Languages of the WEB

Jukka K. Nurminen
24.1.2012

Last Time

- Introduction
- Practicalities (Lectures, Assignments, etc.)

Today

- What are the languages of web
- Presentation and **data**
  - HTML, CSS
  - **XML**
    - **XML tools**
    - **EXI**
  - **JSON**
- Programming
  - JavaScript (client side at browser)
  - PHP, Python, Pearl, Java, … (server side)

- XML Schema Assignment
- Pair formation during break/after lecture
- Signup for assignments by tomorrow
- Android lecture following

HTML

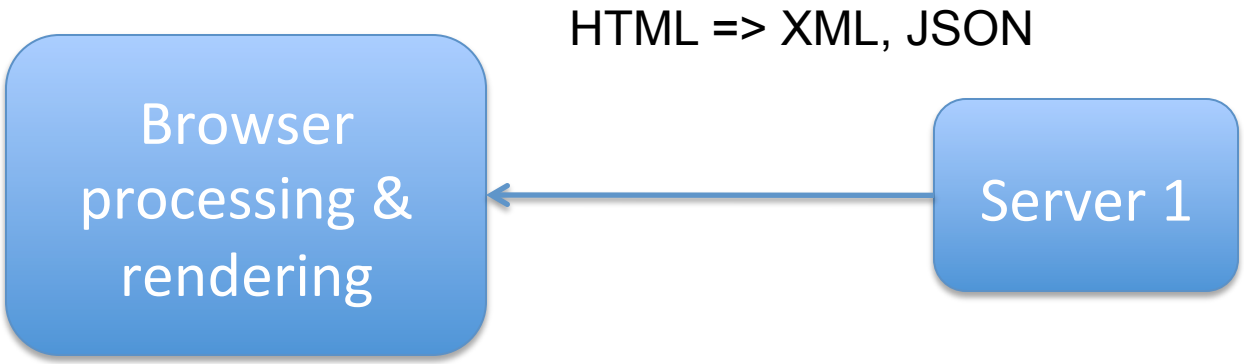
```
<html>
  <head>
    <title> title goes
  here </title>
  </head>

  <body bgcolor="white"
  text="blue">

  <h1> My first page </
  h1>

  This is my first web

  </body>
</html>
```

- Tags specify how the page looks like (e.g. h1)
- This is fine for the browser to show static pages
- But it is not easy to use this form for data transfer

Server-Browser data transfer (static content)

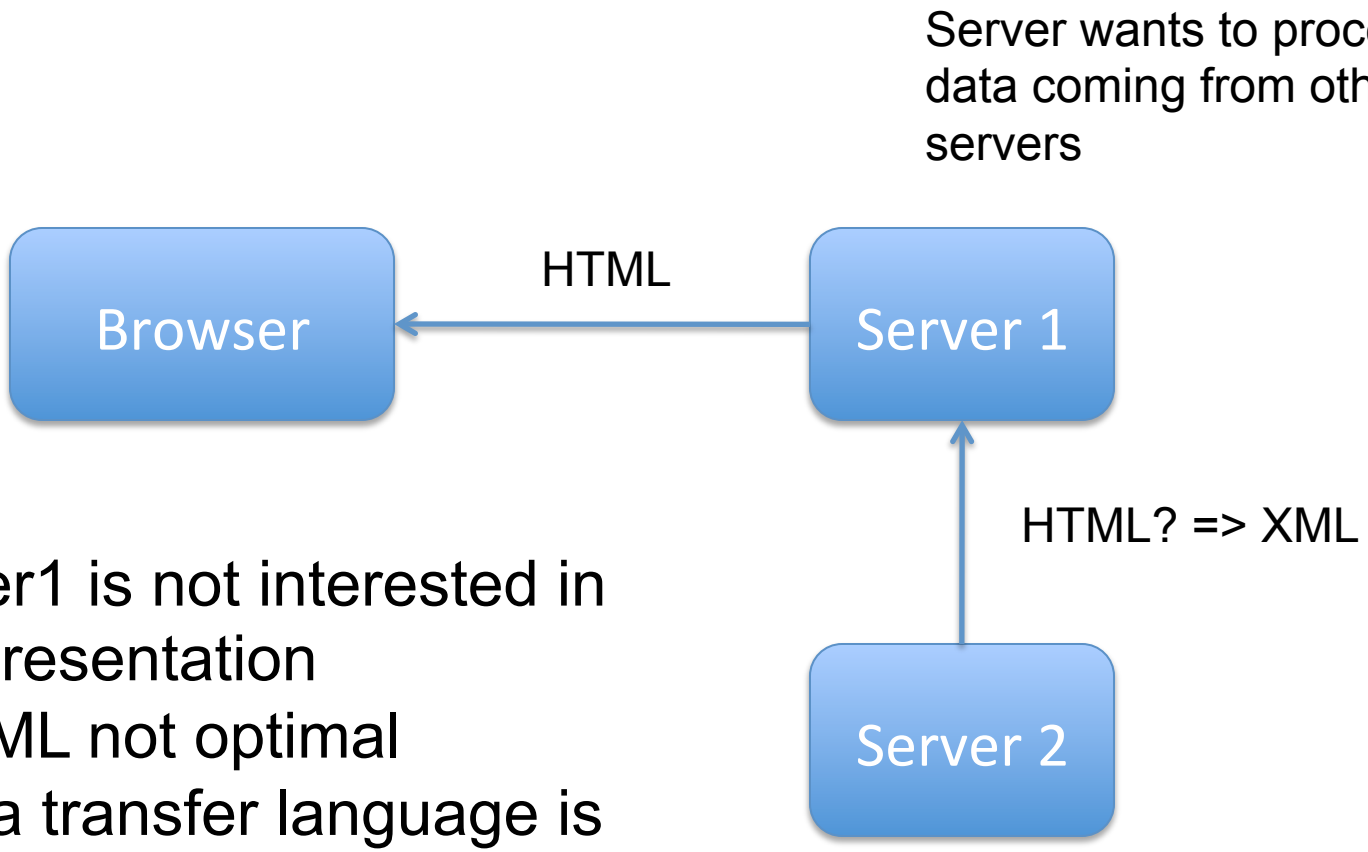
- When the browser just renders the page created by the server HTML works fine

