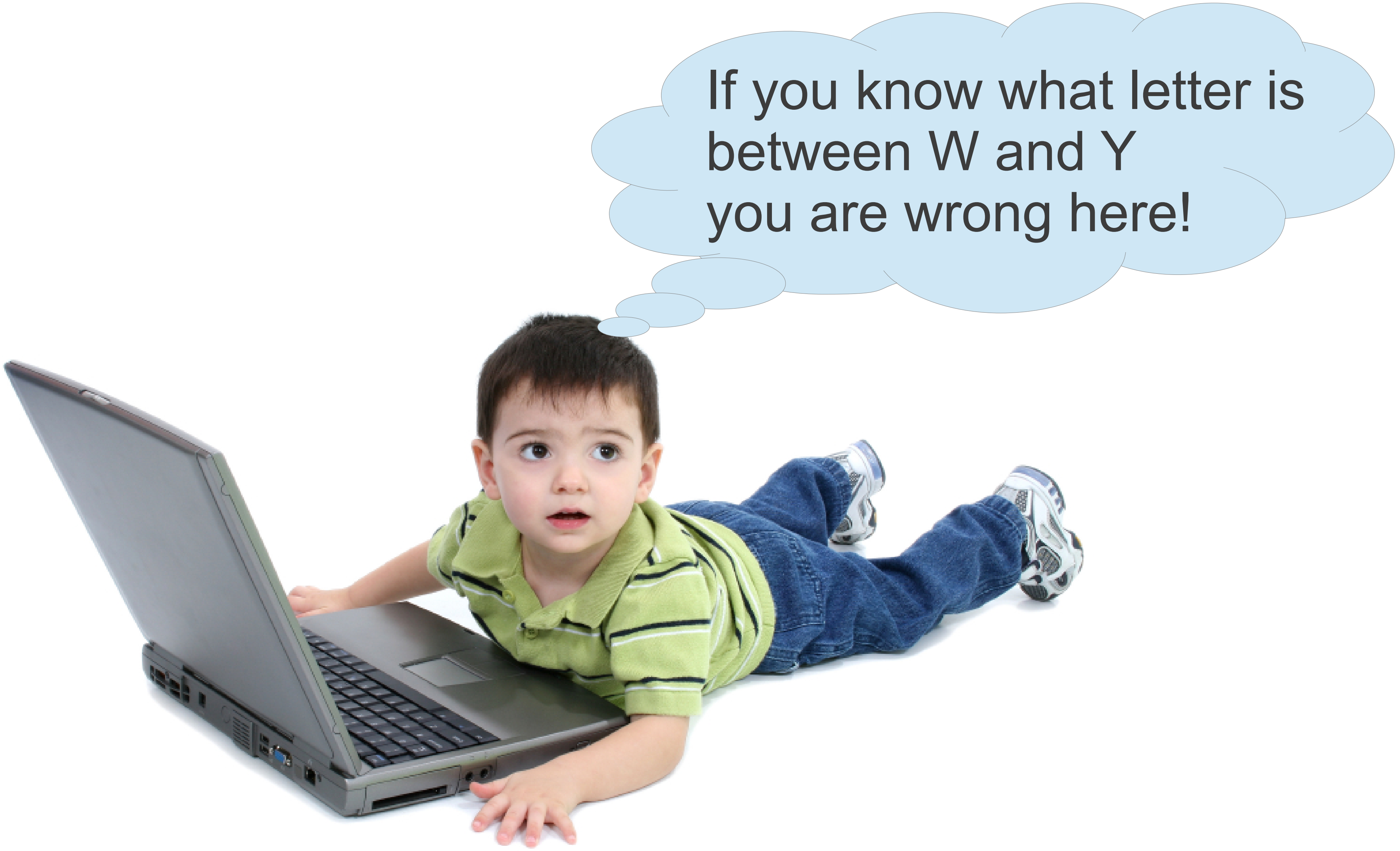
XML

What you didn't know that you wanted to know...

... or maybe you did, and just have a good time

Foudation Class

If you know what letter is between W and Y
you are wrong here!



About me

- ~~Lotus~~ IBM Notes since V2.x
- Studied Law & Economics
- Counsellor for person centric development
- Work for IBM Singapore
- @NotesSensei
- 我说中国话一点

Books harmed for this presentation

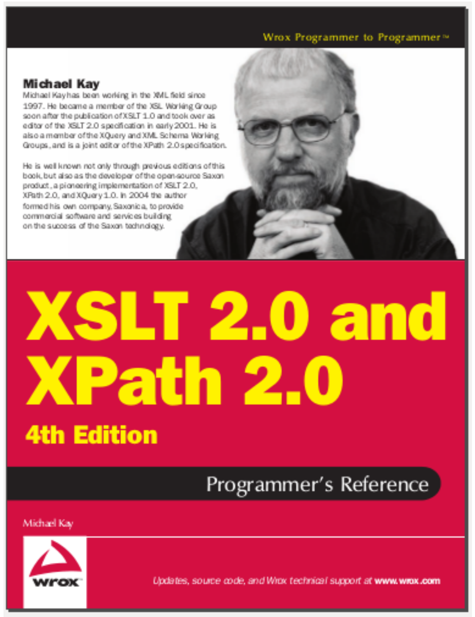
Join the discussion @ p2p.wrox.com

FIFTH EDITION

Beginning XML

Joe Fawcett, Liam R. E. Quin, Danny Ayers

868 pages



Wrox Programmer to Programmer™

Michael Kay
Michael Kay has been working in the XML field since 1997. He became a member of the XSL Working Group soon after the publication of XSLT 1.0 and took over as editor of the XSLT 2.0 specification in early 2001. He is also a member of the XQuery and XML Schema Working Groups, and is a joint editor of the XPath 2.0 specification.

He is well known not only through previous editions of this book, but also as the developer of the open-source Saxon product, a pioneering implementation of XSLT 2.0, XPath 2.0, and XQuery 1.0. In 2004 the author formed his own company, Saxonica, to provide commercial software and services building on the success of the Saxon technology.

