namespaces

```
<?xml version="1.0" standalone="yes"?>
<h:html xmlns:h="http://www.w3.org/1999/xhtml"
        xmlns:a="http://www.amazon.com/books">
  <h:head>
    <h:title>Book Review</h:title>
  </h:head>
  <h:body>
    <a:bookreview>
      <a:title>XML: A Primer</a:title>
      <h:table>
        <h:tr align="center">
          <h:td>Author</h:td><h:td>Price</h:td>
          <h:td>Pages</h:td><h:td>Date</h:td>
        </h:tr>
        <h:tr align="left">
          <h:td><a:author>Simon St. Laurent</a:author></h:td>
          <h:td><a:price>31.98</a:price></h:td>
          <h:td><a:pages>352</a:pages></h:td>
          <h:td><a:date>1998/01</a:date></h:td>
        </h:tr>
      </h:table>
    </a:bookreview>
  </h:body>
</h:html>
```

XML namespaces

```
<?xml version="1.0" standalone="yes"?>
<h:html xmlns:h="http://www.w3.org/1999/xhtml"
        xmlns:a="http://www.amazon.com/books">
  <h:head>
    <h:title>Book Review</h:title>
  </h:head>
  <h:body>
    <a:bookreview>
      <a:title h:style="font-family: Arial;">XML: A Primer</a:title>
      <h:table>
        <h:tr align="center">
          <h:td>Author</h:td><h:td>Price</h:td>
          <h:td>Pages</h:td><h:td>Date</h:td>
        </h:tr>
        <h:tr align="left">
          <h:td><a:author>Simon St. Laurent</a:author></h:td>
          <h:td><a:price>31.98</a:price></h:td>
          <h:td><a:pages>352</a:pages></h:td>
          <h:td><a:date>1998/01</a:date></h:td>
        </h:tr>
      </h:table>
    </a:bookreview>
  </h:body>
</h:html>
```

XML namespaces

```
<?xml version="1.0" standalone="yes"?>
<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:a="http://www.amazon.com/books">
  <head>
    <title>Book Review</title>
  </head>
  <body>
    <a:bookreview>
      <a:title>XML: A Primer</a:title>
      <table>
        <tr align="center">
          <td>Author</td><td>Price</td>
          <td>Pages</td><td>Date</td>
        </tr>
        <tr align="left">
          <td><a:author>Simon St. Laurent</a:author></td>
          <td><a:price>31.98</a:price></td>
          <td><a:pages>352</a:pages></td>
          <td><a:date>1998/01</a:date></td>
        </tr>
      </table>
    </a:bookreview>
  </body>
</html>
```

XML Schema

Data marked up using XML:

```
<ShoeOrder>
  <Color>Brown</Color>
  <Size>43.5</Size>
</ShoeOrder>
```

Incorrect XML:

```
<ShoeOrder>
  <Color>Brown</Color>
</ShoeOrder>

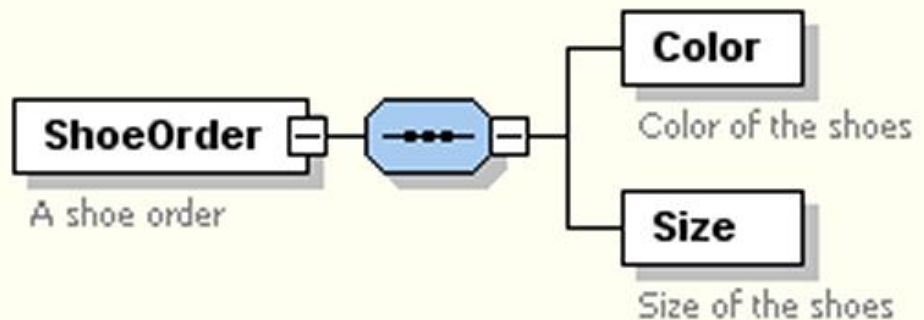
<ShoeOrder>
  <Size>43.5</Size>
  <Color>Brown</Color>
</ShoeOrder>
```

Correct XML:

```
<ShoeOrder>
  <Color>Brown</Color>
  <Size>very big</Size>
</ShoeOrder>
```

Schema defined using a “classic” DTD:

```
<!ELEMENT ShoeOrder (Color,Size)>
<!ELEMENT Color      (#PCDATA)  >
<!ELEMENT Size       (#PCDATA)  >
```



XML Schema

n Problem:

- XML DTD is in a different syntax than XML documents
 - è use XML syntax
- XML DTD is not expressive enough for rich data modeling
 - è add datatypes

n Solution: **XML Schema**

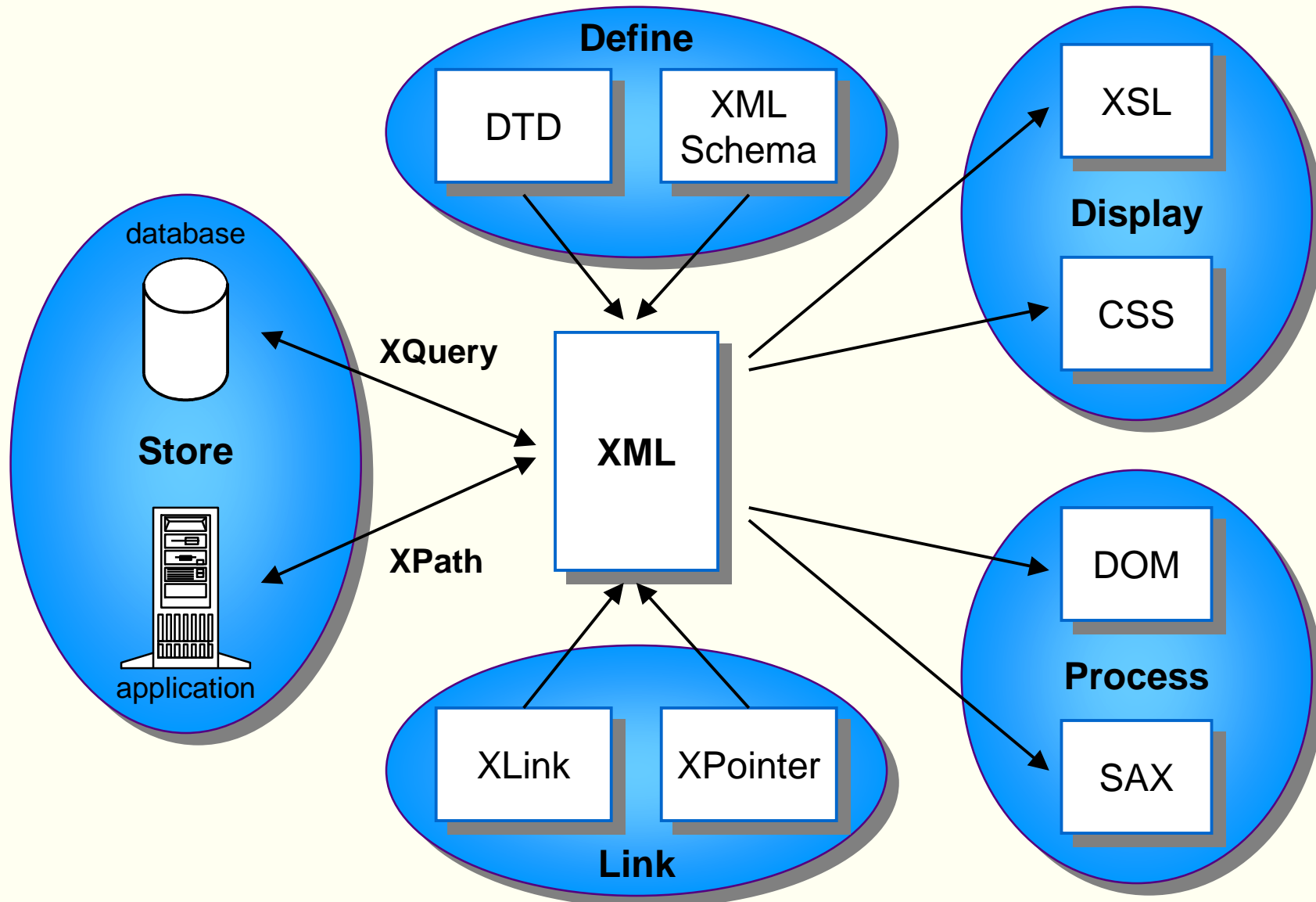
- Part 1: **Structure**
 - defining XML DTDs using XML syntax
- Part 2: **Datatypes**
 - defining datatypes
 - defining datatype constraints to be applied to the contents of XML elements or attributes

n Status: W3C Recommendation (May 2, 2001)