Server-Browser data transfer (dynamic content)



- If the server needs to process the incoming data HTML is problematic.
  - E.g. in if only part of the page needs to be redisplayed (as is the case in AJAX)
  - This requires that the browser is able to parse and understand the data to be able to process it

Server-Server data transfer



- Server1 is not interested in page presentation
  => HTML not optimal
- A data transfer language is needed
  => XML

Personalized tags needed

- HTML started with very few tags …
- Language evolved, as more tags were added
  - forms
  - tables
  - fonts
  - frames

⇒ Need for personalized tags for different domains

⇒ E.g. for math, music, purchase orders, …

MathML

- Designed to express ***semantics*** of maths
- Cut & paste into Maple, Mathematica

- **$x^2 + 4x + 4 = 0$**

```
<mrow>
 <mrow>
    <msup> <mi>x</mi> <mn>2</mn> </msup>
   <mo>+</mo>
      <mrow>
       <mn>4</mn>
       <mo>&invisibletimes;</mo>
       <mi>x</mi>
      </mrow>
  <mo>+</mo>
  <mn>4</mn>
 </mrow>
  <mo>=</mo>
  <mn>0</mn>
</mrow>
```

Applications need to parse the data

- HTML syntax was loosely defined which makes its programmatic manipulation difficult
  - Closing tags are optional
  - Different browsers tolerated different violations of HTML syntax
  - (Newer HTML specs XHTML, HTML5 stricter in this sense, but their adoption is slow because of backward compatibility issues)

$\Rightarrow$ Need for a more structured and well-defined presentation

XML Example

```
<?xml version="1.0"?>
<product barcode="2394287410">
  <manufacturer>Verbatim</manufacturer>
  <name>DataLife MF 2HD</name>
  <quantity>10</quantity>
  <size>3.5"</size>
  <color>black</color>
  <description>floppy disks</description>
</product>
```

Elements and attributes

- <messages>
  <note id="p501">
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
  </note>
  <note id="p502">
  <to>Jani</to> …

Attributes (for auxiliary data)

Elements between tags

Borderline between elements and attributes is not clear. Attributes are mainly for auxiliary data. Elements for main data

Suitability of XML

- Suitable for storing and exchanging any data that can plausibly be encoded as text.
- Unsuitable for digitized data such as photographs, recorded sound, video, and other very large bit sequences
  - But it works well for storing the metadata of such items
- Uses of XML
  - Data transfer between applications
  - Configuration files

Example Uses

- A web browser, such as Netscape Navigator or Internet Explorer, that displays the document to a reader
- A word processor, such as StarOffice Writer, that loads the XML document for editing
- A database, such as Microsoft SQL Server, that stores the XML data in a new record
- A personal finance program, such as Microsoft Money, that sees the XML as a bank statement
- A syndication program that reads the XML document and extracts the headlines for today's news

What XML is not

- Not a programming language
  - you cannot execute XML
- Not a network transport protocol
  - Actual sending with HTTP, FTP, etc.
- Not a database
  - But XML can be stored and retrieved from databases
  - Some databases are able to return query results in XML form