XSLT 2.0 and XPath 2.0

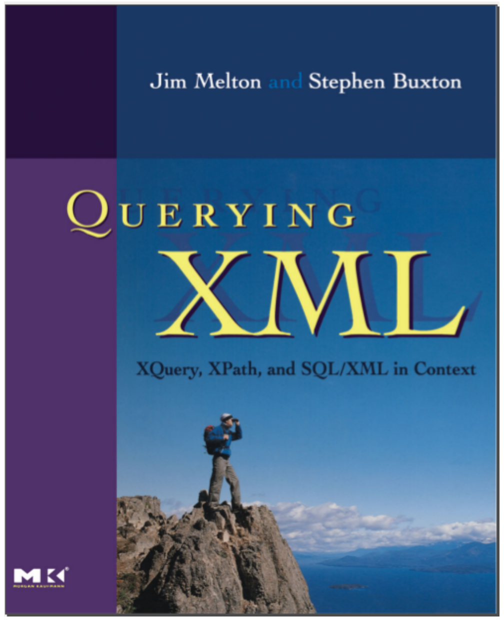
4th Edition

Programmer's Reference

Michael Kay

Updates, source code, and Wrox technical support at www.wrox.com

1371 pages

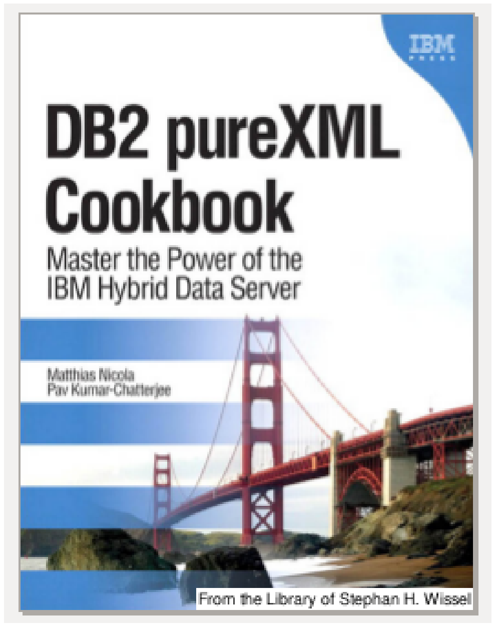


Jim Melton and Stephen Buxton

QUERYING XML

XQuery, XPath, and SQL/XML in Context

845 pages



IBM PRESS

DB2 pureXML Cookbook

Master the Power of the IBM Hybrid Data Server

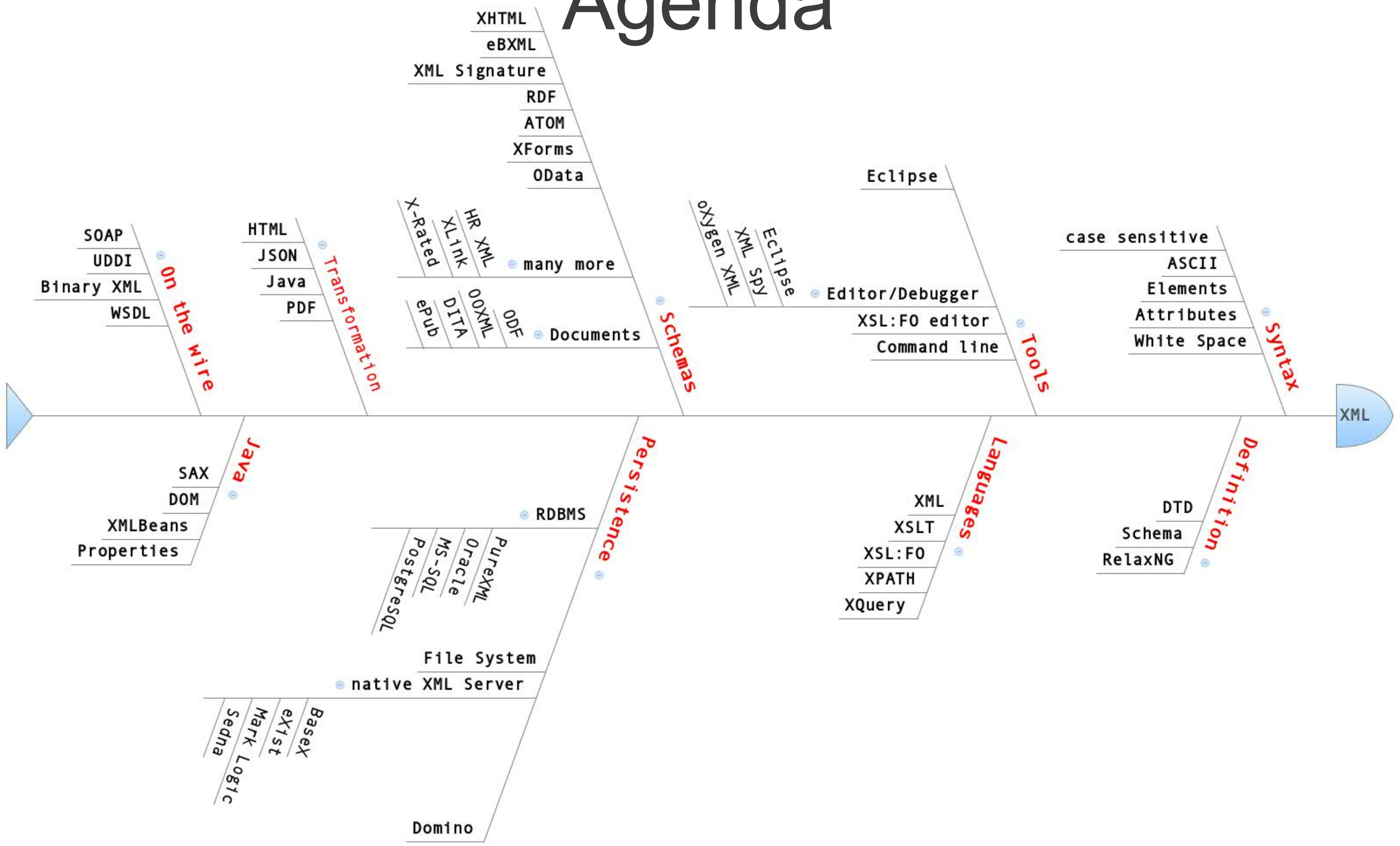
Matthias Nicola
Pav Kumar-Chatterjee

From the Library of Stephan H. Wissel

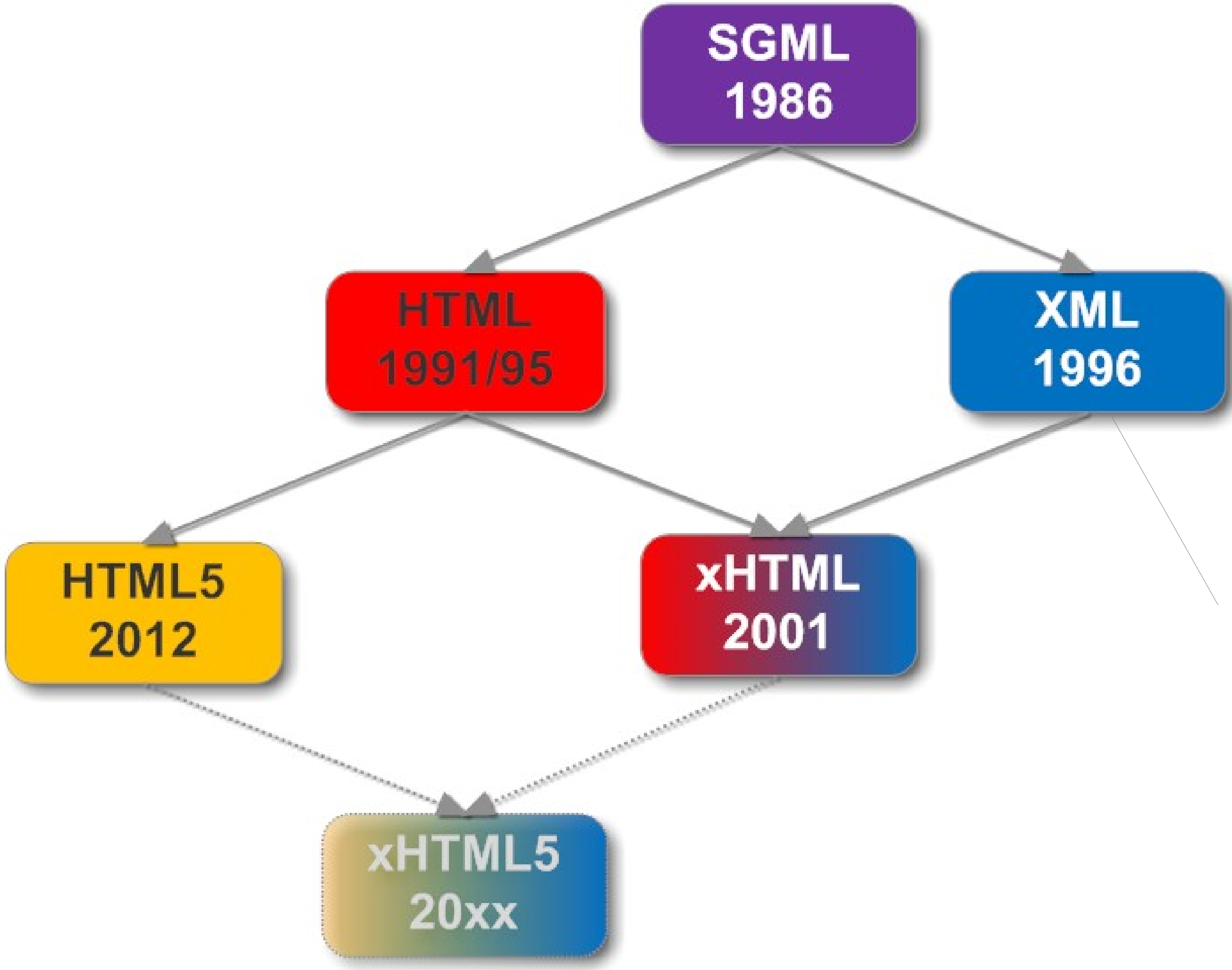
793 pages

Agenda

XML

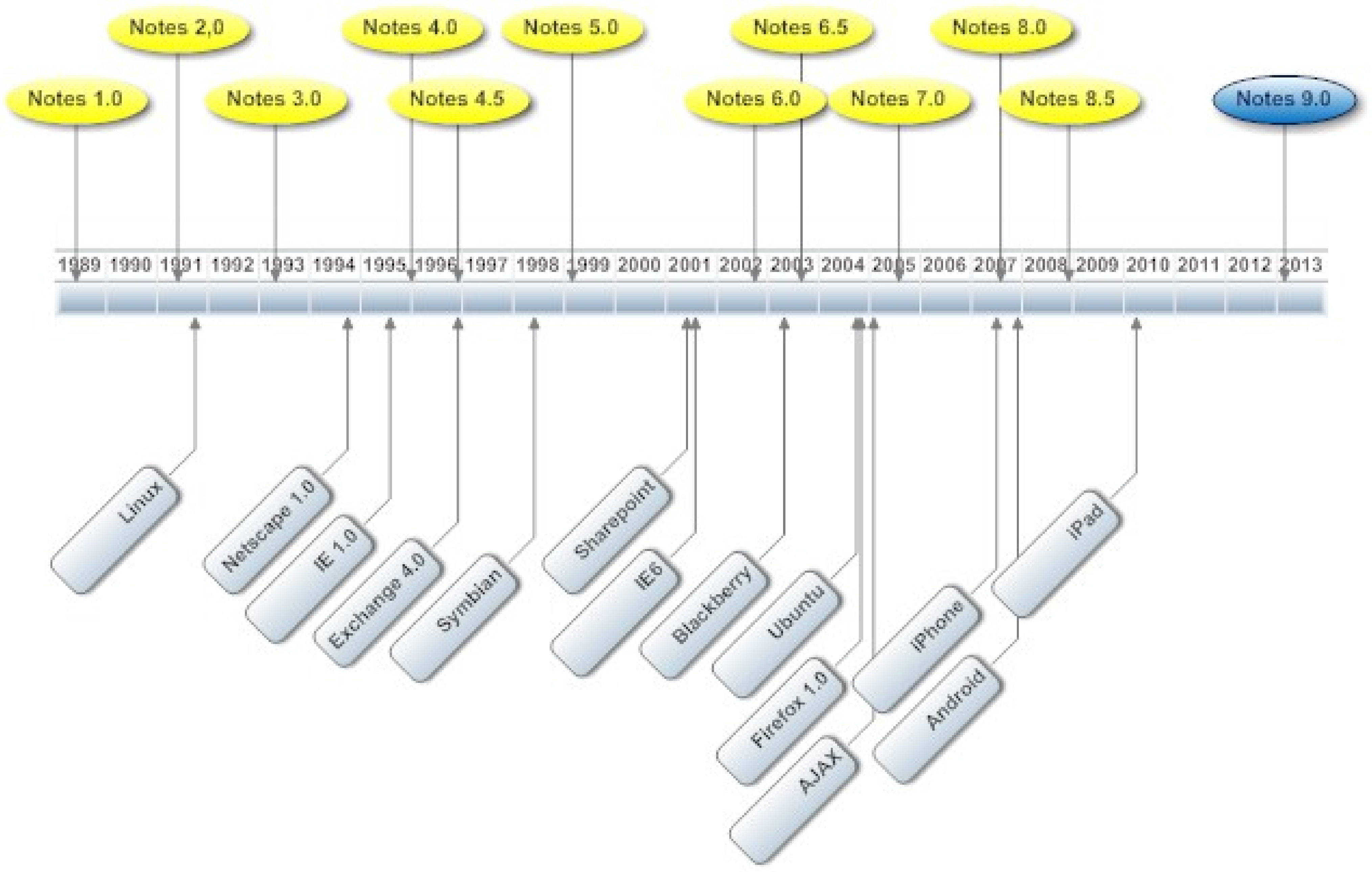

XML
Syntax
case sensitive
ASCII
Elements
Attributes
White Space
Definition
DTD
Schema
RelaxNG
Tools
Eclipse
Editor/Debugger
Eclipse
XML Spy
oXygen XML
XSL:FO editor
Command line
Languages
XML
XSLT
XSL:FO
XPATH
XQuery
Schemas
XHTML
eBXML
XML Signature
RDF
ATOM
XForms
OData
many more
HR XML
XLink
X-Rated
Documents
ODF
OOXML
DITA
ePub
Persistence
RDBMS
PureXML
Oracle
MS-SQL
PostgreSQL
File System
native XML Server
BaseX
eXist
Mark Logic
Sedna
Domino
Transformation
HTML
JSON
Java
PDF
Java
SAX
DOM
XMLBeans
Properties
On the wire
SOAP
UDDI
Binary XML
WSDL

