XML and friends

- **history/background**
  - GML (1969)
  - SGML (1986)
  - HTML (1992)
  - World Wide Web Consortium (W3C) (1994)
- **XML** (1998)
  - core language
  - vocabularies, namespaces: XHTML, RSS, SVG, MathML, Schema, …
  - XML validation: Schema, DTD
  - XML parsers: SAX, DOM
  - processing XML documents: XPath, XSLT, XQuery, …
  - web services based on XML: SOAP, WSDL, UDDI, …
- **alternatives**
  - JSON, YAML, HDF5, ASN.1, ...
- **official source: www.w3.org**

Markup languages

- **"mark up" documents with human-readable tags**
  - content is separate from description of content
  - not limited to describing visual appearance
- **XML is a meta-language for markup**
  - language for describing grammar and vocabularies of other markup languages that deal with hierarchical textual data
  - element: data surrounded by markup that describes it

    ```
    <person>George Washington</person>
    ```
  - attribute: named value within an element

    ```
    <body bgcolor="green">
    ```
  - extensible: tags & attributes can be defined as necessary
  - strict rules of syntax: where tags appear, what names are legal, what attributes are associated with elements
  - instances are specialized to particular applications

    HTML: tags for document presentation

    XHTML: HTML with precise syntax rules

XML: eXtensible Markup Language

- **an extensible way to describe any kind of hierarchical data**
- **a notation for describing trees (only)**
- **each internal node in the tree is an element**
- **leaf nodes are either attributes or text**
- **"well formed": the instance is a tree**
  - everything balanced, terminated, quoted, etc.
- **"valid": satisfies syntactic rules given in a DTD or schema**
  - valid tags & attributes, proper order, right number, …
- **human-readable text only (Unicode), not binary**
  - can process with standard tools
  - independent of proprietary tools and representations
- **not a programming language**
  - XML doesn't do anything, just describes
  - programs read, process, and write it
- **not a database**
  - programs convert between XML and databases

XML vs HTML

- **HTML:**
  - tags describe presentation or appearance of content
  - not much if any semantic structure
  - set of valid tags and attributes is predefined and can't be changed
  - practically, browsers are very forgiving about invalid structure, unkonwn tags, etc.