XML Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema"
            elementFormDefault="qualified">
  <xsd:element name="ShoeOrder">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Color" type="shoeColors"/>
        <xsd:element name="Size" type="shoeSizes"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:simpleType name="shoeColors">
    <xsd:restriction base="xsd:string">
      <xsd:enumeration value="Black"/>
      <xsd:enumeration value="Brown"/>
      <xsd:enumeration value="Grey"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="shoeSizes">
    <xsd:restriction base="xsd:float">
      <xsd:minInclusive value="17"/>
      <xsd:maxInclusive value="47"/>
      <xsd:pattern value="\d\d(\.5)?"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:schema>
```

Standards for building XML solutions



Standards for building XML solutions

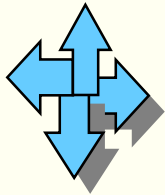
“how to do something with XML documents/data”



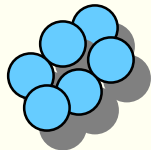
XQuery : XML Query Language
how to query XML



XSL : **E**x**t**ensible **S**tylesheet **L**anguage
how to transform and render XML



XLL : **E**x**t**ensible **L**inking **L**anguage
how to link in XML



DOM: **D**ocument **O**bject **M**odel
SAX : **S**imple **A**PI for **X**ML
how to process XML

XQuery: XML Query Language

n Language for querying XML based on:

- **XML Query Requirements**: W3C Working Draft (Feb 15, 2001)
 - declarative query language
 - must have both an XML and a non-XML syntax
 - suited for querying XML documents and data, full-text retrieval, ...
 - examples in XML Query Use Cases: W3C Working Draft (Feb 15, 2001)
- **XML Query Data Model**: W3C Working Draft (Feb 15, 2001)
 - node-labeled, tree-constructor representation of a tree
 - including the concept of node identity to simplify the representation of XML reference values (e.g. IDREF, XPointer, and URI values)
- **XML Query Algebra**: W3C Working Draft (Feb 15, 2001)
 - well-defined semantics for the query language
 - projection, iteration, selection, quantification
 - joining, grouping, restructuring, sorting, aggregation
 - set of equivalence laws useful for query optimization

XQuery: XML Query Language

n XQuery syntax

- based on XML Query Data Model and XML Query Algebra
- ideas taken from SQL: use of patterns of query clauses
 - "FLWR" expressions: FOR, LET, WHERE and RETURN
- ideas taken from OQL: functional language with nestable expressions
- uses a *locating* language **XPath**

W3C Recommendation (November 16, 1999)

- to search for particular element structures
- to search for elements with particular content / particular attributes

n Note:

- before XQuery, number of competing suggestions
 - XQL (**X**ML **Q**uery **L**anguage), XML-QL, Lorel, ...

most XML data servers support (a subset/superset of) XQL

- XQL = XPath + extensions, so not a query language!

XPath: XML Path Language

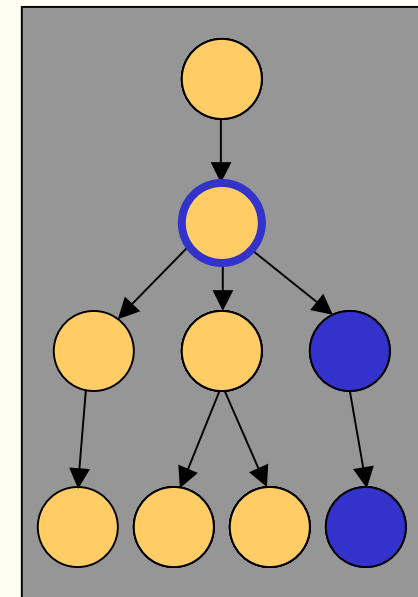
n Language for locating things in an XML document

- by stepping through the document using a **location path**
- location path is string-based rather than tag-based

n Location path:

- is built from **location steps** concatenated with /
 - `location step/location step/location step`
- a location step specifies a point in the document relative to another point in the document:
 - **absolute**
 - starting from the root "/"
 - **relative** from the *context node*
 - starting from the current node
 - from a well-defined **position**
 - using `id()` or `document()` functions

along a certain **axis**



XPath: examples

n `id("c725")` the element with unique identifier c725

n `root()` *abbreviated to* `/` the root of the document

n `child::para` *abbreviated to* `para`

`self::node()/para` *abbreviated to* `./para`

- all of the para children of the context node

n `parent::node()/para` *abbreviated to* `../para`

- all of the para children of the parent of the context node

n `/descendant::node()/para[position()=5]`

abbreviated to `//para[5]`

- the fifth para element anywhere in the document

n `attribute::align` *abbreviated to* `@align`

- the align attribute of the context node

XPath: examples

n list/item

- all of the item children of all of the list children of the context node

n list//para

- all of the para elements (nested arbitrarily deep) under all of the list children of the context node. In other words, this location path matches list/item/para, list/item/note/para, ...

n note[@type="warning"]

- only note children that have a type attribute with the value "warning"

n section[title="Introduction"]

- all the section children with a title of "Introduction"

n //section[title="Examples"]/example[last()]

- the last example element in all section elements (anywhere in the document) with a title element of "Examples"

Storing XML data

The screenshot displays two windows. The top window, Microsoft Internet Explorer, shows the source code of an XML file named 'memos.xml'. The code defines a root element 'memos' containing three 'memo' elements with different priorities: Medium, Low, and High. Each memo includes a 'head' section with 'from', 'to', and 'subj' attributes, and a 'body' section with a 'para' element containing a 'sign' element with the author's name.

The bottom window, eXcelon Explorer, shows the XML tree structure of the same file. The tree is rooted at 'DOCTYPE' (memos SYSTEM "memosxml.dtd") and branches into 'memos', 'memo', 'priority', 'head', 'from', 'to', 'subj', 'body', 'para', and 'sign' elements. The 'Value' column shows the corresponding values for each element, such as 'Medium', 'Hans C. Arents - senior consultant', and 'XML seminar price'.