History, Format & Standards



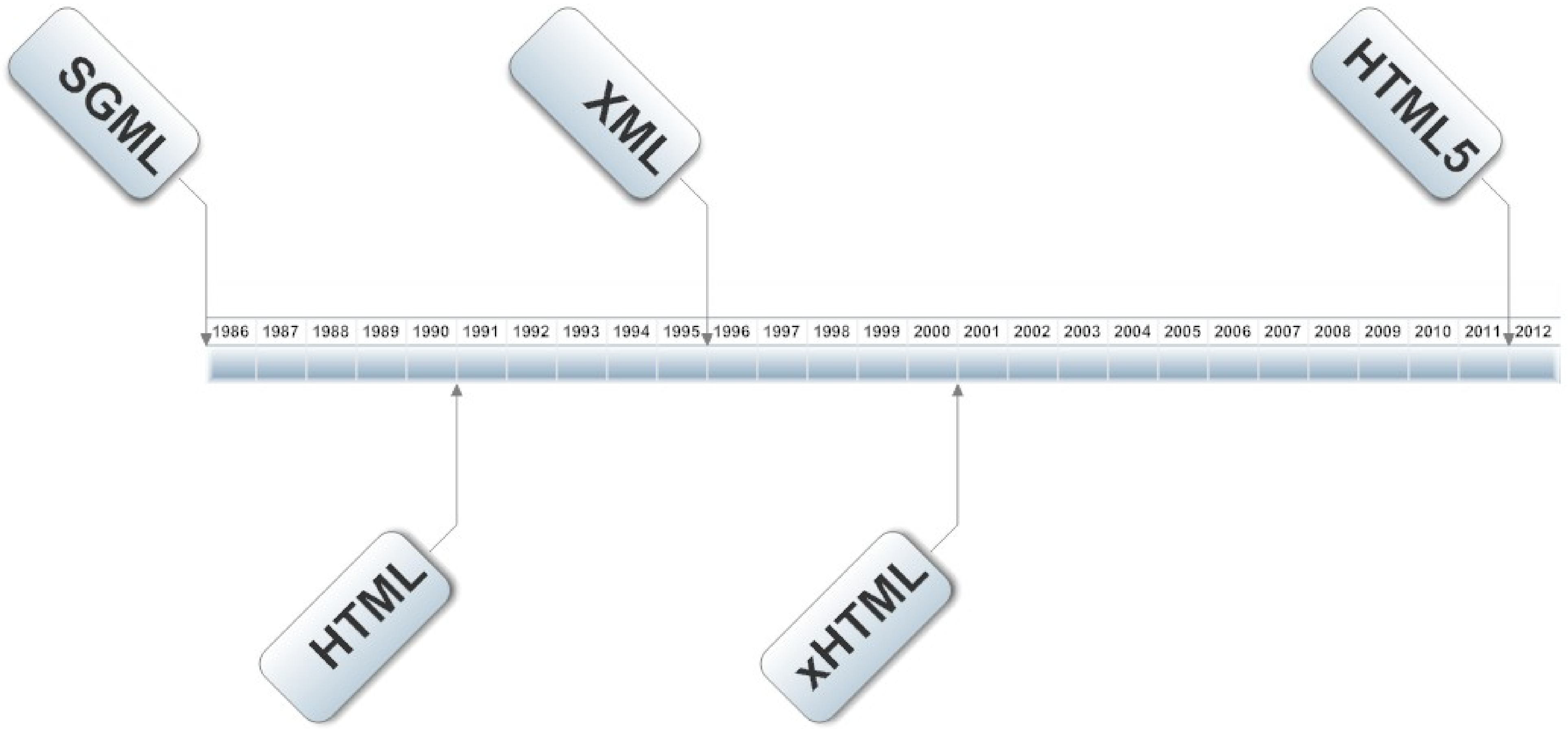
Here is the fun!
Tons of DSL!!!!

Timelines


Notes 1.0
Notes 2,0
Notes 3.0
Notes 4.0
Notes 4.5
Notes 5.0
Notes 6.0
Notes 6.5
Notes 7.0
Notes 8.0
Notes 8.5
Notes 9.0
1989 1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 2010 2011 2012 2013
Linux
Netscape 1.0
IE 1.0
Exchange 4.0
Symbian
Sharepoint
IE6
Blackberry
Ubuntu
Firefox 1.0
AJAX
iPhone
Android
iPad

XML

Timelines


SGML
XML
HTML5
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
HTML
xHTML

X 18+

RESTRICTED

Restricted to 18 and over

Contains naked code!

Syntax

```
<?xml version=”1.0”?>
<root>
     <stuff>
        <morestuff id=”some id”>
               <evenmorestuff />
         </morestuff>
      </stuff>
     <stuff> Some text <bla /></stuff>
      <otherstuff> Some fancy Text </otherstuff>
      <!-- Witty comment -->
</root>
```

XML Declaration (optional, recommended)

Root element (there can only be one!)

Element

Attribute

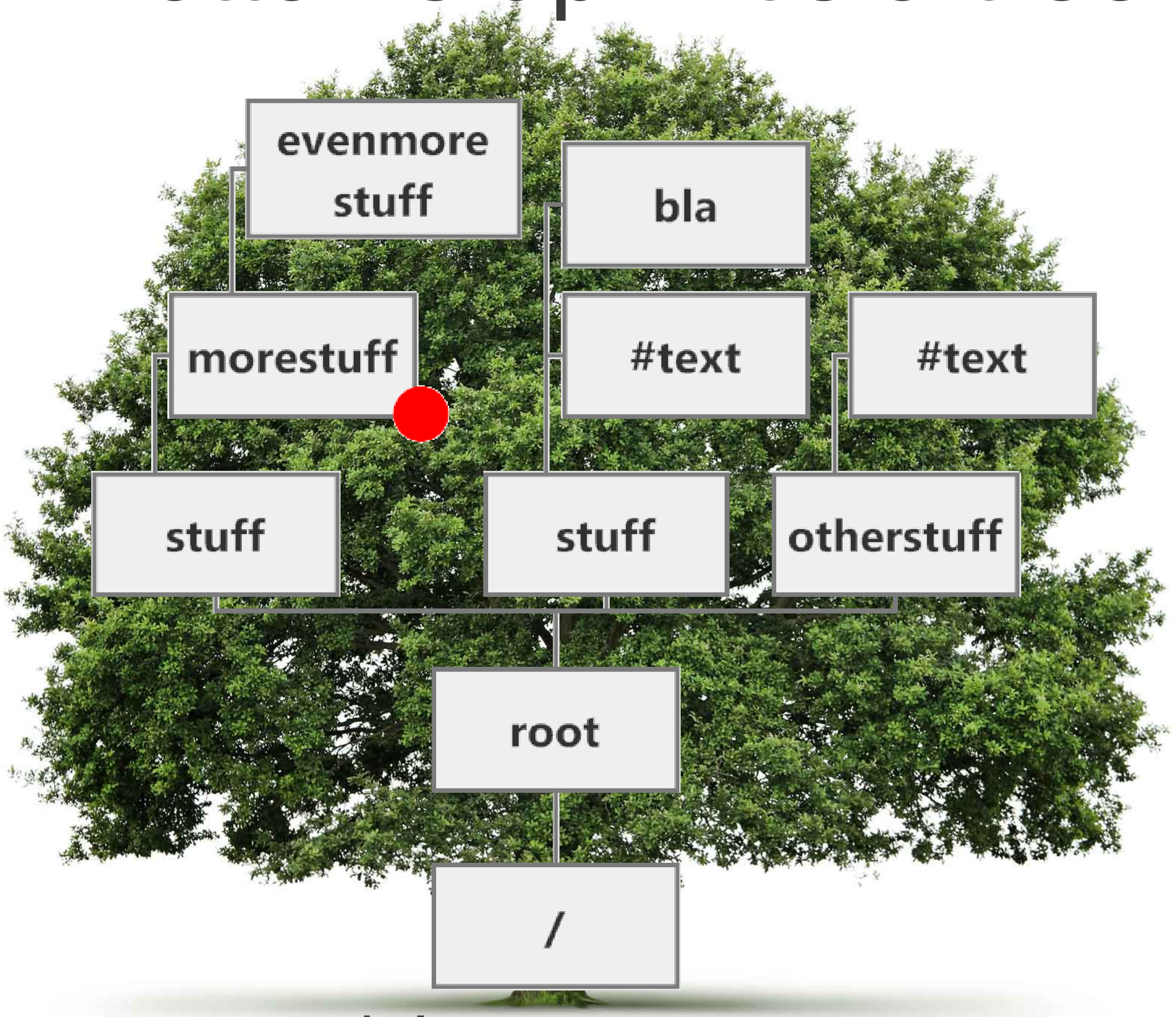
Empty Element

Text Node

Comment

What you open, you must close

Bottoms up – it's a tree!



树 (Shù)

Syntax

- One root element only
- Elements must be closed
  - <element></element>
  - <element />
- Must not start with xml (in any case)
- Case sensitive
- No spaces
- White space neutral
- Attribute sequence must not matter

Syntax Bloopers

Don't try this at home!