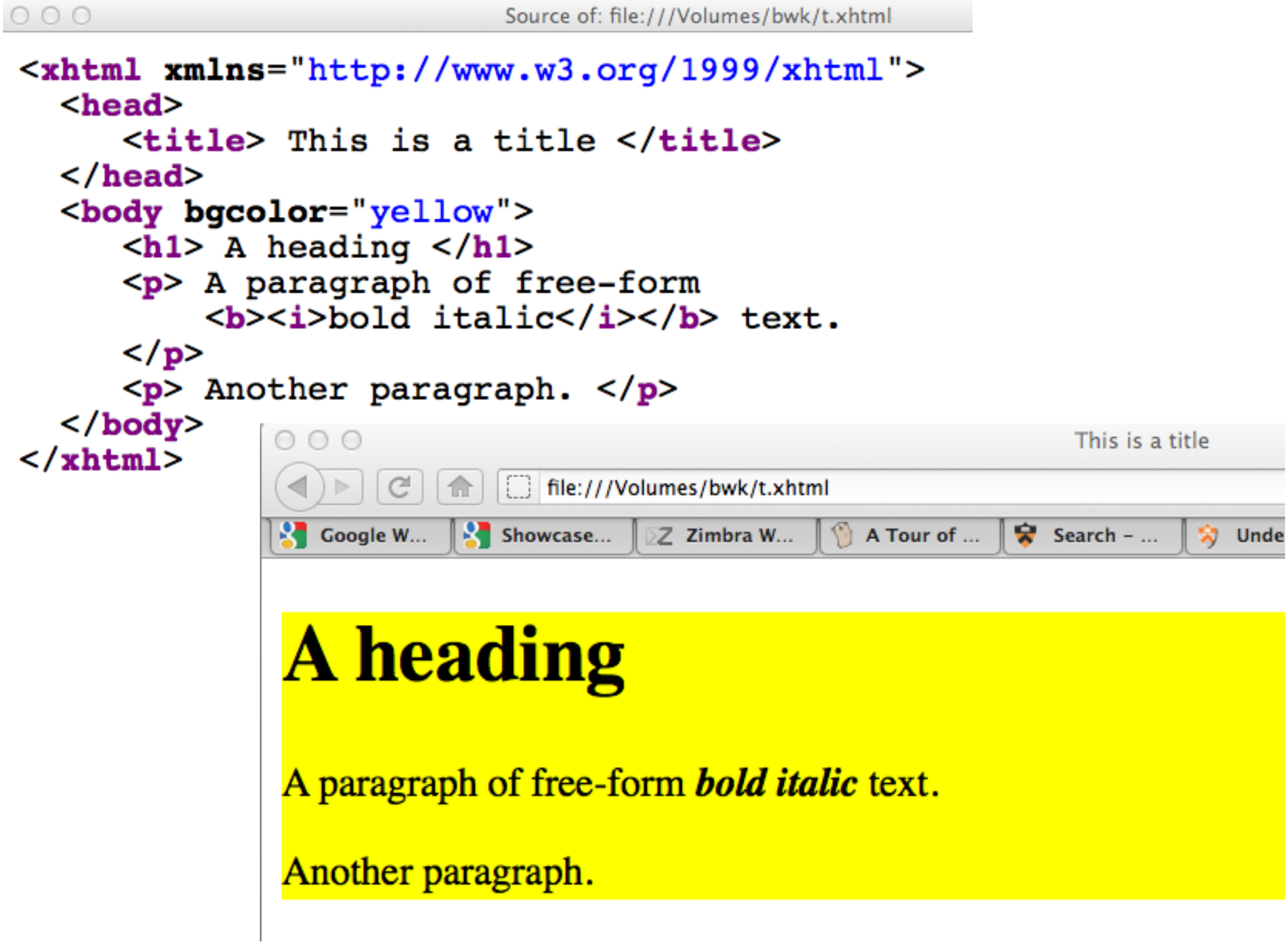
- **XML:**
  - tags describe semantic structure of the content
  - in general, no presentation or appearance aspect
  - can define own tags and attributes
  - can check the tags and attributes for validity

XML in use

- **two common kinds of use**
  - document-centric: ordinary text documents with markup
  - data-centric: representation and exchange of data with applications
- **XHTML**
  - an example of document-centric view
  - XHTML is HTML with more stringent rules
    - everything balanced and terminated and quoted; names are case sensitive

```
<xhtml xmlns="http://www.w3.org/1999/xhtml">
  <head>
     <title> This is a title </title>
  </head>
  <body bgcolor="yellow">
     <h1> A heading </h1>
     <p> A paragraph of free-form
         <b><i>bold italic</i></b> text.
     </p>
     <p> Another paragraph. </p>
  </body>
</xhtml>
```

XML as seen by browsers



```
<xhtml xmlns="http://www.w3.org/1999/xhtml">
  <head>
     <title> This is a title </title>
  </head>
  <body bgcolor="yellow">
     <h1> A heading </h1>
     <p> A paragraph of free-form
         <b><i>bold italic</i></b> text.
     </p>
     <p> Another paragraph. </p>
  </body>
</xhtml>
```

A heading

A paragraph of free-form ***bold italic*** text.

Another paragraph.

Why XML?

- **increasing use of web services**
  - too hard to extract semantics from HTML
  - closed and/or binary systems are too hard to work with, too inflexible
- **XML is open, non-proprietary**
- **text-based**
  - can see what it does
  - standard tools work on it
  - there are standard parsers, transformers, generators, etc.