```

<?xml version="1.0" ?>
<!DOCTYPE memos (View Source for full doctype...)>
- <memos>
- <memo priority="Medium">
- <head>
  <from>hca@offis.be</from>
  <to>participant1@xmlseminar.itworks.be</to>
  <to>participant2@xmlseminar.itworks.be</to>
  <subj>What is XML?</subj>
</head>
- <body>
+ <para>
  <sign>Hans C. Arents - senior consultant</sign>
</body>
</memo>
- <memo priority="Low">
- <head>
  <from>paul@protext.be</from>
  <to>participant2@xmlseminar.itworks.be</to>
  <to>participant3@xmlseminar.itworks.be</to>
  <subj>XML seminar price</subj>
</head>
- <body>
+ <para>
  <sign>Paul Hermans - senior consultant</sign>
</body>
</memo>
- <memo priority="High">
- <head>
  <from>bmarchal@pineapplesoft.com</from>
  <to>participant1@xmlseminar.itworks.be</to>
  <to>participant3@xmlseminar.itworks.be</to>
  <subj>Why use XML for databases?</subj>
</head>
- <body>
+ <para>
  <sign>Benoit Marchal - consultant</sign>
</body>
</memo>
</memos>
  
```

For Help, press F1

Querying XML data using XPath

The image shows a screenshot of a software interface with two main windows. On the left is the 'Query Wizard' dialog box, and on the right is the 'eXcelon Explorer' window.

Query Wizard: The title bar reads 'Query Wizard'. Below the title bar is the instruction 'Build a query by selecting elements and, optionally, defining constraints'. The main area contains a tree view under the heading 'Get element'. The tree shows a hierarchy: 'memos' (selected) contains 'memo' (selected), which contains 'priority', 'head', and 'body'. 'body' contains 'para', 'bold', 'ital', and 'sign'. The 'body', 'para', 'bold', 'ital', and 'sign' elements are checked. Below the tree is a checkbox for 'Display Context' (unchecked) and a text field for the 'Query:' containing the XPath expression: `/memos/memo[@priority="High"]/body`. An 'OK' button is at the bottom right.

eXcelon Explorer: The title bar reads 'eXcelon Explorer - [Query: TestXMLStore:MemosXQL/query1.xql]'. The menu bar includes 'Connection', 'File', 'Edit', 'Tools', 'Window', and 'Help'. The main area displays the XML output of the query, with the XPath expression `/memos/memo[@priority="High"]/body` highlighted in red. The XML content is as follows:

```
<?xml version="1.0" ?>
- <xmlns:xlsql:result xmlns:xlsql:count="2" xmlns:xlsql="http://www.ObjectDesign.com/eXcelon/namespaces/query">
- <body>
- <para>
  The price for the "XML in Perspective" seminar (22 June 1999) is
  <b>15.500 BEF</b>
  /384,23 EUR (+ 21% VAT). The price for the "XML at Work" workshop (23 June 1999) is
  <b>17.500 BEF</b>
  /433,81 EUR (+ 21% VAT). If you register for both events, the price is only
  <b>29.500 BEF</b>
  /731,29 EUR (+ 21% VAT). The participation price includes the event, syllabus, dinner, coffee/tea, the book "Teach Yourself XML in 21 Days", and a lot of background information on XML. The number of participants for the seminar (22 June 1999) is limited to
  <i>125</i>
  , but the number of participants for the workshop (23 June 1999) is limited to
  <i>60</i>
  .
</para>
<sign>Paul Hermans - senior consultant</sign>
</body>
- <body>
- <para>
  Database developers are likely to be involved with XML for several reasons. You might develop applications that query databases and format the query results as XML documents. You might have to develop databases and write code to support business-to-business integration or other forms of electronic data interchange (
  <b>EDI</b>
  ). You might also have to store XML documents in a database. Today's object-relational DBMS products from Oracle, Informix, and IBM support text databases and sophisticated text searching. It's no surprise they are developing XML-centric extensions to their product lines. Some XML advocates have asserted that XML and databases are divergent technologies. This assertion is based in part on an erroneous assumption that a DBMS can manage row and column data, but not tree-structured data. Database veterans recognize that hierarchical and Conference on Data Systems Languages (
  <b>CODASYL</b>
  ) model databases have no problem with hierarchies or trees. Those are legacy technologies, but object and object-relational databases also store text data and operate with tree structures.
</para>
<sign>Benoit Marchal - consultant</sign>
</body>
</xlsql:result>
```

At the bottom of the eXcelon Explorer window, there is a status bar that reads 'For Help, press F1' and a 'NUM' indicator.

Querying XML data using XPath

The screenshot displays the eXcelon Explorer application interface. On the left, the 'Query Wizard' dialog box is open, showing a tree view of an XML document structure. The tree is rooted at 'memos' and contains a 'memo' element with sub-elements 'priority', 'head', 'from', 'to', 'subj', 'body', 'para', 'bold', 'ital', and 'sign'. The 'body' and 'para' elements are selected. Below the tree, the 'Query:' field contains the XPath expression: `/memos/memo[head/to='participant1@xmls]`. The 'OK' button is visible at the bottom of the wizard.

The main window of eXcelon Explorer shows the XML document being queried. The XML content is as follows:

```
<?xml version="1.0" ?>
- <xlnxql:result xlnxql:count="2" xmlns:xlnxql="http://www.ObjectDesign.com/eXcelon/namespaces/query">
- <body>
- <para>
  E
  <bold>x</bold>
  tensible
  <bold>M</bold>
  arkup
  <bold>L</bold>
  ange (XML) was approved just a little more than a year ago by the World Wide Web Consortium (W3C) as an official
  Web standard. It is rapidly becoming the strategic instrument for defining networked corporate data across a number
  of application domains. The properties of XML markup make it suitable for representing
  <ital>data, concepts, and contexts</ital>
  in an
  <ital>open platform-, vendor-, and language-neutral manner</ital>
  . XML enables users to deliver structured data over the Web and is applicable to functions such as authoring, browsing,
  content analysis, application integration and others.
</para>
<sign>Hans C. Arents - senior consultant</sign>
</body>
- <body>
- <para>
  Database developers are likely to be involved with XML for several reasons. You might develop applications that query
  databases and format the query results as XML documents. You might have to develop databases and write code to
  support business-to-business integration or other forms of electronic data interchange (
  <bold>EDI</bold>
  ). You might also have to store XML documents in a database. Today's object-relational DBMS products from Oracle,
  Informix, and IBM support text databases and sophisticated text searching. It's no surprise they are developing XML-
  centric extensions to their product lines. Some XML advocates have asserted that XML and databases are divergent
  technologies. This assertion is based in part on an erroneous assumption that a DBMS can manage row and column
  data, but not tree-structured data. Database veterans recognize that hierarchical and Conference on Data Systems
  Languages (
  <bold>CODASYL</bold>
  ) model databases have no problem with hierarchies or trees. Those are legacy technologies, but object and object-
  relational databases also store text data and operate with tree structures.
</para>
<sign>Benoit Marchal - consultant</sign>
</body>
```


XQuery: XML Query Language

n Examples:

- list the titles of books published by Morgan Kaufmann in 1998

```
FOR $b IN document("bib.xml")/book
WHERE $b/publisher = "Morgan Kaufmann" AND $b/year = "1998"
RETURN $b/title
```

- list each publisher and the average price of its books

```
FOR $p IN distinct(document("bib.xml")//publisher)
LET $a := avg(document("bib.xml")/book[publisher = $p]/price)
RETURN
  <publisher>
    <name> $p/text() </name> ,
    <avgprice> $a </avgprice>
  </publisher>
```

- n Note: XQuery does not contain insert/update/delete operations, so it is not an XML data manipulation language (yet)!

Manipulating XML data (eXcelon)

```
<?xml version="1.0"?>
<xlnupdate version="1.0">

  <remove select="/bookstore/book[-1]">
  </remove>

  <update select="/bookstore">
    <element location="lastchild">
      <book style="tour">
        <title>Sightseeing in Trenton</title>
        <price>5.99</price>
      </book>
    </element>
  </update>

  <update select="/bookstore/magazine[1]">
    <attribute name="frequency">weekly</attribute>
  </update>

  <update select="/bookstore/book/title='History of Trenton'">
    <element>
      <title>History of Trenton 1800-1900</title>
    </element>
  </update>