Parsing XML

DOM

- Stores the entire XML document into memory before processing
- Occupies more memory
- We can insert or delete nodes
- Traverse in any direction.
- DOM is a tree model parser
- Document Object Model (DOM) API
- Preserves comments
- SAX generally runs a little faster than DOM

SAX

- Parses node by node
- Doesn't store the XML in memory
- We cant insert or delete a node
- SAX is an event based parser
- SAX is a Simple API for XML
- doesn't preserve comments
- SAX generally runs a little faster than DOM

Performance comparison

Test #	1	2	3	4	5	5
Parser	Small Read (s)	Large Read (s)	Large Nav (s)	Build Large (s)	Build Huge (s)	Max Size (elements)
SunDOM	0.022	3.732	0.21	0.496	12.33	440,358
OracleDOM	0.014	2.976	0.06	0.926	8.23	281,308
XercesDOM	0.042	2.482	0.078	0.81	10.11	389,044
SunSAX	0.018	0.7	-	-	-	-
OracleSAX	0.01	0.546	-	-	-	-
XercesSAX	0.036	1.3	-	-	-	-
XPSAX	0.016	0.458	-	-	-	-

Table 1: Test Results

Source: http://www.devx.com/xml/Article/16922/1954

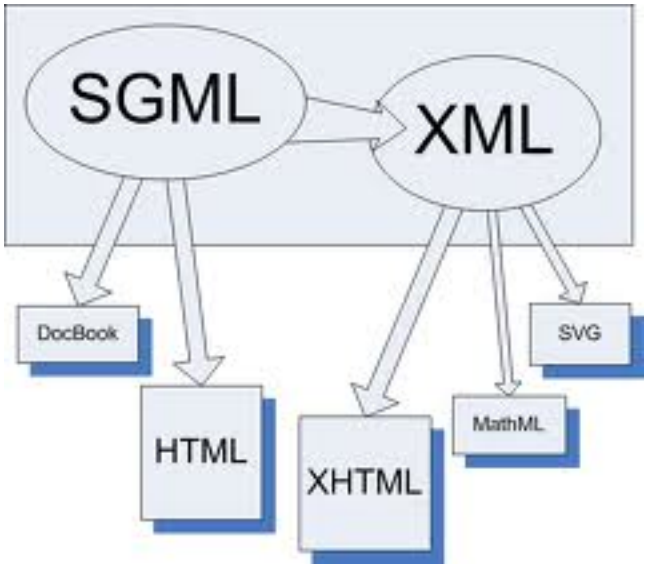
HTML vs. XML

- HTML
  - Fixed set of tags
  - Presentation oriented
  - The language of web pages
- XML
  - Extensible set of tags
  - Content oriented
  - The meta language for defining new domain specific languages

SGML

- SGML
  - Standard Generalized Markup Language
  - Meta language for defining languages
  - Complex, sophisticated, powerful
  - Very advanced but also very complicated

Old markup languages: troff, tex, etc



Well-formed and Valid

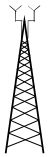
- Well-formed
  - Syntactically correct XML
- Valid
  - Matches a defined XML Schema

New (or not so new) Challenges

Internet-of-Things (M2M communication)

Rise of dominant proprietary solutions (Google, Facebook, etc.)

Data center



Cellular access

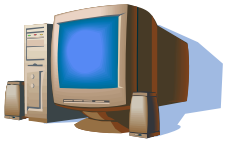
Server to server communication

Mobile clients, heterogeneous clients

Phone

Internet

Data center



PC

WLAN/ADSL access

User generated content

Social networking

Usability of battery powered devices, energy cost, and environmental concerns

XML Summary

- Metamarkup language, standardized by W3C
- In comparison to HTML nothing about presentation
- Elements within tags
- XML Applications
  - Individual or organizations agree to use a set of tags
- Well-formed - Syntactically correct XML
- Valid - Matches Schema
- Schema definitions
  - DTD
    - document type definition
  - XML Schema

Related standards

- Namespaces
  - Modular document definition, multiple inheritance, collision avoidance
- XPath / XQuery
  - Navigation and query of parts of the document
- XML linking language (Xlink)
  - Associations between multiple resources
  - Rules for traversal
- XML Schema
  - definition of document structure and custom data types
- XSLT
  - Extensible Stylesheet Language Transformation
  - Transformation of documents