- **simple, extensible**
  - existing vocabularies for important areas
  - can define new vocabularies for specific areas

- **most XML use is data-centric**
  - standard exchange format for web services
  - configuration info inside systems

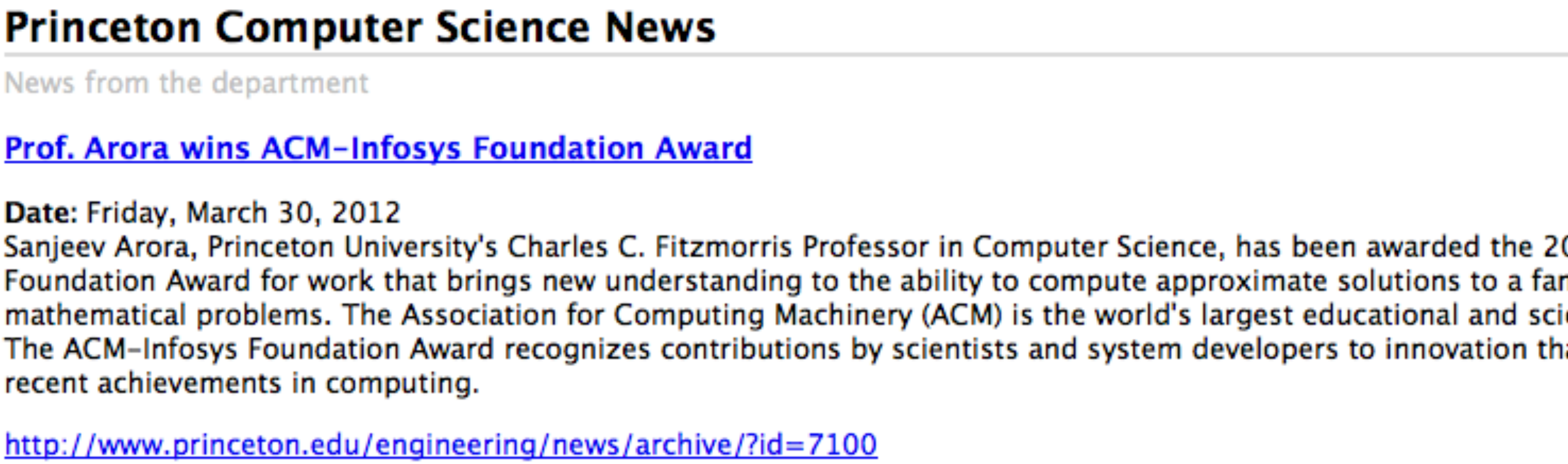
XML vocabularies and namespaces

- **a *vocabulary* is an XML description for a specific domain**
  - Schema
  - XHTML
  - RSS (really simple syndication)
  - SVG (scalable vector graphics)
  - MathML (mathematics)
  - EPUB (electronic book format)
  - Android screen layout
  - ...

- **namespaces**
  - mechanism for handling name collisions between vocabularies

```
<ns:some_tag> ... </ns:some_tag>
<ns2:some_tag> ... </ns2:some_tag>
```