</xlnupdate>
```

XSL: Extensible Stylesheet Language

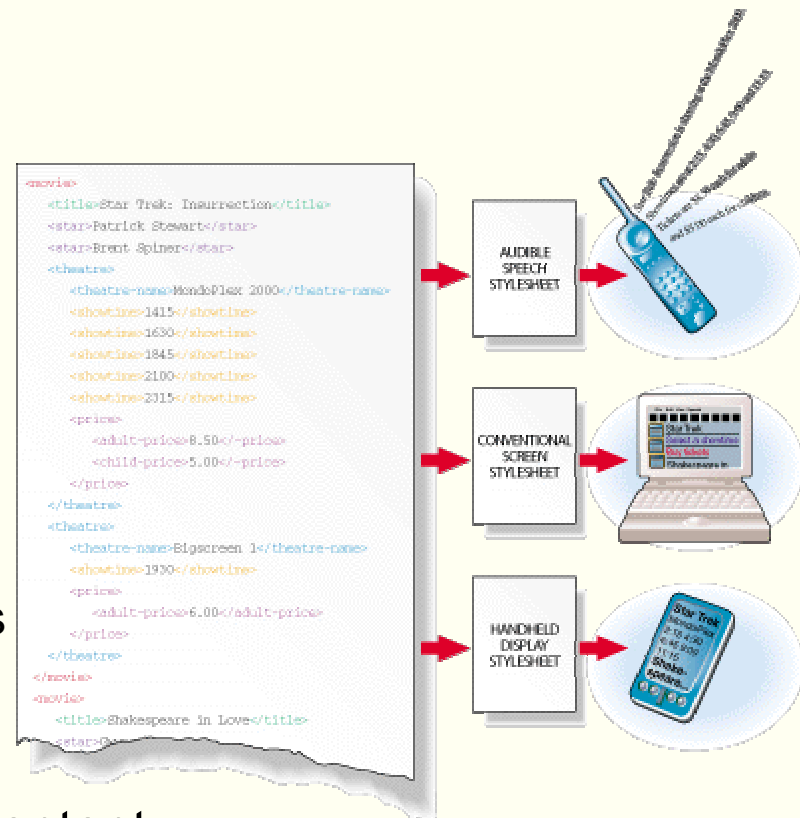
n XML stylesheet mechanism

- based on DSSSL
- compatible with CSS 1.0, 2.0 & 3.0
- advanced capabilities:
 - contents can be reordered/sorted
 - text can be deleted/duplicated/moved
 - new information can be “computed” from existing information or structures

n Useful for:

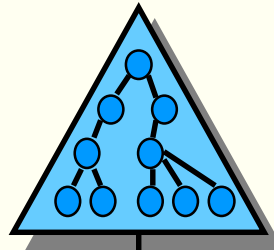
- keeping presentation separate from content
- single source, multiple output media
- reformatting / re-using content

n Not just document formatting, but also data restructuring!



XSL: Extensible Stylesheet Language

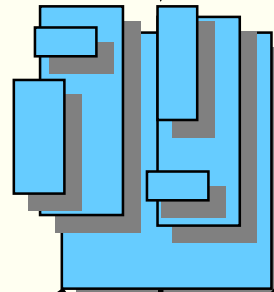
source tree



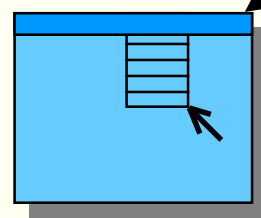
XSL stylesheet

- different views
- different layouts
- dynamic contents

result tree with
formatting
objects



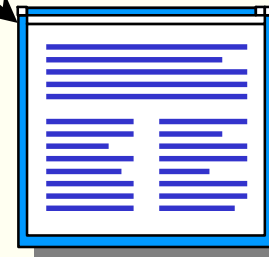
output
media



Java GUI



paper
document



Web browser

...

XSL: key concepts

n Two separate processes:

- transformation: **XSLT** (XSL **T**ransformations) using XPath
- formatting: **XSLFO** (XSL **F**ormating **O**bjects)

☒ Transformation

- how an XML source tree should be transformed
- recursive and declarative programming

• Formatting

- how the XML result tree should be presented
- specified using **f**ormating **o**bjects (FOs)

n Status:

- XSLT: W3C Recommendation (November 16, 1999)
 - XSLT 1.1: W3C Working Draft (December 12, 2000)
 - XSLT 2.0 Requirements: W3C Working Draft (February 14, 2001)
- XSLFO: W3C Candidate Recommendation (November 21, 2000)

XSLT = transformation

Input:

```
<birthday><date>December 3, 1997</date>  
      <fullname>John Doe</fullname>  
</birthday>
```

Stylesheet:

```
<xsl:stylesheet xmlns=...>  
  <xsl:template match="birthday/date">  
    <fo:block font-style="italic" color="green">  
      <xsl:apply-templates/>  
    </fo:block>  
  </xsl:template>  
</xsl:stylesheet>
```

Output from XSLT engine:

```
<fo:block font-style="italic" color="green">  
  December 3, 1997  
</fo:block>
```

XSLFO = formatting

Input for XSLFO engine:

```
<fo:block font-style="italic" color="green">  
    December 3, 1997  
</fo:block>
```

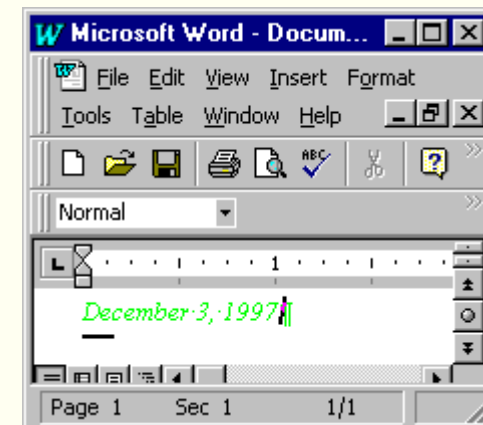
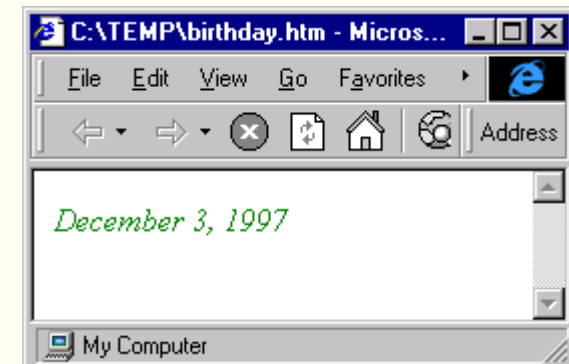
Output:

From XSLFO engine for HTML:

```
<p style="font-style:italic;color:green">  
    December 3, 1997  
</p>
```

From XSLFO engine for RTF:

```
{\c6\f12\i December 3, 1997\par}
```



XSLFO = formatting

- n Character, block, list, page, graphics, numbering, ...
 - with their appropriate formatting characteristics

n Example:

```
<xsl:template match="chapter/title">  
  <fo:block  
    font-family="sans-serif"  
    font-weight="bold"  
    keep-with-next="true"  
    page-break-inside="avoid">  
    <fo:character  
      color="blue">  
      <xsl:apply-templates/>  
    </fo:character>  
  </fo:block>  
</xsl:template>
```


XSL » XSLT

Input:

```
<birthday><date>December 3, 1997</date>  
      <fullname>John Doe</fullname>  
</birthday>
```

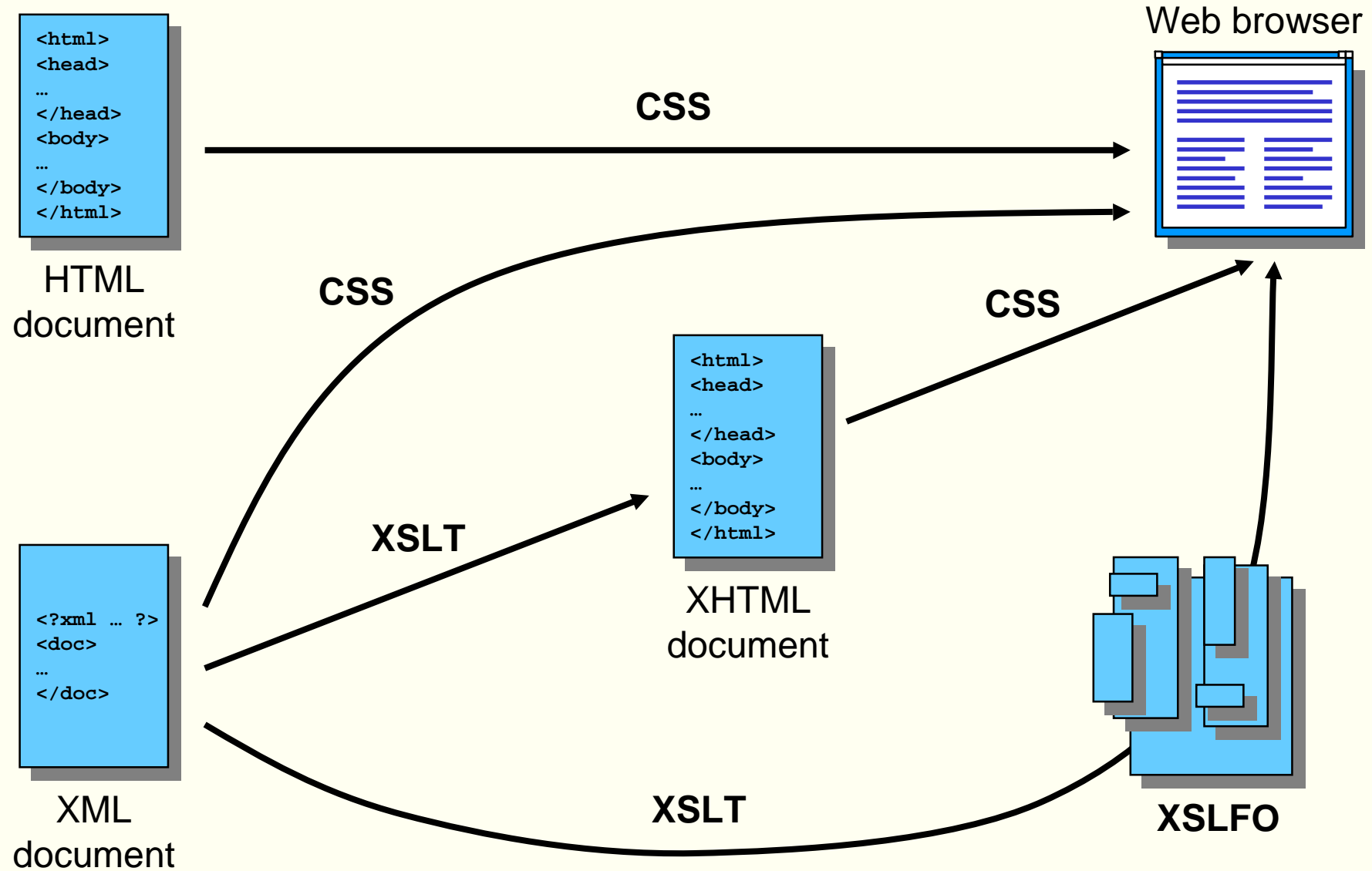
Stylesheet:

```
<xsl:stylesheet xmlns=...>  
  <xsl:template match="birthday/date">  
    <p style="font-style:italic;color:green">  
      <xsl:apply-templates/>  
    </p>  
  </xsl:template>  
</xsl:stylesheet>
```

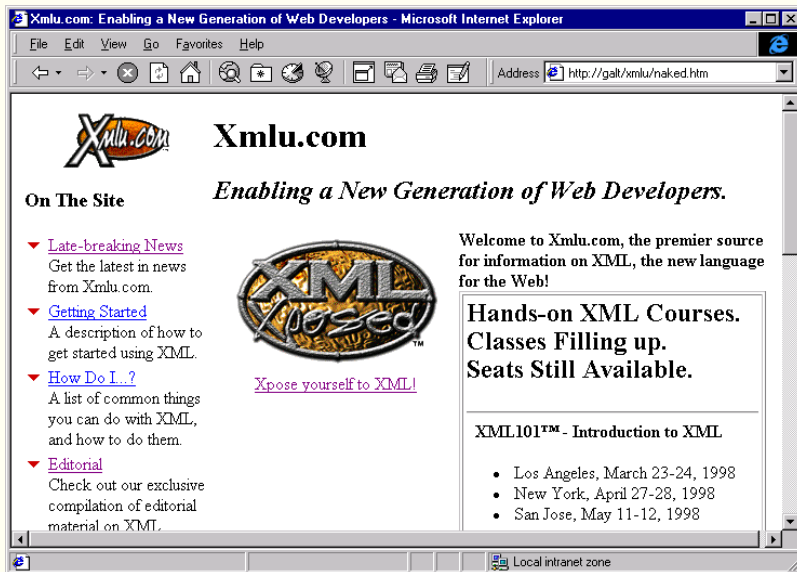
Output from XSLT engine:

```
<p style="font-style:italic;color:green">  
  December 3, 1997  
</p>
```

How to display XML in a Web browser



CSS (Cascading Style Sheets)



```
<style type='text/css'>
  BODY      {font-family:Trebuchet MS,Verdana,Arial;
             color:black;
             background-color:#FFF80}

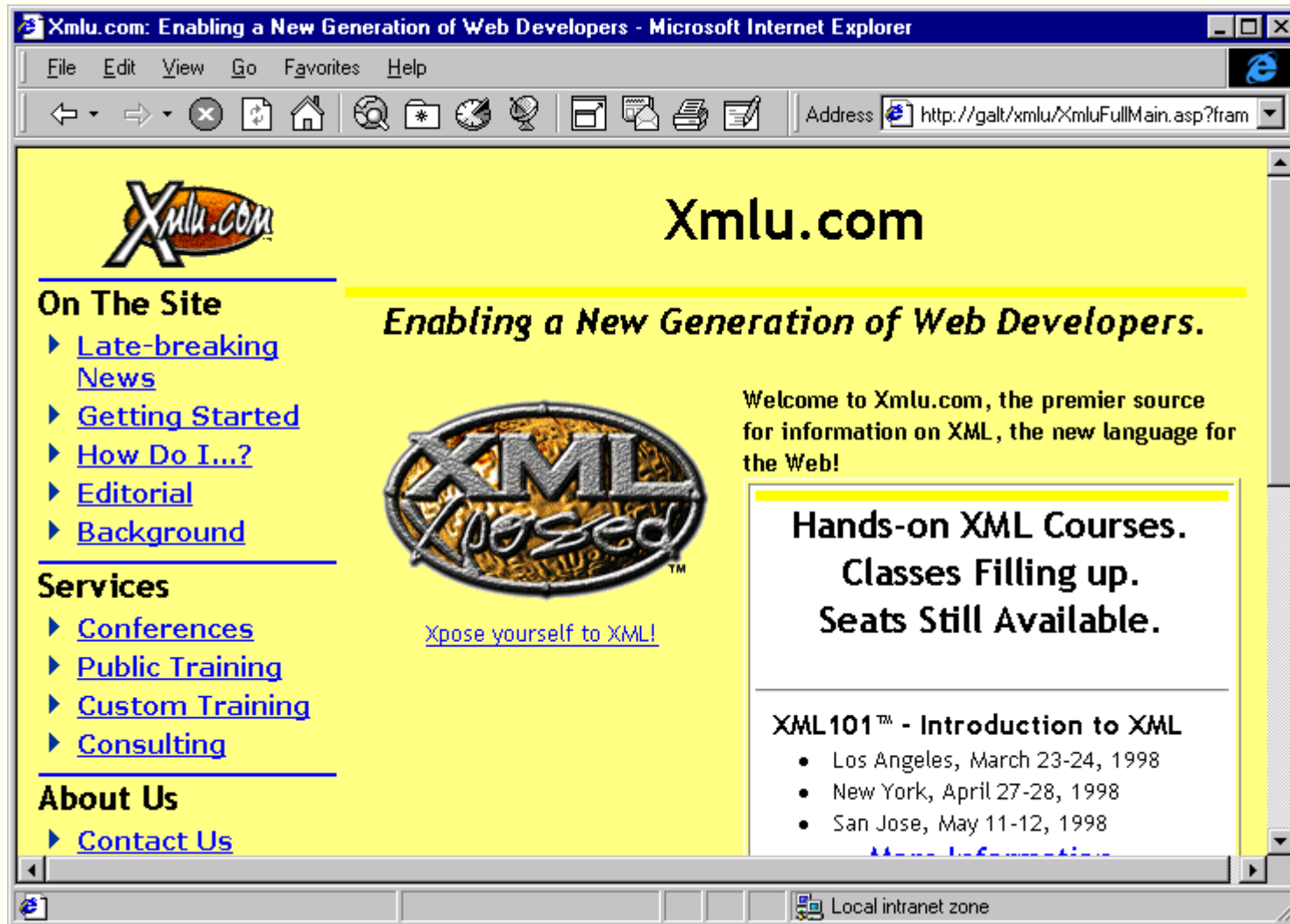
  A:link    {color: blue;}
  A:visited {color: blue;}
  A:hover   {color: red;}
  center    {text-align:center;}
  TD        {font-size:10pt;bottom-margin:0;
             top-margin:0;}