- <eleMENT></ELEment>
- <element att1=”something” att1=”something” />
- <element att1=something />
- <e1><e2>Some Text<e3></e2></e3></e1>
- <e1> a message </e1>
  <e1> a message </e1>
- <fancy element>stuff</fancy element>
- < 小老虎 > 跑快 </ 小老虎 >

NameSpaces

- Bank -

bank
Namespace:
Money & Finance

bank
Namespace:
Nature & Geography

bank
Namespace:
Aeronautics

NameSpaces*

Can be made up (just like news)

- For each element separately
  <bla xmlns=”http://www.foxnews.com/bias” >
  Debt is good for you</bla>

- At the root element with alias
  <news xmlns=”http://thetruth.org”
  xmlns:fox=”http://www.foxnews.com/bias” >
  <topic>Aliens are with us</topic>
  <fox:bla>Climate change is humbug</fox:bla>
  </news>

* more on popular NameSpaces later

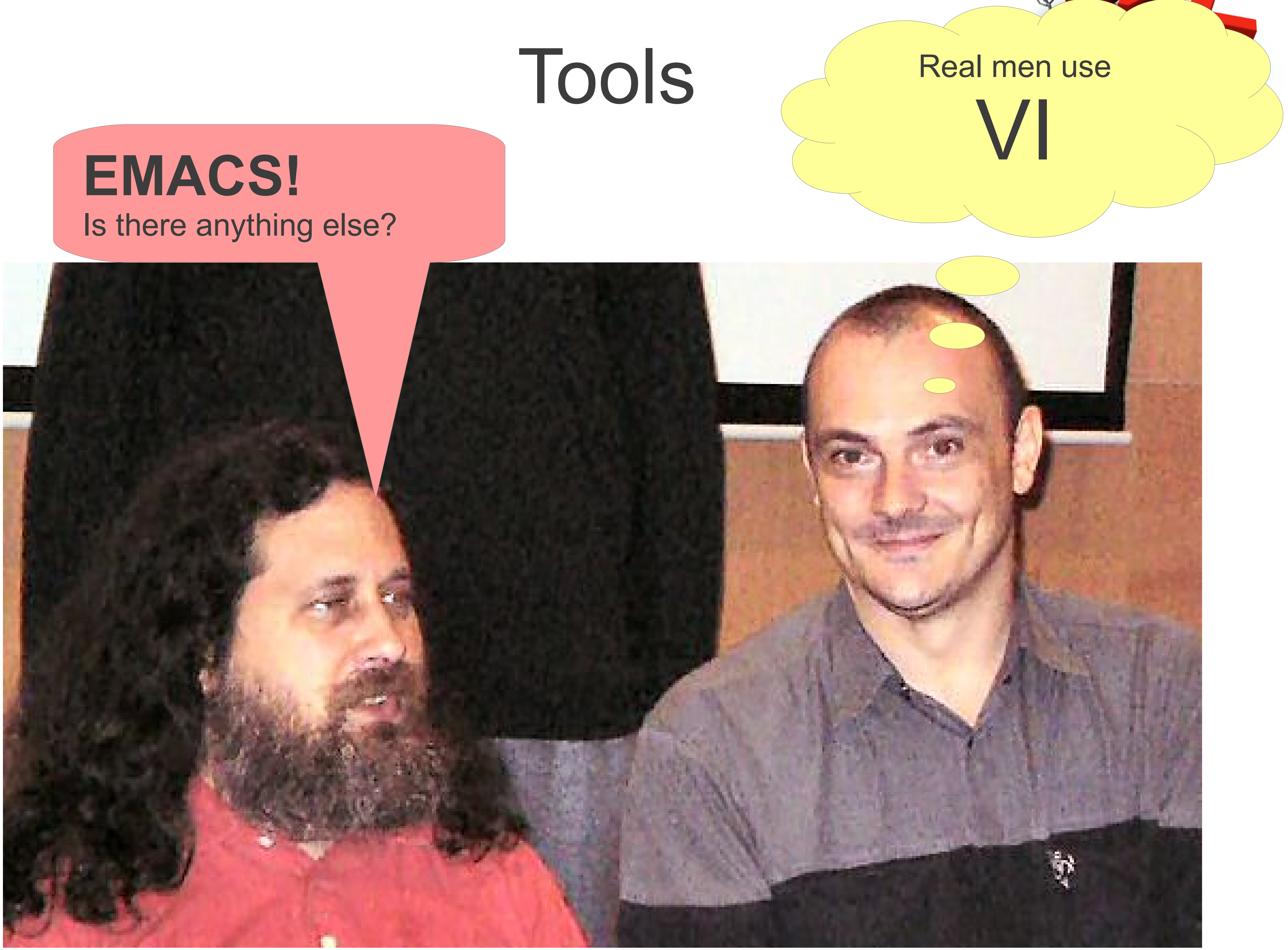
XML & JSON*

```
<book isbn="1234">
    <rdf:author>Peter
    </rdf:author>
    <publisher id="221">
        Random House
    </publisher>
    <synopsis>
<![CDATA[
<h1>Hillarious</h1>
<p>It is "funny"</p>

]]>
</book>
```

```
{ "isbn" : "1234",
  "rdfAuthor" : "Peter",
  "publisher" : {

      "id" : "221",
      "name" :
       "Random House"},
    "synopsis" : "<h1>
     Hillarious</h1><p>
     It is \"funny\"</p>"
}
```

* more on the how -> later

Tools



Tools

- A syntax aware editor
  (Geany, Sublime, TextPad++)
- A general purpose IDE
  (Eclipse, IntelliJ, Visual Studio, etc)
- A specialized XML IDE with debugger
  - XML Spy
  - Oxygen XML (that's what I use)
  - Stylus Studio
- A decent browser
- FOP Editor:
  http://www.java4less.com/fopdesigner/fodesigner.php

Notepad is **NOT** on this list!

Also as plug-in
For the general
purpose IDEs

Command Line Tools

- put
```
#!/bin/bash
curl $1  -X PUT --netrc --basic -k -v -L -T $2 -o
$3 $4 $5 $6 $7
```
- get
```
#!/bin/bash
curl $1 --netrc -G --basic -v -k -L -o $2 $3 $4 $5
$6 $7
```
- .netrc
```
machine server1.acme.com login road password runner
machine demo.mybox.local login carl password coyote
```

Command Line Tools II

- xslt
  ```
  #!/bin/bash
  java -cp /home/stw/bin/saxon9he.jar
  net.sf.saxon.Transform -t -s:$1 -xsl:$2 -o:$3
  ```
- fop `-xml foo.xml -xsl foo.xsl -pdf foo.pdf`
- unid
  ```
  #!/bin/bash
  java -cp /home/stw/bin MakeUNID
  ```

```
import java.util.UUID;
public class MakeUNID {
        public static void main(String[] args) {
                System.out.println(UUID.randomUUID().toString());
                 System.exit(0);
        }
}
```

Schema & DTD

- Multiple Standards available
  - Document Type Definition
  - XML Schema
  - RelaxNG
  - Schematron

Defined in XML!

- Define content structure
- Used by validating parsers
- IMHO most confusing part

DTD

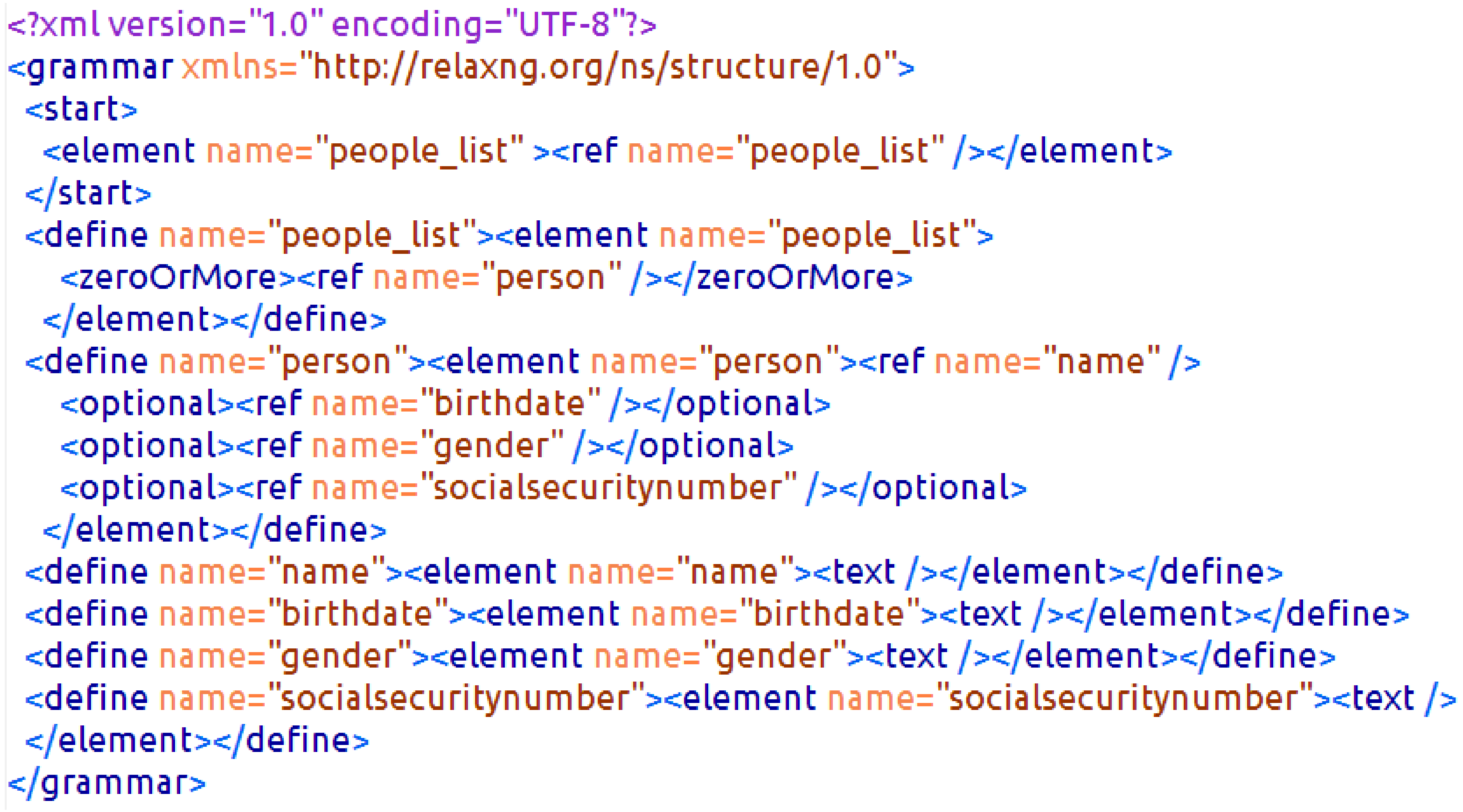
```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE people_list [
  <!ELEMENT people_list (person*)>
  <!ELEMENT person (name, birthdate?, gender?, socialsecuritynumber?)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT birthdate (#PCDATA)>
  <!ELEMENT gender (#PCDATA)>
  <!ELEMENT socialsecuritynumber (#PCDATA)>
]>
<people_list>
  <person>
    <name>Fred Bloggs</name>
    <birthdate>2008-11-27</birthdate>
    <gender>Male</gender>
  </person>
</people_list>
```