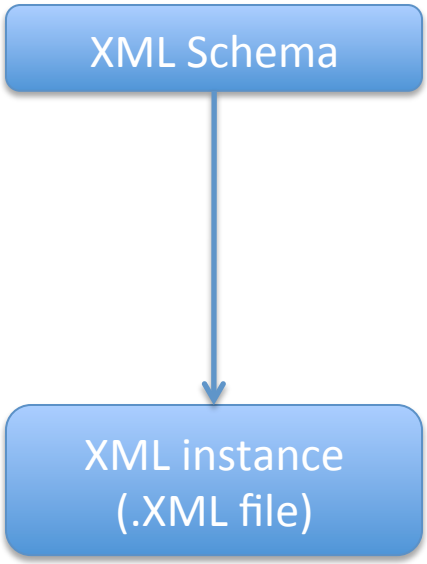
XML Schema and other standards

XML Schemas and instances

- To be useful all parties have to agree what tags are available and what they mean
- => Need to describe what tags are available and how they are used
- Also what kind of values should they contain

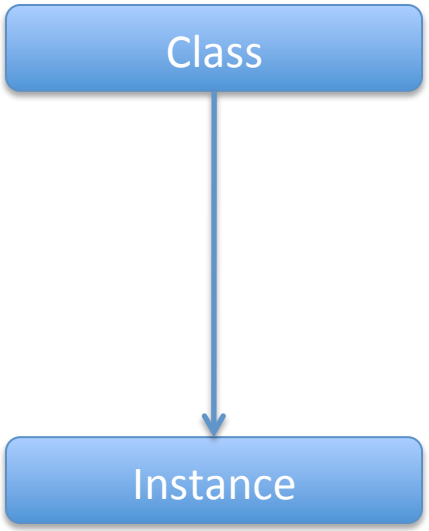
XML Schemas and instances

Defines tags and other parameters



Provides the data

Compare to object-oriented programming



Schema definition

- Main alternatives
  - DTD
    - Document Type Definition
    - Borrowed from SGML
  - XML Schema
    - XML based way to define XML schemas

DTD

```
<!DOCTYPE NEWSPAPER [
<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE
(HEADLINE,BYLINE,LEAD,BODY,NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>
]>
```

Limitations of DTD

- Difficult to define other constraints than element nesting and recurrence
- DTD syntax is not XML

XML Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>
<xs:element name="note">
 <xs:complexType>
    <xs:sequence>
       <xs:element name="to" type="xs:string"/>
       <xs:element name="from" type="xs:string"/>
       <xs:element name="heading” type="xs:string"/>
       <xs:element name="body" type="xs:string"/>
    </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

About XML Schema

- XML language for describing and constraining the content of XML documents
- A W3C Recommendation
- Used to specify
  - The allowed structure of an XML document
  - The allowed data types contained in XML documents
- XML Schema documents are XML documents
- Schema document: elements, attributes, and type definitions + annotations

Defining a simple element

- A simple element is defined as
  <xs:element name="*name*" type="*type*" />
  where:
  - *name* is the name of the element
  - the most common values for *type* are
    xs:boolean xs:integer
    xs:date xs:string
    xs:decimal xs:time
- Example
- <xs:element name="to" type="xs:string"/>

Defining an attribute

- Attributes themselves are always declared as simple types
- An attribute is defined as
  <xs:attribute name="*name*" type="*type*" />
  where:
  - *name* and *type* are the same as for xs:element
- Other attributes a simple element may have:
  - default="*default value*" *if no other value is specified*
  - fixed="*value*" *no other value may be specified*
  - use="optional" *the attribute is not required (default)*
  - use="required" *the attribute must be present*
- Example
  - <xs:attribute name="orderid" type="orderidtype" use="required"/>

Restrictions, or “facets”

- The general form for putting a restriction on a text value is:
  - <xs:element name="*name*"> *(or* xs:attribute*)*

```
<xs:restriction base="type">
    ... the restrictions ...
</xs:restriction>
</xs:element>
```

- For example:
  - <xs:element name="age">

```
<xs:restriction base="xs:integer">
    <xs:minInclusive value="0">
    <xs:maxInclusive value="140">
</xs:restriction>
</xs:element>
```

Complex elements

- A complex element is defined as

```
<xs:element   name="name">
    <xs:complexType>
        ... information about the complex type...
    </xs:complexType>
</xs:element>
```

- Example:

```
<xs:element   name="person">
    <xs:complexType>
        <xs:sequence>
            <xs:element  name="firstName"  type="xs:string" />
            <xs:element  name="lastName"  type="xs:string" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Local definition of complex type

- <xs:sequence> says that elements must occur in this order

Globally defined complex type

- Definition

```
<xs:complexType name=“personType”>
    <xs:sequence>
        <xs:element  name="firstName"  type="xs:string" />
        <xs:element  name="lastName"  type="xs:string" />
    </xs:sequence>
</xs:complexType>
```

- Use
- <xs:element name="student" type="personType"/>
- <xs:element name="professor" type="personType"/>

XSLT

How to transform XML documents to other kinds of documents?

XSLT

- XSLT stands for Extensible Stylesheet Language Transformations
- XSLT is used to transform XML documents into other kinds of documents, e.g. to HTML or other kind of XML
- XSLT uses *two* input files:
  - The XML document containing the actual data
  - The XSL document containing both the "framework" in which to insert the data, *and* XSLT commands to do so

Why transform?

- Convert one schema to another
- Rearrange data for formatting
- Some special transforms
  - XML to HTML— for old browsers
  - XML to LaTeX—for TeX layout
  - XML to SVG—graphs, charts, trees
  - XML to plain-text—occasionally useful

Very simple example

- File data.xml:

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/xsl" href="render.xsl"?>
<message>Hello</message>
```