  LI        {margin-left:0}
  H1        {text-align:center;
             font-size:24pt;
             font-weight:bold;}
  H2        {text-align:center;
             font-size:16pt;
             font-weight:bold;
             border-top:thick yellow solid;}
  H3        {font-size:14pt;
             font-weight:bold;
             border-top:thin blue solid;
             margin-bottom:0;
             margin-top:5}

</style>
```



CSS (Cascading Style Sheets)



CSS vs. XSL

n CSS (Cascading Style Sheets)

- one-to-one mapping between an XML element and a built-in browser formatting object
"sprinkling presentation hints on top of XML"
- structure of output formatted document closely follows structure of input XML document
- client-side only styling

n XSL (Extensible Stylesheet Language)

- XML element mapped to a formatting object consisting of multiple built-in formatting objects
"defining detailed presentation using XML"
- structure of output formatted document is not constrained by structure of input XML document
- client-side and server-side styling

XLL: Extensible Linking Language

n XML linking mechanism: **XLink** and **XPointer** (using XPath)

- subset of HyTime and TEI (Text Encoding Initiative)
- compatible with existing URL linking
- additional functionality:
 - any element can be a link
 - links can be bi-directional
 - links can be indirect

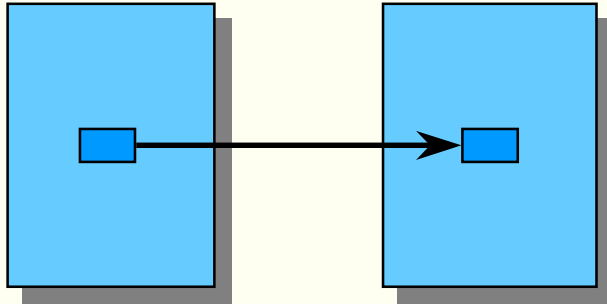
n Useful for:

- rich hypertext functionality
- Web site management

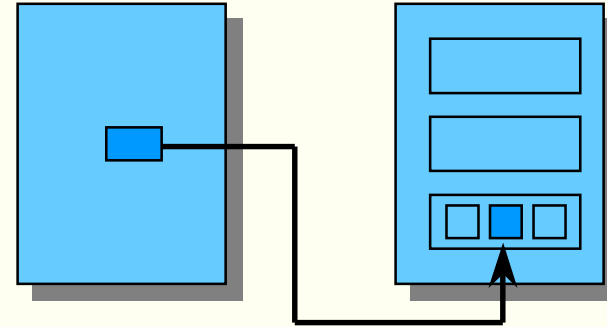
n Status:

- XLink: W3C Proposed Recommendation (December 20, 2000)
- XPointer: W3C Last Call Working Draft (January 8, 2001)

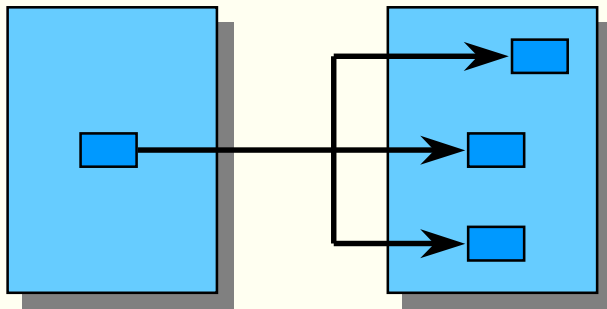
XLink: XML Linking Language



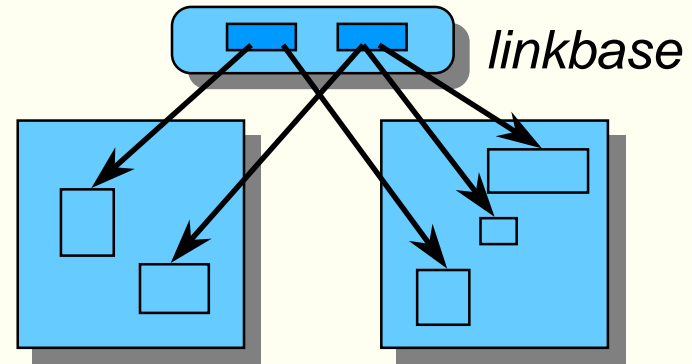
simple link



simple link (symbolic)



inline extended link



out-of-line extended link

Example of an extended link

The image shows a sequence of events in an XML browser application. The main window, titled "Scheduled Flights - JFK - XML Browser", displays a table of flight options from London (LHR) to New York (JFK) on Sunday, July 4, 1999. A context menu is open over the 10:00 am flight, with options like "Show remaining seats", "Book flight", "Show fare restrictions", and "Enter new itinerary". A red dashed arrow points from the "Book flight" option to a "Flight Confirmation" dialog box. The dialog asks for confirmation to enter the reservation, with "Yes", "No", and "Cancel" buttons. A yellow dashed arrow points from the "Show fare restrictions" option to a "Fare restrictions" dialog box. The restrictions include: "Must stay over a Saturday night", "Tickets must be purchased within 24 hours of reservation and not less than 7 days prior to flight", and "Tickets are nonrefundable. Changes to itinerary will result in \$75 fee and payment of difference in fare." A blue dashed arrow points from the 2:00 pm flight to a "Softland Airlines Flight Finder" page. This page features the airline logo, a "Book a flight" section with input fields for departure/return dates and times, and a "More" button.