en.wikipedia.org/wiki/Document_type_definition

Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="people_list">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" maxOccurs="unbounded" ref="person"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="person">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="name"/>
        <xs:element minOccurs="0" ref="birthdate"/>
        <xs:element minOccurs="0" ref="gender"/>
        <xs:element minOccurs="0" ref="socialsecuritynumber"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="name" type="xs:string"/>
  <xs:element name="birthdate" type="xs:string"/>
  <xs:element name="gender" type="xs:string"/>
  <xs:element name="socialsecuritynumber" type="xs:string"/>
</xs:schema>
```

RelaxNG



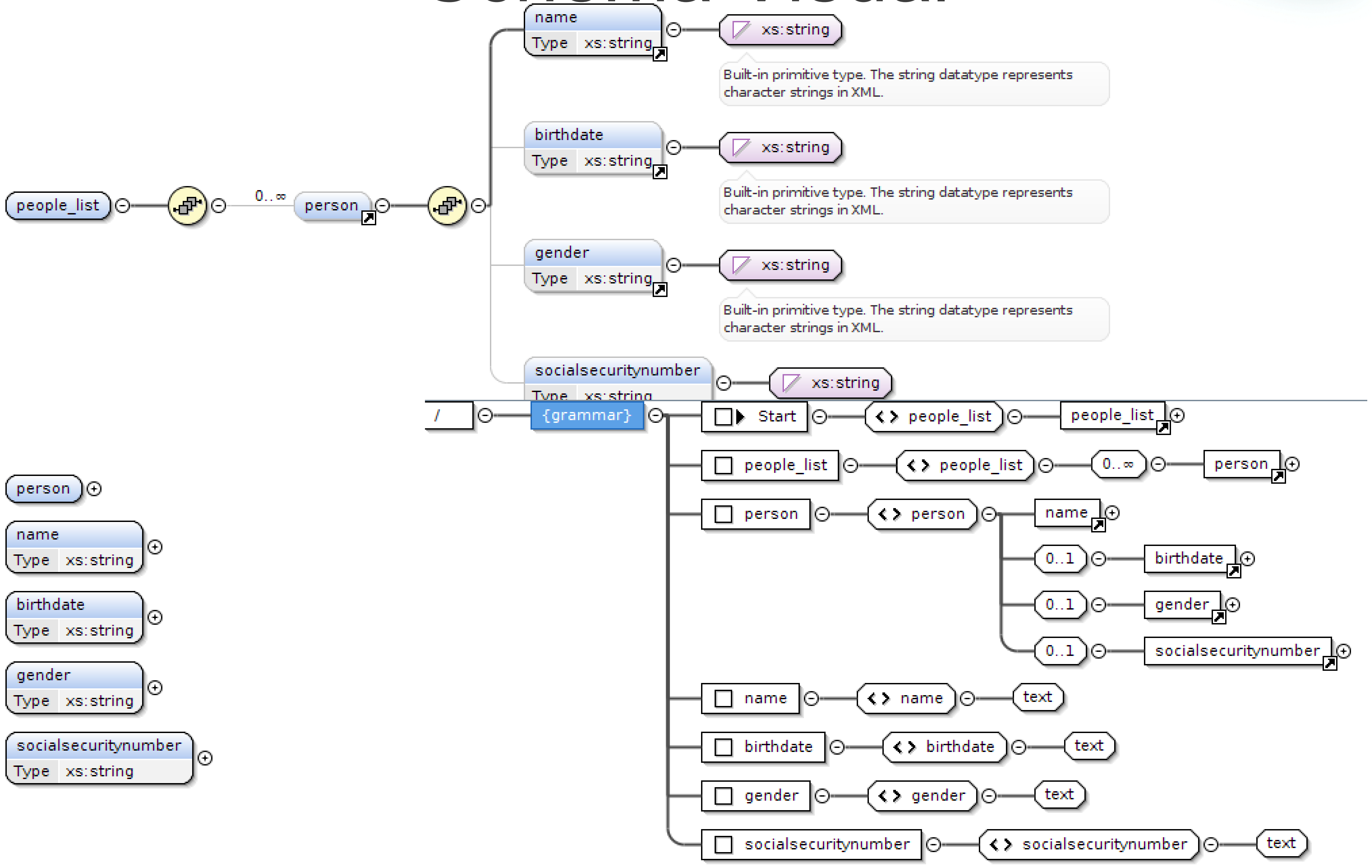
```
<?xml version="1.0" encoding="UTF-8"?>
<grammar xmlns="http://relaxng.org/ns/structure/1.0">
 <start>
  <element name="people_list" ><ref name="people_list" /></element>
 </start>
 <define name="people_list"><element name="people_list">
    <zeroOrMore><ref name="person" /></zeroOrMore>
  </element></define>
 <define name="person"><element name="person"><ref name="name" />
    <optional><ref name="birthdate" /></optional>
    <optional><ref name="gender" /></optional>
    <optional><ref name="socialsecuritynumber" /></optional>
  </element></define>
 <define name="name"><element name="name"><text /></element></define>
 <define name="birthdate"><element name="birthdate"><text /></element></define>
 <define name="gender"><element name="gender"><text /></element></define>
 <define name="socialsecuritynumber"><element name="socialsecuritynumber"><text />
 </element></define>
</grammar>
```

Schematron