- File render.xsl:

```
<?xml version="1.0"?>
<xsl:stylesheet version="1.0”
        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <!-- one rule, to transform the input root (/) -->
  <xsl:template match="/">
    <html><body>
      <h1><xsl:value-of select="message"/></h1>
    </body></html>
  </xsl:template>
</xsl:stylesheet>
```

Tells which part of the tree to process

Finds the “message” element

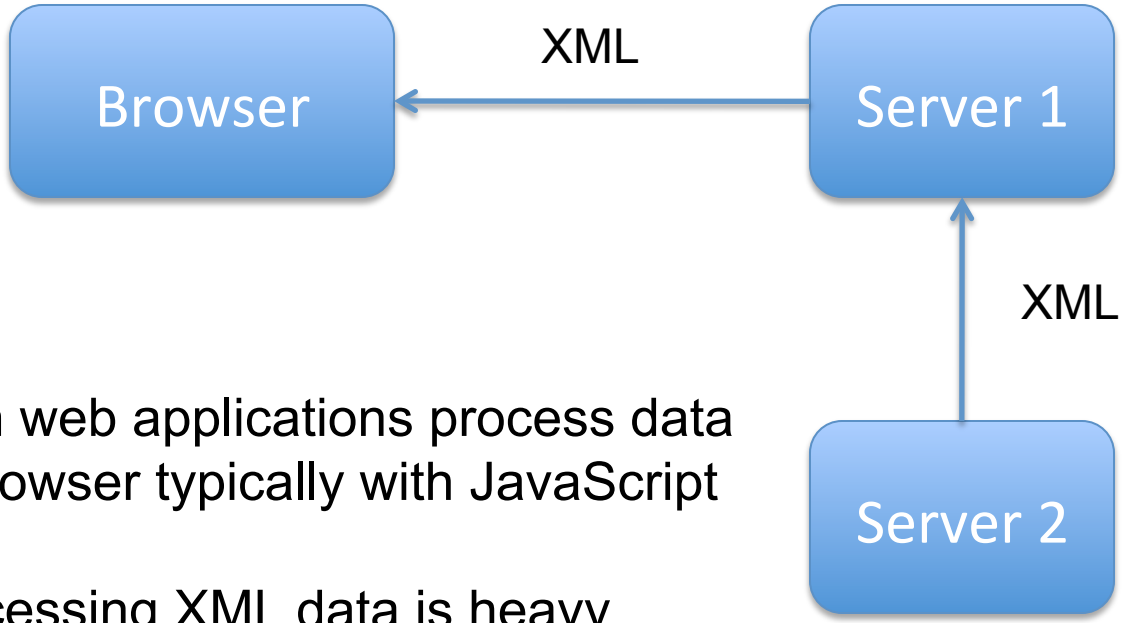
JSON

JavaScript Object Notation

The trouble with XML?

XML documents tend to be verbose
=> Communication overhead

Not very easy for humans to write or read



- Modern web applications process data in the browser typically with JavaScript (AJAX)
- => processing XML data is heavy

JSON

- JavaScript Object Notation
- Minimal
- Textual
- Subset of JavaScript

- Increasingly popular
  - Many services return data in JSON format
- Yet another example where simpler is better

JSON Example

```
{"skills": {
"web":[
   {"name": "html",
    "years": "5”
     },
   {
     "name": "css",
     "years": "3”
}],
"database":[
   {"name": "sql",
    "years": "7"}]
}}
```

What is JSON?

- Lightweight data-interchange format
  - Compared to XML
- Simple format
  - Easy for humans to read and write
  - Easy for machines to parse and generate
- JSON is a text format
  - Programming language independent
  - But is especially well suited for JavaScript manipulation

JSON & JavaScript

- You can evaluate the JSON object in JavaScript
  - Assign it to a variable and access via normal JavaScript mechanisms for objects and arrays
- Compare this with the XML approach where you have to traverse the XML tree to extract the values before you can do something with them

Similarities between JSON and XML

- They are both 'self-describing' meaning
  - that values are named, and thus 'humanreadable'
- Both are hierarchical. (i.e. You can have values within values.)
- Both can be parsed and used by lots of programming languages
- Both can be passed around using AJAX (i.e. httpWebRequest)