Time	Date	Day	Duration	Origin	Destination	Airline	Class
8:00 am	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	115
8:45 am	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	118
8:55 am	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	120
10:00 am	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	118
10:55 am	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	121
12:00 pm	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	119
1:15 pm	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	117
1:55 pm	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	122
2:00 pm	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	125
2:00 pm	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	127
2:05 pm	7/4/99	Sun	7h 55m	London(LHR)	New York(JFK)	Softland Airlines	129

insert an image

run a small program

show hidden text

link to another page

XPointer: XML Pointer Language

- n Addressing scheme for pointing into an XML document
 - identifying locations in an XML document to be jumped to
 - identifying parts of an XML document to be embedded

n Syntax:

xpointer(*an XPath expression*)

to be used in an URL:

document.xml#xpointer(...)

- loads the whole document and jumps to the location being pointed to

document.xml|xpointer(...)

- only loads the part of the document being pointed to

n Example:

xlink:href="document.xml#xpointer(/section[3]/para[2])"

DOM: Document Object Model

n Standard API for XML documents

- **Level 1**: access to document contents
- **Level 2**: access to document stylesheet + mouse events + traversal
- **Level 3**: loading/saving/serializing documents + presentation views + keyboard & device independent events
- advanced capabilities:
 - expanding/collapsing text & dynamic tables of contents
 - client-side active documents

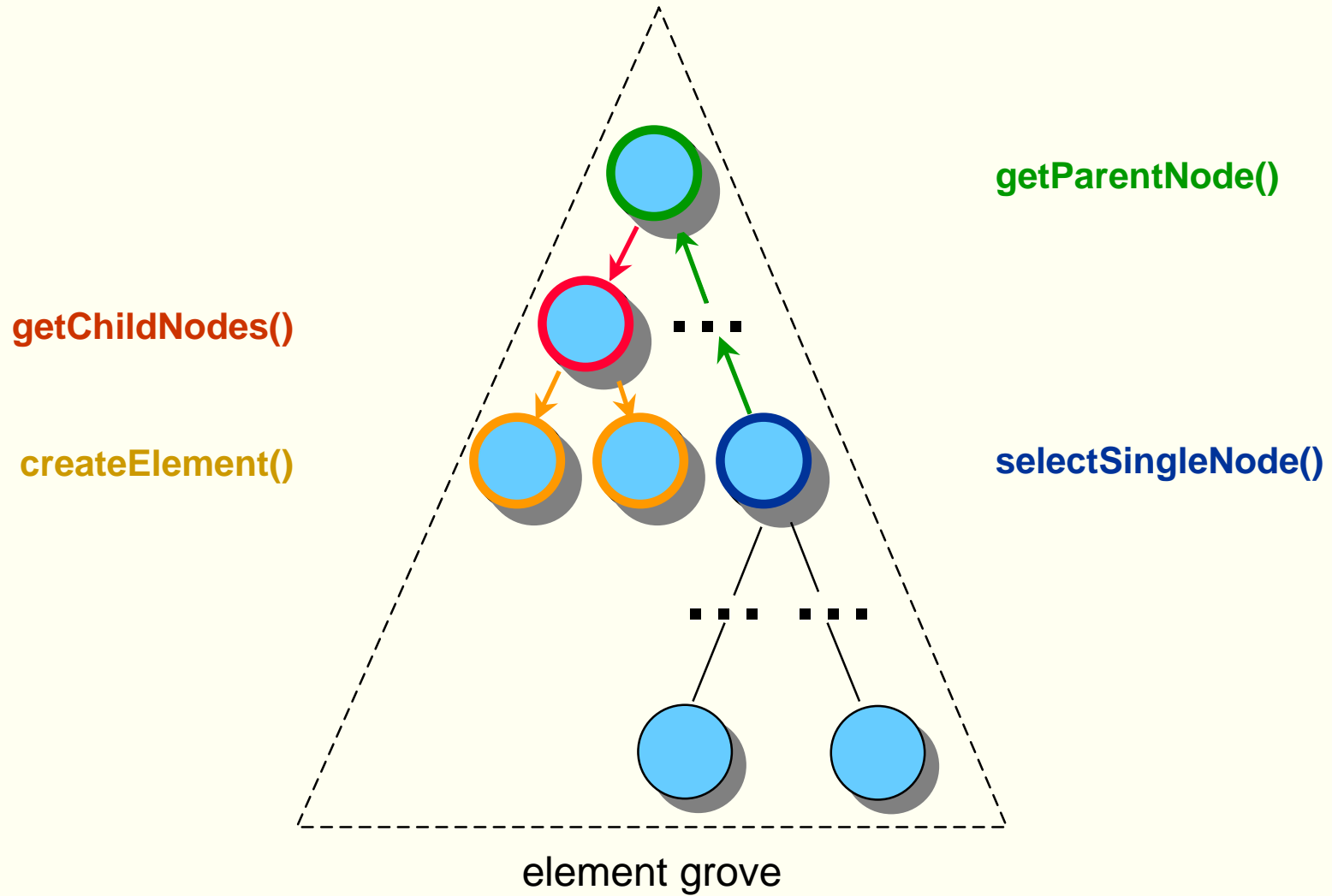
n Useful for:

- editing XML documents and interacting with XML data structures

n Status:

- Level 1: W3C Recommendation (October 1, 1998)
- Level 2: W3C Recommendation (November 13, 2000)
- Level 3: W3C Working Draft (February 9, 2001)

Example of a DOM tree manipulation



DOM: key concepts

n Document is a tree of objects

- Document, Node, NodeList, NamedNodeMap, ...
- Element, Attribute, Text, Comment, Entity, ...

n Language-independent standard API for navigating, querying and modifying these objects

- navigating: getParentNode(), getChildNodes(), ...
- querying: selectSingleNode(), selectNodes(), ...
- modifying: createElement(), createAttribute(), setAttribute(), ...

to be used in JavaScript, VBScript, ...

n Alternative: **SAX** (Simple API for XML)

- for event-based parsing instead of building a parse tree
- better for handling large XML documents, but it is harder to manipulate the structure of the XML documents

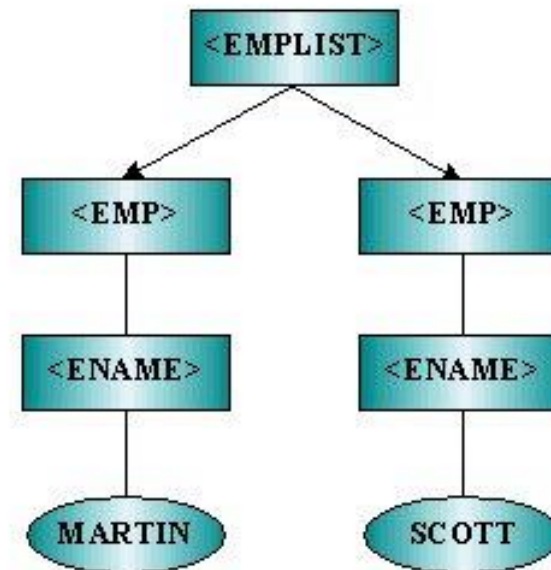
SAX vs. DOM processing