```
<schema xmlns="http://purl.oclc.org/dsdl/schematron">
   <pattern>
      <title>Date rules</title>
      <rule context="Contract">
         <assert test="ContractDate &lt; current-date()">ContractDate should be
 in the past because future contracts are not allowed.</assert>
      </rule>
   </pattern>
</schema>
```

https://en.wikipedia.org/wiki/Schematron

Schema Visual



Built-in primitive type. The string datatype represents character strings in XML.

Built-in primitive type. The string datatype represents character strings in XML.

Built-in primitive type. The string datatype represents character strings in XML.

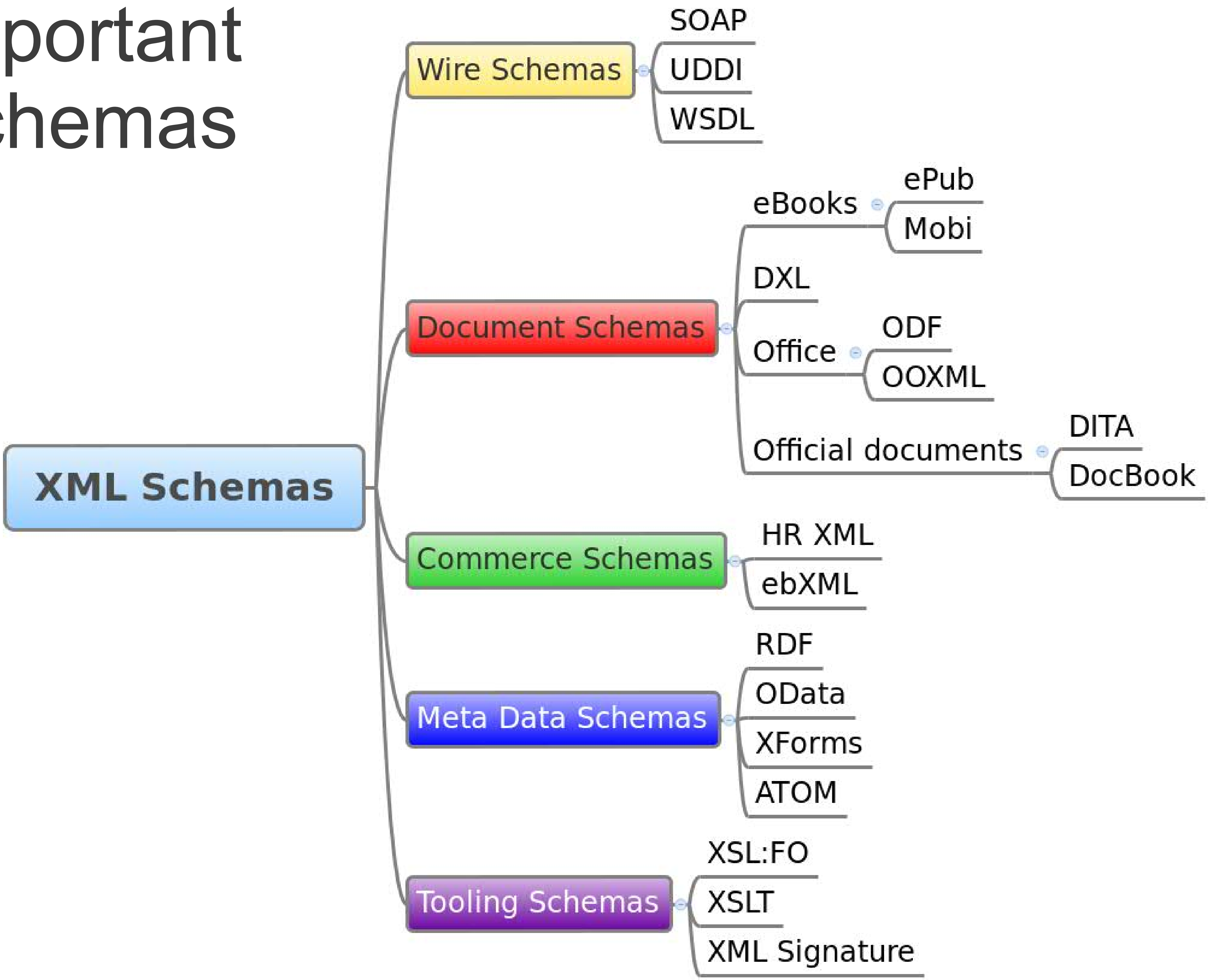
Important Schemas

- Your's!
- Wire Schemas
- Document Schemas
- Commerce Schemas
- Meta Data Schemas

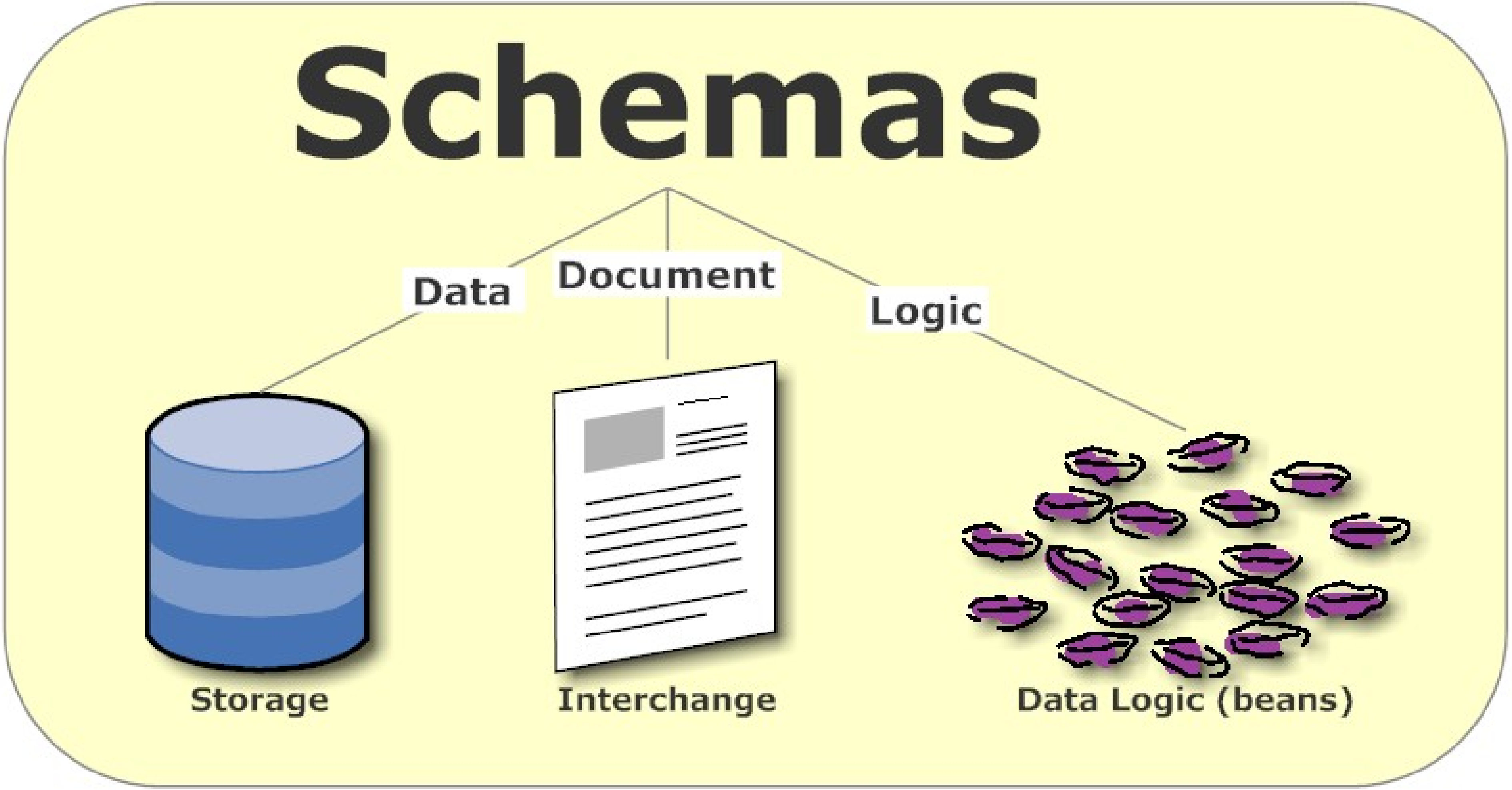
Note:

A schema if often created by a standard commitee (or the subversion of one). Don't expect them to be sleek!

Important Schemas



Schema Wars*

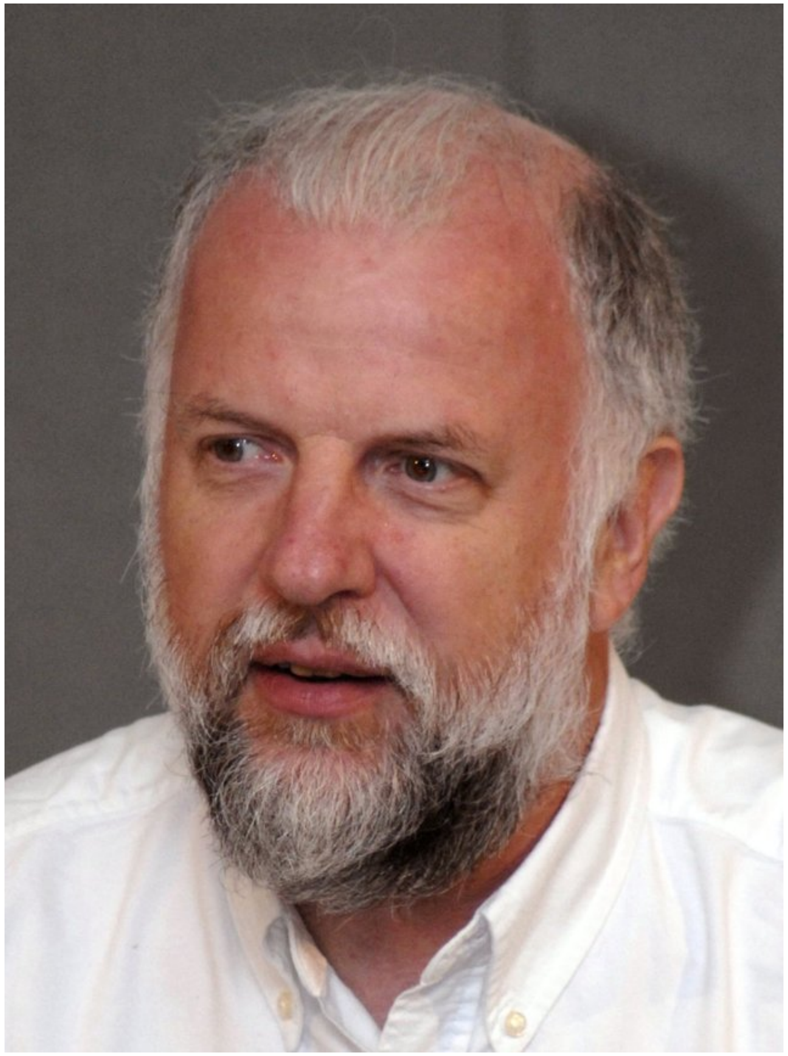


* UML as peace keeper?

Transform using XSLT

- Pattern matching
- Templates and XPath expressions
- Nightmare for "procedure guys"
- Performance traps!

His fault!



- Michael Kay
- Wrote SAXON parser
- Invented XPath
- Must have an EXTRABRAIN
- Very helpful
- On Mulberry mailing list

Sample XSLT

- Copy all NameSpaces into the XSLT
- Matching is by URL, not by prefix
  (Keeping the prefix is common practise)
- Add output definition
- Add (one or) more xsl:template with matching clauses (that's XPath)
- Run and have fun

XSLT - NameSpaces

-
  ```
  <xsl:stylesheet exclude-result-prefixes="xs xd" version="1.0"
  xmlns:cc="http://web.resource.org/cc/"
  xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:dcmitype="http://purl.org/dc/dcmitype/"
  xmlns:dcterms="http://purl.org/dc/terms/"
  xmlns:pgterms="http://www.gutenberg.org/rdfterms/"
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xd="http://www.oxygenxml.com/ns/doc/xsl"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  >
  ```

XSLT common elements

- <xsl:output encoding="UTF-8" indent="yes" method="xml" omit-xml-declaration="no" />
- <xsl:template match="somexpath">
- <xsl:apply-templates select=”somexpath”/>
- <xsl:value-of select=”somexpath” />
- <xsl:for-each select="somexpath">
- <xsl:element name=”usefulname”>
- <xsl:attribute name=”attname”>
- <xsl:variable name="aName" select="somexpath"/>

XPATH

Standard constructs

- **Start template**
```
<xsl:template match="/"><xsl:apply-templates />
</xsl:template>
```

- **Build in catch all template (2 pieces)**
```
<xsl:template match="*">
        <xsl:variable name="curTagname" select="name()"/>
        <xsl:element name="{$curTagname}">
            <!-- Walk through the attributes -->
            <xsl:apply-templates select="@*" />
            <!-- process the children -->
            <xsl:apply-templates />
        </xsl:element>
    </xsl:template>