JSON vs. XML

- Lighter and faster than XML as on-the-wire data format
  - Relevant for mobile devices, data transfer costs
- JSON is less verbose
  - Quicker for humans to write, and probably easier to read
- JSON objects are typed while XML data is typeless
  - JSON types: string, number, array, boolean,
  - XML data are all string
- Native data form for JavaScript code
  - XML data needed to be parsed and assigned to variables through tedious DOM APIs
  - Data is readily accessible as JSON objects in your JavaScript code
  - Retrieving values is as easy as reading from an object property in your JavaScript code

Efficient XML Interchange (EXI)

- **What is EXI?**
- EXI stands for Efficient XML Interchange. It is commonly known as "binary XML"
- EXI enables you to operate on XML without being aware that you are using a much smaller binary-formatted XML
- EXI is a W3C recommendation
- **Benefits of EXI?**
- EXI avoids the bloatedness of text-formatted XML
- Applications can generate EXI directly and can operate on EXI directly, without first converting to text-formatted XML
- You can validate a binary-formatted XML document in the same way you validate a text-formatted XML document

notebook.xml

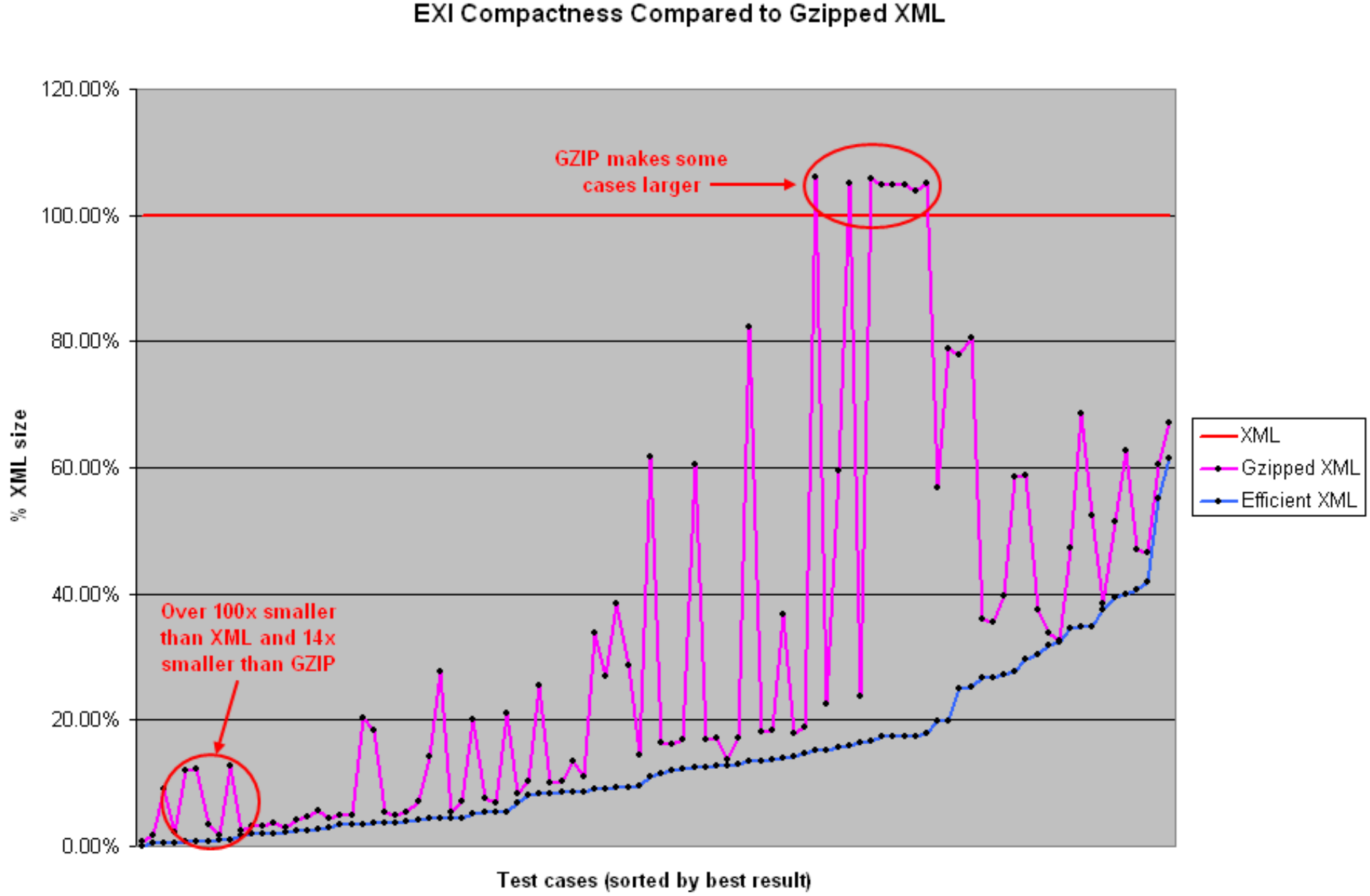
```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="notebook.xsl"?>
<!-- Comment before the root element -->
<nbk:notebook date="2007-09-12"
     xmlns:nbk="http://www.notebook.org"
     xmlns:c="http://www.category.org">
    <?realaudio version="5.0" frequency="5.5kHz" bitrate="16Kbps"?>
    <!-- Commend inside the root element -->
    <nbk:note date="2007-07-23" c:category="EXI">
        <nbk:subject>EXI</nbk:subject>
        <nbk:body>Do not forget it!</nbk:body>
    </nbk:note>
    <nbk:note date="2007-09-12">
        <nbk:subject>shopping list</nbk:subject>
        <nbk:body>milk, honey</nbk:body>
    </nbk:note>
</nbk:notebook>
<?test a="blah"?>
<!-- Comment after the root element -->
```