```
<?xml version="1.0"?>
  <EMPLIST>
    <EMP>
      <ENAME>MARTIN</ENAME>
    </EMP>
    <EMP>
      <ENAME>SCOTT</ENAME>
    </EMP>
  </EMPLIST>
```

SAX

```
start document
start element: EMPLIST
start element: EMP
start element: ENAME
characters: MARTIN
end element: EMP
start element: EMP
start element: ENAME
characters: SCOTT
end element: EMP
end element: EMPLIST
end document
```

DOM



SAX vs. DOM processing

n Event-based API: SAX (**S**imple **A**PI for **X**ML)

- uses *callbacks* to report parsing events to the application
- application deals with these events through customized *event handlers*
- è use for: - batch processing, extracting from large volumes of data
 - when the parse tree does not have to be manipulated

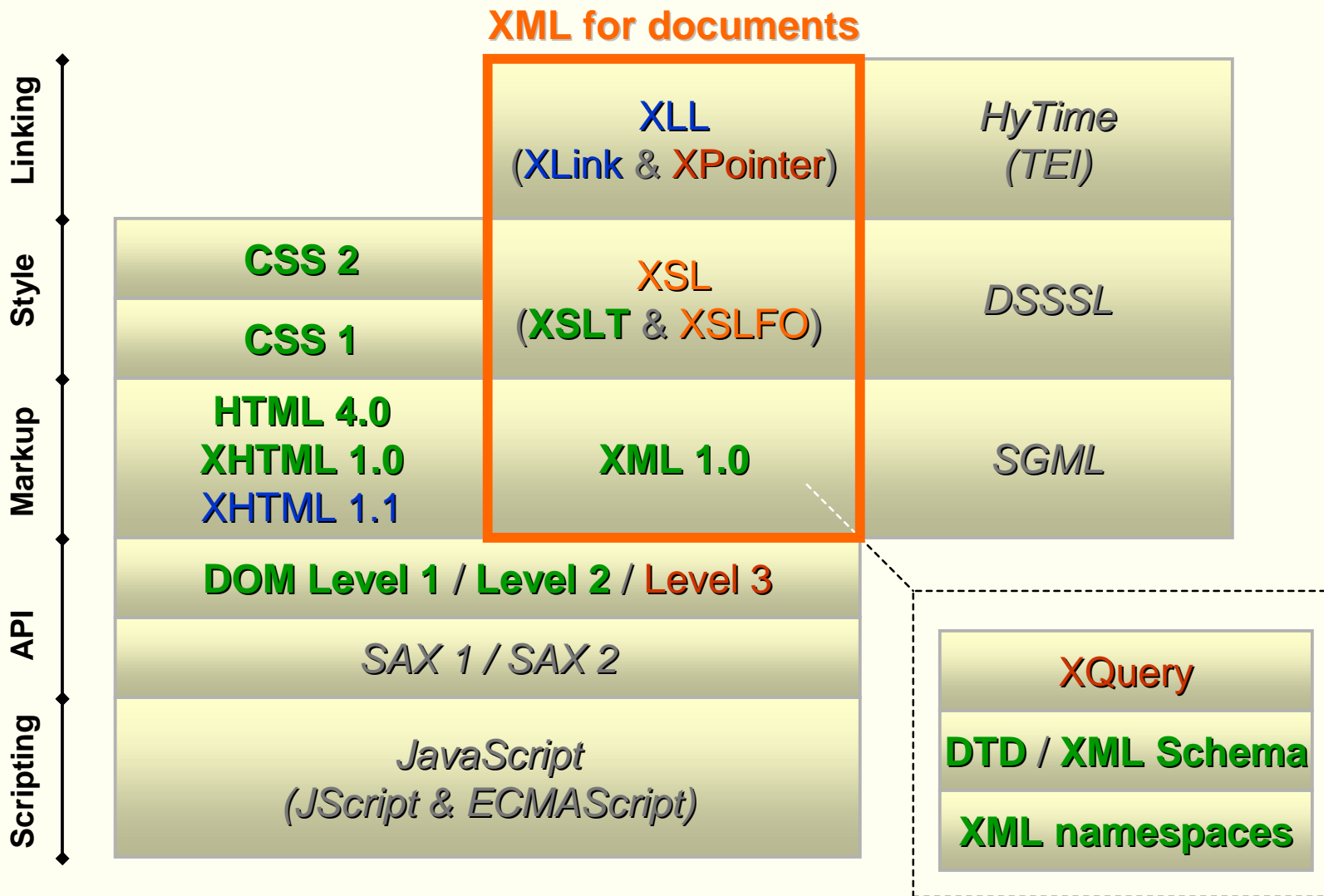
n Tree-based API: DOM (**D**ocument **O**bject **M**odel)

- provides *objects* and *methods* to be used by the application
- application uses these methods to navigate and manipulate the tree
- è use for: - interactive processing, modifying small volumes of data
 - adding / modifying / deleting elements and attributes

n SAX is faster, better suited for data (development-intensive)

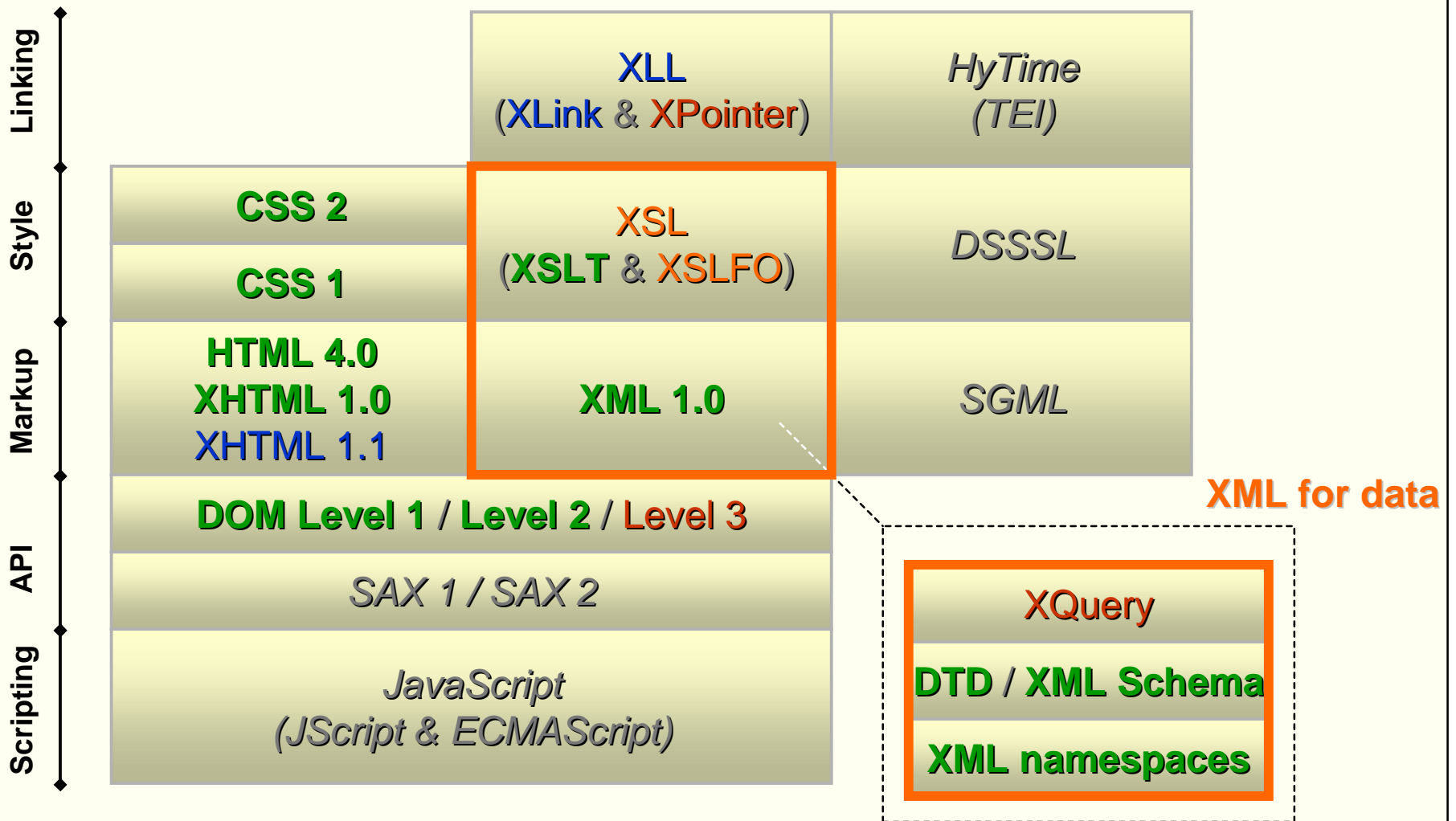
DOM is more flexible, better suited for documents (memory-intensive)

The XML standards family



W3C: [Note](#) > [Work. Draft](#) > [Cand. Recom.](#) > [Prop. Recom.](#) > [Recommendation](#) / *Not W3C*

The XML standards family



W3C: Note > Work. Draft > Cand. Recom. > Prop. Recom. > Recommendation / Not W3C

Conclusions

n XML for documents standards:

- XML: standard completed, commercial tools
- XSL:
 - XSLT: standard completed, commercial tools
 - XSLFO: standard being refined, prototype tools
- XLL:
 - XLink: standard almost completed, prototype tools
 - XPointer: standard just started, experimental tools



n XML for documents is as good as ready for industrial-strength, mission-critical usage

- web site content management
- electronic document management

... so it's already being widely used by developers

Conclusions

n XML for data standards:

- XML: standard completed, commercial tools
- XSL:
 - XSLT: standard completed, commercial tools
 - XSLFO: standard being refined, prototype tools
- XML namespaces: standard completed, commercial tools
- XML Schema: standard just completed, prototype tools
- XQuery: standard just started, commercial tools only support XQL



n XML for data is not completely ready for industrial-strength, mission-critical usage

- enterprise application integration
- business-to-business e-commerce

... but it's being used in commercial tools and solutions anyway