<xsl:template match="@*" mode="genRead">
       <xsl:variable name="curAttName" select="name()"/>
        <xsl:attribute name="{$curAttName}">
            <xsl:value-of select="."/>
        </xsl:attribute>
    </xsl:template>
```

Standard constructs II

- **Catch all – supress output**
  <xsl:template match=”*” />
  Still produces whitespace
- **Sort stuff**
  <xsl:apply-templates><xsl sort />
  </xsl:apply-templates>
- **Render directive**
  <?xml-stylesheet type="text/xsl" href="some.xslt"?>
- **Note the difference***:
  - <xsl:element name=”test”></xsl:element>
  - <test></test>

* Hint: Namespace!

XPath

- A little like URLs, file path...
  ... when you begin

  and then:

XPath

- / = root of the XML before the first element
- ns:someelement = child element of the current element
- @attname = attribute of current element
- /oneele/twoele/three/@attname = absolute path to an attribute 3 levels deep
- //@attname = attribute anywhere in the tree
- * = every element
- @* = every attribute

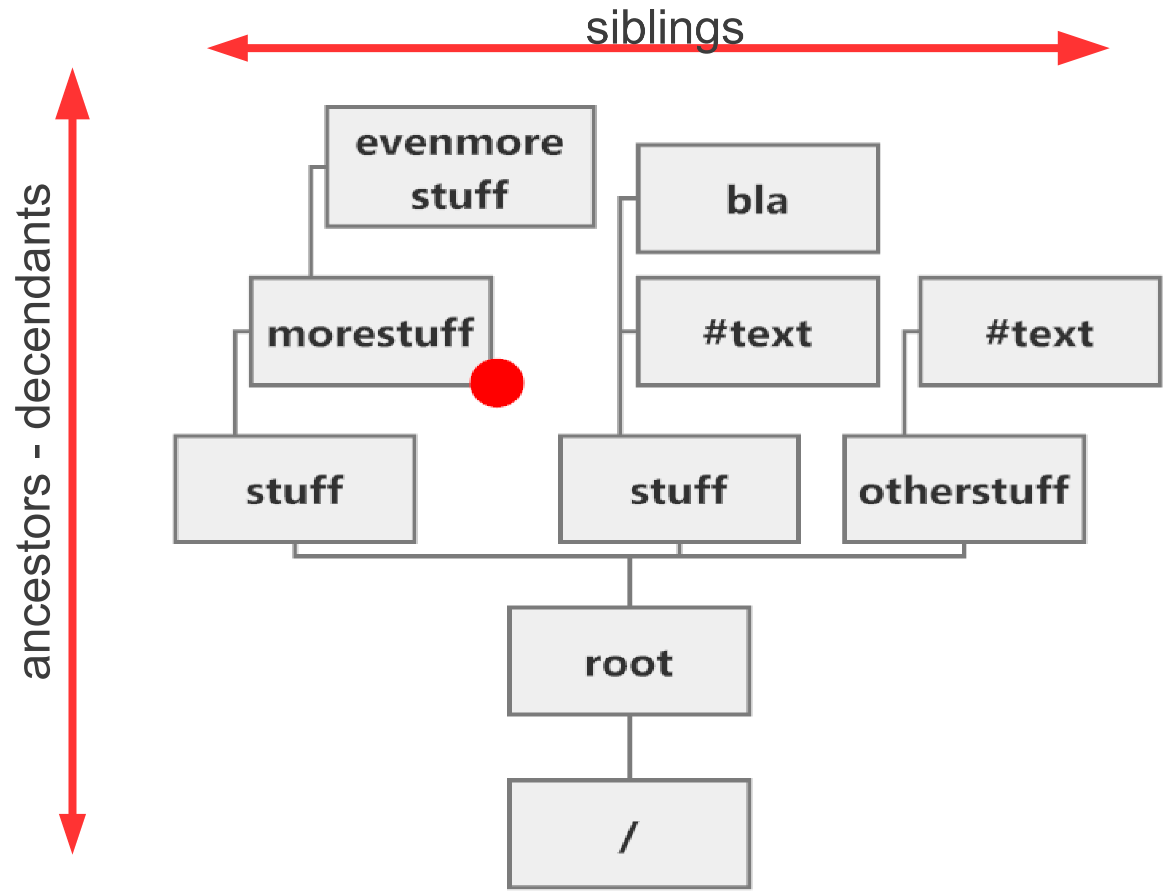
XPath

Then the AXIS kicks in:

- **ForwardAxis**
  child :: descendant :: attribute :: self :: descendant-or-self :: following-sibling :: following :: namespace ::
- **ReverseAxis**
  parent :: ancestor :: preceding-sibling :: preceding :: ancestor-or-self ::

XPath

- preceding-sibling :: title = title of element before
- descendant :: @url = all URL attributes



XPath Conditions & Functions

- //player[goals &gt; 0]
- xy:gene[@mutant='true']
- book[substring(preceding-sibling::title,1) != substring(title,1)]
- name() = name of element or attribute
- node() = whole element or attribute
- position() = position in current selection including last()

Priorities

- The better the match the higher the priority
- Tricky!
- “*” lowest priority
- “sometelement < somelement[somecondition]
- Concurrent conditions undefined!
  - <ele taste=”hot” color=”red”>....</...>
  - ele[@taste='hot'] ~ ele[@color='red']
  - ele[@taste='hot' and @color='red']

Mode

- Allows to run through elements multiple times
- Whole or partial tree
- Can be a performance drag
- Flexible

Book List Sample
Spring Clean Sample

Java

Jesse Gallagher:
XML manipulation in Java is like a sick joke

Reading XML in Java

- Tree (DOM)

- Stream (SAX)

Reading XML in Java

- Tree (DOM)
- In **memory** model
- XPath queries
- Manipulating content
- Flexible

- Stream (SAX)
- Series of events
- Fast
- Lean
- Suitable for large files

Read into DOM

- Any Stream can be used

-
  ```
  DocumentBuilderFactory factory =
  DocumentBuilderFactory.newInstance();
  factory.setValidating(false); // Will blow if set to true
  factory.setNamespaceAware(true);
  InputSource source = new InputSource(new
  StringReader(sourceString));
  DocumentBuilder docb = factory.newDocumentBuilder();
  Document d = docb.parse(source);
  ```
- Document (XML) & Document (Notes) = Headache

Read with SAX

-
  ```
  XMLReader xmlReader = XMLReaderFactory.createXMLReader();
  FileReader reader = new FileReader("somefile.xml");
  InputSource inputSource = new InputSource(reader);
  xmlReader.setContentHandler(new SaxReadExample());
  xmlReader.parse(inputSource);
  ```

-
  ```
  public void characters(char[] ch, int start, int length) throws SAXException {}
  public void endDocument() throws SAXException {}
  public void endElement(String arg0, String arg1, String arg2) throws SAXException {}
  public void endPrefixMapping(String arg0) throws SAXException {}public void
  ignorableWhitespace(char[] arg0, int arg1, int arg2) throws SAXException {}
  public void processingInstruction(String arg0, String arg1) throws SAXException {}
  public void setDocumentLocator(Locator arg0) {}
  public void skippedEntity(String arg0) throws SAXException {}
  public void startDocument() throws SAXException {}
  public void startElement(String arg0, String arg1, String arg2, Attributes arg3) throws
  SAXException {}
  public void startPrefixMapping(String arg0, String arg1) throws SAXException {}
  ```

Write from DOM

- Document.toString() doesn't work

-
  ```
  TransformerFactory tFactory =
  TransformerFactory.newInstance();
  Transformer transformer = tFactory.newTransformer();
  StreamResult xResult = new StreamResult(new StringWriter());
  DomSource source = new DOMSource(dom);
  // Suppress the XML declaration in front
  transformer.setOutputProperty("omit-xml-declaration", "yes");
  transformer.transform(source, xResult);
  ```
- `String result = xResult.getWriter().toString();`

Write from SAX

-
  ```
  PrintWriter pw = new PrintWriter(out);
  StreamResult streamResult = new StreamResult(pw);
  SAXTransformerFactory tf = (SAXTransformerFactory)
  TransformerFactory.newInstance();TransformerHandler hd =
  tf.newTransformerHandler();
  Transformer serializer = hd.getTransformer();
  serializer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
  serializer.setOutputProperty(OutputKeys.METHOD,"xml");
  serializer.setOutputProperty(OutputKeys.INDENT, "yes");
  hd.setResult(streamResult);
  hd.startDocument();
  atts.addAttribute("", "", "someattribute", "CDATA", "test");
  atts.addAttribute("", "", "moreattributes", "CDATA", "test2");
  hd.startElement("", "", "MyTag", atts);
  String curTitle = "Something inside a tag";
  hd.characters(curTitle.toCharArray(), 0, curTitle.length());
  hd.endElement("", "", "MyTag");
  hd.endDocument();
  ```

Avoid low level XML!

- JAXP
- ATOM
- ODATA
- Apache POI
- Apache ODF Toolkit
- IBM Social Business Toolkit

JAXP

- XML equivalent to Google GSON
- @XmlRootElement(name = "SomeName")
- @XmlElement(name = "SomeName")
- JAXBContext context = JAXBContext.newInstance(BookingList.class); Marshaller m = context.createMarshaller(); m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE); m.marshal(this, out);

```
JAXBContext context =
JAXBContext.newInstance(BookingList.class);
Marshaller m = context.createMarshaller();
m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
m.marshal(this, out);
```

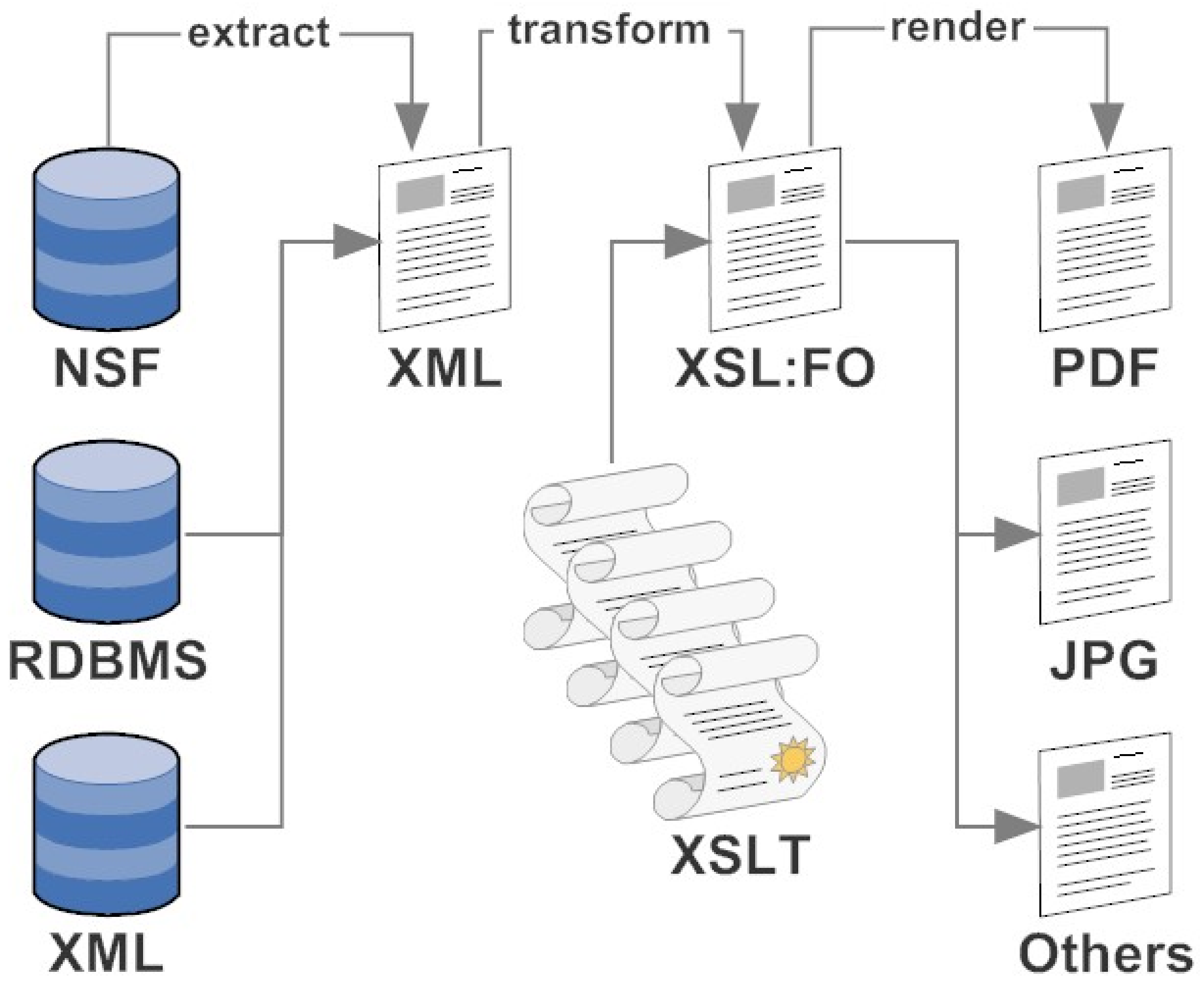
- Unmarshaller u = context.createUnmarshaller(); BookingList b = (BookingList) u.unmarshal(in);