Contains comments, PIs, namespace prefixes.

Source: Roger L. Costello, 2011 http://www.xfront.com/EXI/index.html

notebook.exi

```
  Ú ‹ËÝÝÝË››Ý X›ÛÚË›Ü™Â[ ›Ý X›ÛÚÒ
ÈÂèÊ d``nZ`rZbe +s{£*@
Œ
ËL
ËLŒè http://www.category.org
          category EXI¨ æêÄÔÊÆé€Ê ±7²<âh ä
Íî„ÍîLì®„
.„&€? îm
î
-Ìä
 .n€ ÚÒØÖX@ÐÞÜÊòP
```


EXI Compactness Compared to Gzipped XML
120.00%
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
% XML size
GZIP makes some cases larger
Over 100x smaller than XML and 14x smaller than GZIP
Test cases (sorted by best result)
XML
Gzipped XML
Efficient XML

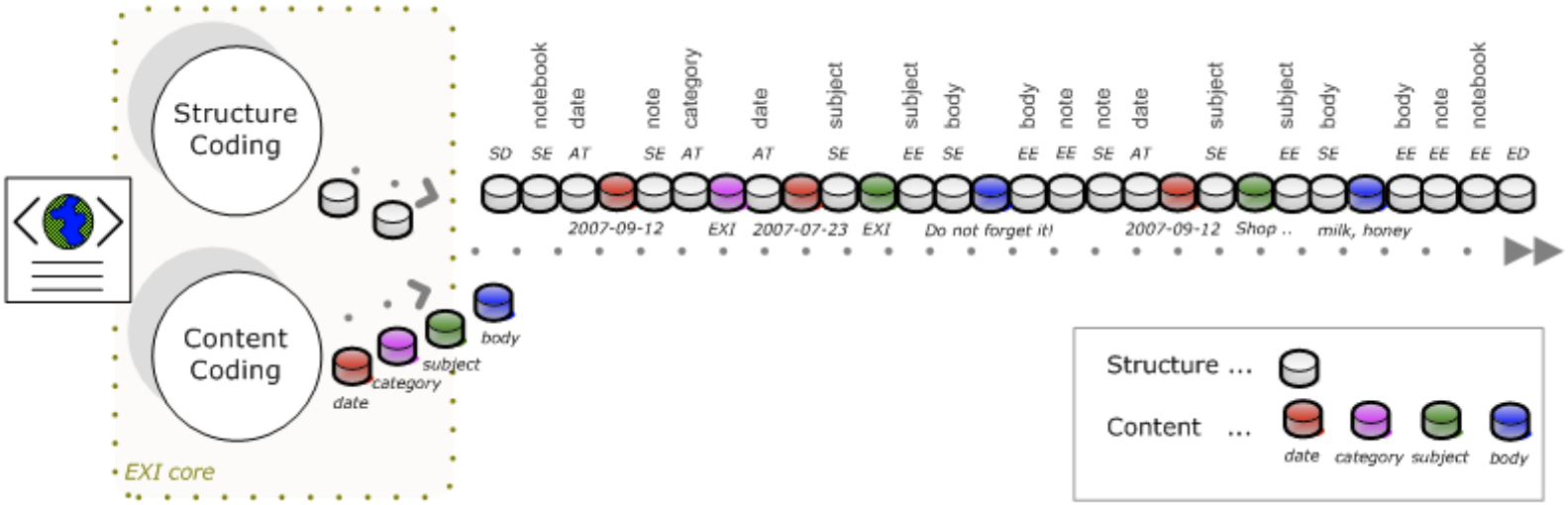
How EXI compression works

Example 2-1. Notebook (XML Document)

```
<?xml version="1.0" encoding="UTF-8"?>
<notebook date="2007-09-12">
 <note category="EXI" date="2007-07-23">
  <subject>EXI</subject>
  <body>Do not forget it!</body>
 </note>
 <note date="2007-09-12">
  <subject>Shopping List</subject>
  <body>milk, honey</body>
 </note>
</notebook>
```

See: http://www.w3.org/TR/2009/WD-exi-primer-20091208/

Convert into stream of events



Event types

Table 2-5. EXI Event types

EXI Event Type	Grammar Notation	Information Items	
		Structure	Content
Start Document	SD		
End Document	ED		
Start Element	SE (*qname*)	[*prefix*]	
	SE (*uri*:*)	*local-name*, [*prefix*]	
	SE (*)	*qname*, [*prefix*]	
End Element	EE		
Attribute	AT (*qname*)	[*prefix*]	*value*
	AT (*uri*:*)	*local-name*, [*prefix*]	
	AT (*)	*qname*, [*prefix*]	
Characters	CH		*value*
Namespace Declaration[1]	NS	*uri, prefix, local-element-ns*	
Comment[1]	CM	*text*	
Processing Instruction[1]	PI	*name, text*	
DOCTYPE[1]	DT	*name, public, system, text*	
Entity Reference[1]	ER	*name*	
Self Contained[1]	SC		

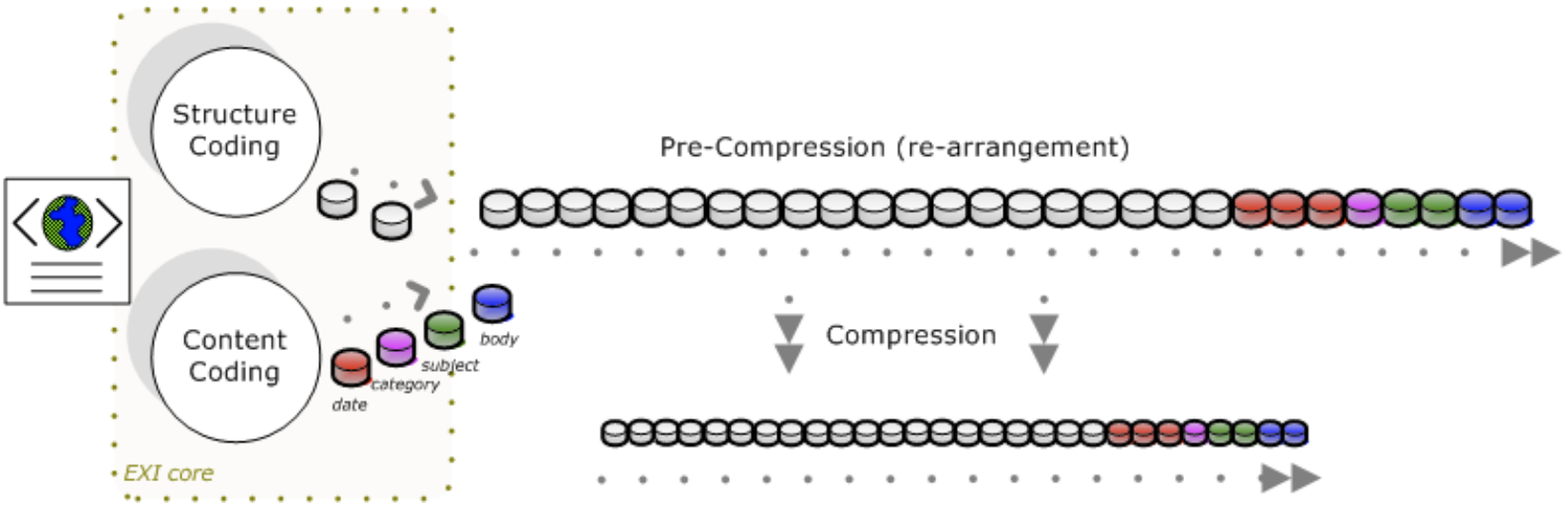
[1] EXI Options such as preserve and selfContained can be used to prune events from the EXI stream to realize a more compact representation.

Efficient coding of events

Event	EventCode			#bits
AT(date)	0			2
AT(category)	1			
EE	2	0		2 + 3
AT(*)	2	1		
NS	2	2		
SE(*)	2	3		
CH	2	4		
CM	2	5	0	2 + 3 + 1
PI	2	5	1	
#distinct values	3	6	2	

EXI Event Code Assignment

Better compression by re-arrangement



Summary of today

- HTML
- XML and related languages
  - XML
  - DTD
  - XML Schema
  - XSLT
- JSON
- EXI