```
Unmarshaller u = context.createUnmarshaller();
BookingList b = (BookingList) u.unmarshal(in);
```

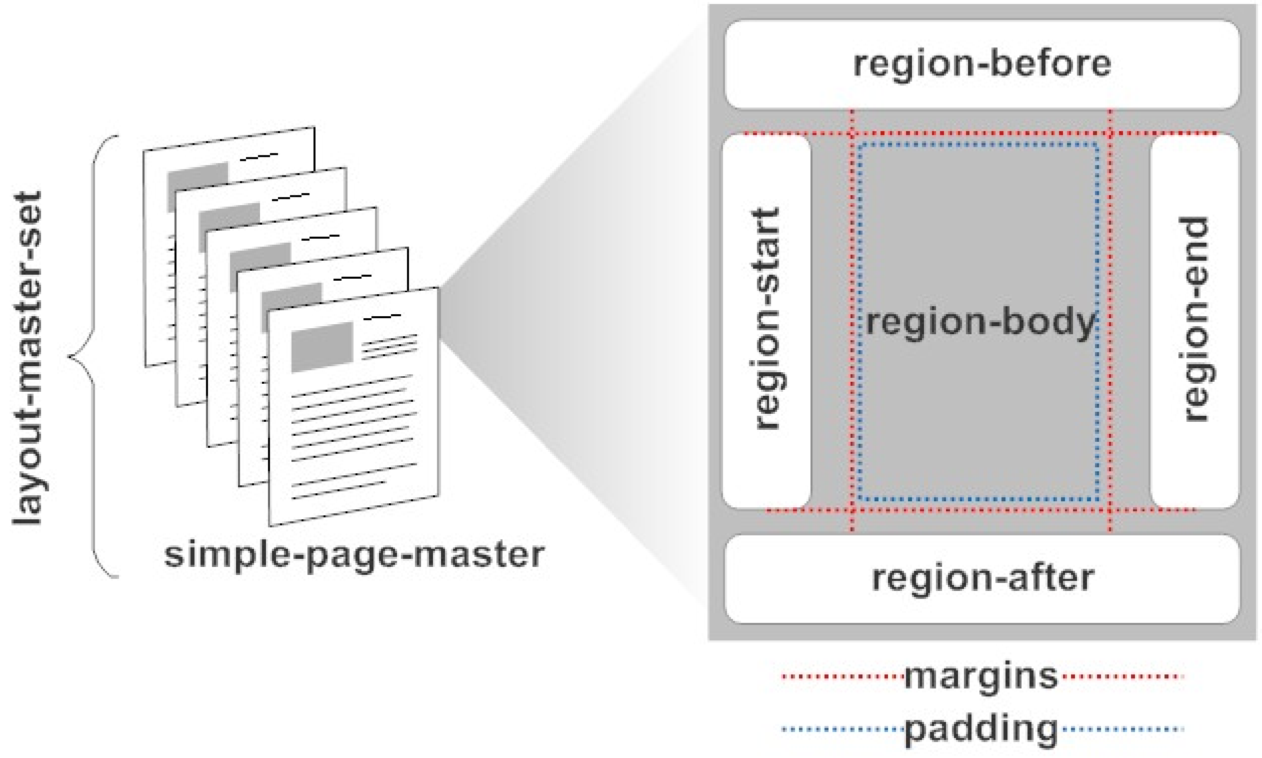
Signature

- Platform, vendor & language independent signing of XML data
- Handles white space challenge
- Requires a key
- http://www.w3.org/Signature/
- http://santuario.apache.org/
- KMIP emerging standard support
  some lobby work needed
- https://en.wikipedia.org/wiki/Key_Management_Interoperability_Protocol

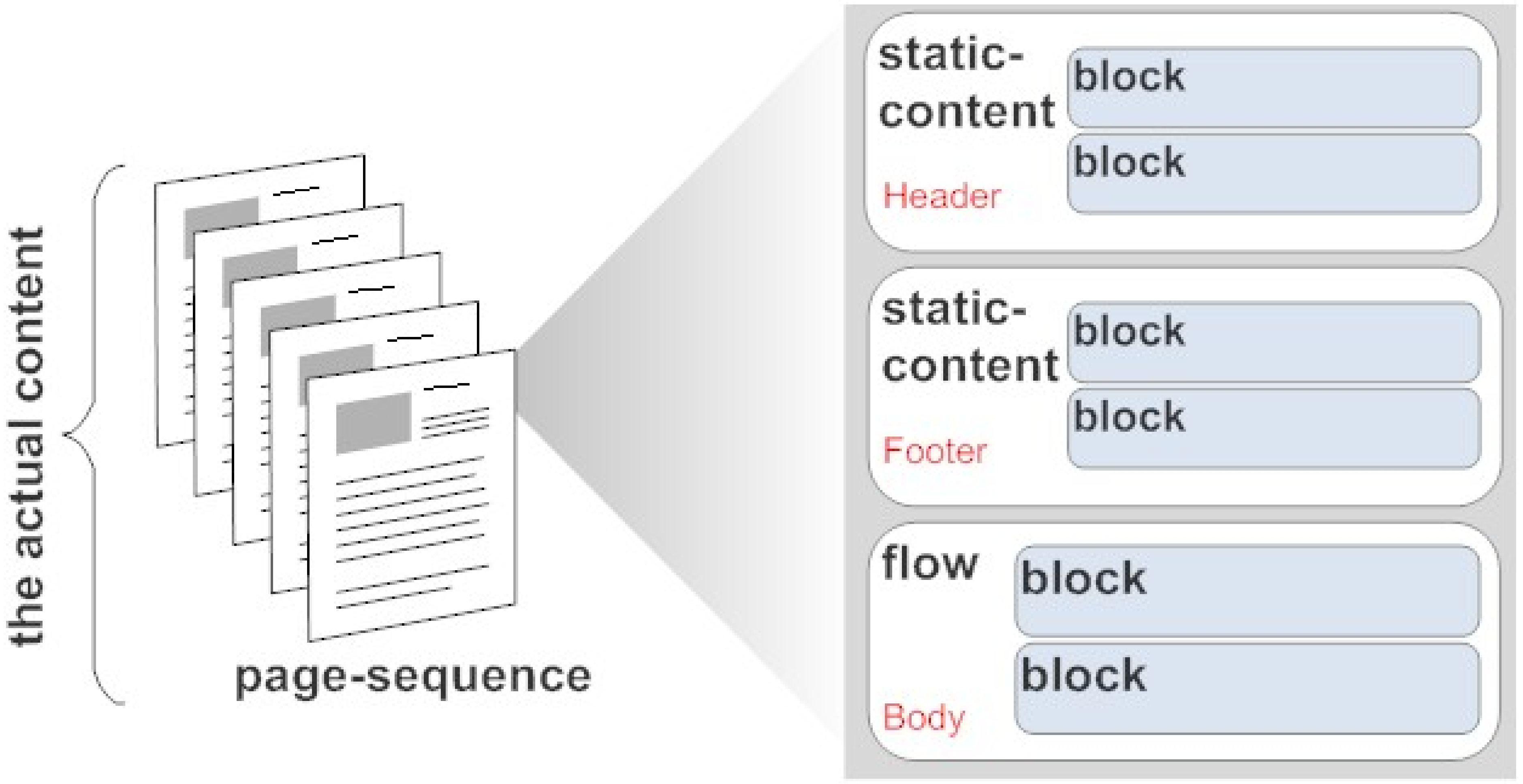
Transform using XSL:FO



Transform using XSL:FO



Transform using XSL:FO



Transform using XSL:FO

- FOP as only one input and one output!
- Input needs to be a FOP String
- Usually produced by an XSLT transformation

-
  ```
  FopFactory fopFactory  = FopFactory.newInstance();
  FOUserAgent ua = fopFactory.newFOUserAgent();
  Fop fop = this.fopFactory.newFop(MimeConstants.MIME_PDF, ua, out);
  InputSource fopSrc = new InputSource(in);
  SAXParser parser = this.getParser();
  DefaultHandler dh = fop.getDefaultHandler();
  parser.parse(fopSrc, dh);
  ```

XML and HTML

- If you are lucky it is xHTML
- For the rest there is Jericho and HTMLCleaner
-

XML and JSON

- Best using JXP and GSON
- Second XSLT

XML as Data Source

- XML Document object (Scope, Bean etc)
- Xpath expressions for Data bindings
- ${xpath:document:/person/firstName}

Fun with DXL and XPages sources

- Make an XPage out of a view
- Make an XPage, Form, View from a schema

DB/2 PureXML

- The closest you get in the RDBMs world to a Domino Document
- That's what NotesDB2 should have looked like!
-
```
create view commentview(itemID, itemname, commentID, message) as
select i.id, i.itemname, t.CommentID, t.Message
from items i,
xmltable('$c/Comments/Comment' passing i.comments as "c"
columns CommentID integer path 'CommentID',
Message varchar(100) path 'Message' ) as t;
```