RSS: Really Simple Syndication



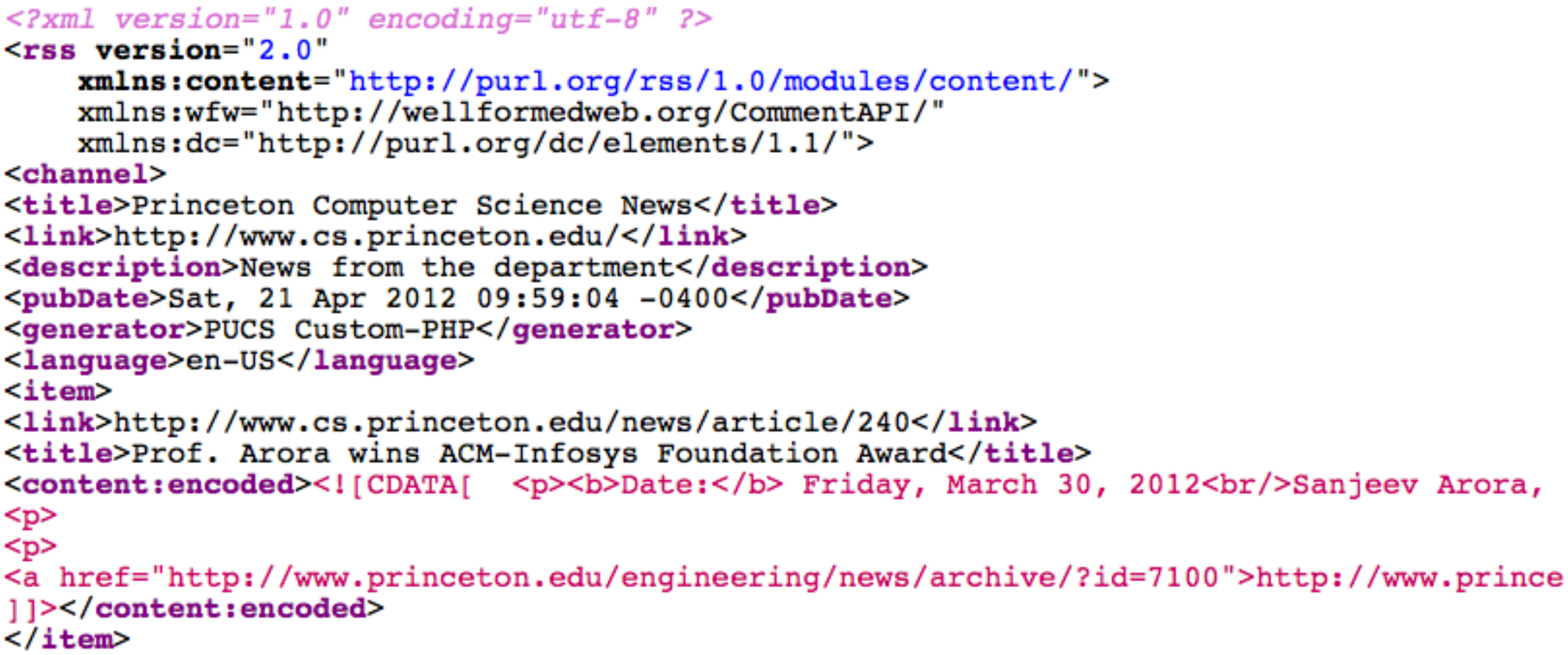
Princeton Computer Science News

News from the department

Prof. Arora wins ACM-Infosys Foundation Award

Date: Friday, March 30, 2012
Sanjeev Arora, Princeton University's Charles C. Fitzmorris Professor in Computer Science, has been awarded the 20
Foundation Award for work that brings new understanding to the ability to compute approximate solutions to a far
mathematical problems. The Association for Computing Machinery (ACM) is the world's largest educational and sci
The ACM-Infosys Foundation Award recognizes contributions by scientists and system developers to innovation tha
recent achievements in computing.

http://www.princeton.edu/engineering/news/archive/?id=7100



```
<?xml version="1.0" encoding="utf-8" ?>
<rss version="2.0"
    xmlns:content="http://purl.org/rss/1.0/modules/content/">
    xmlns:wfw="http://wellformedweb.org/CommentAPI/"
    xmlns:dc="http://purl.org/dc/elements/1.1/">
<channel>
<title>Princeton Computer Science News</title>
<link>http://www.cs.princeton.edu/</link>
<description>News from the department</description>
<pubDate>Sat, 21 Apr 2012 09:59:04 -0400</pubDate>
<generator>PUCS Custom-PHP</generator>
<language>en-US</language>
<item>
<link>http://www.cs.princeton.edu/news/article/240</link>
<title>Prof. Arora wins ACM-Infosys Foundation Award</title>
<content:encoded><![CDATA[  <p><b>Date:</b> Friday, March 30, 2012<br/>Sanjeev Arora,
<p>
<p>
<a href="http://www.princeton.edu/engineering/news/archive/?id=7100">http://www.prince
]]></content:encoded>
</item>
```

XML describes trees

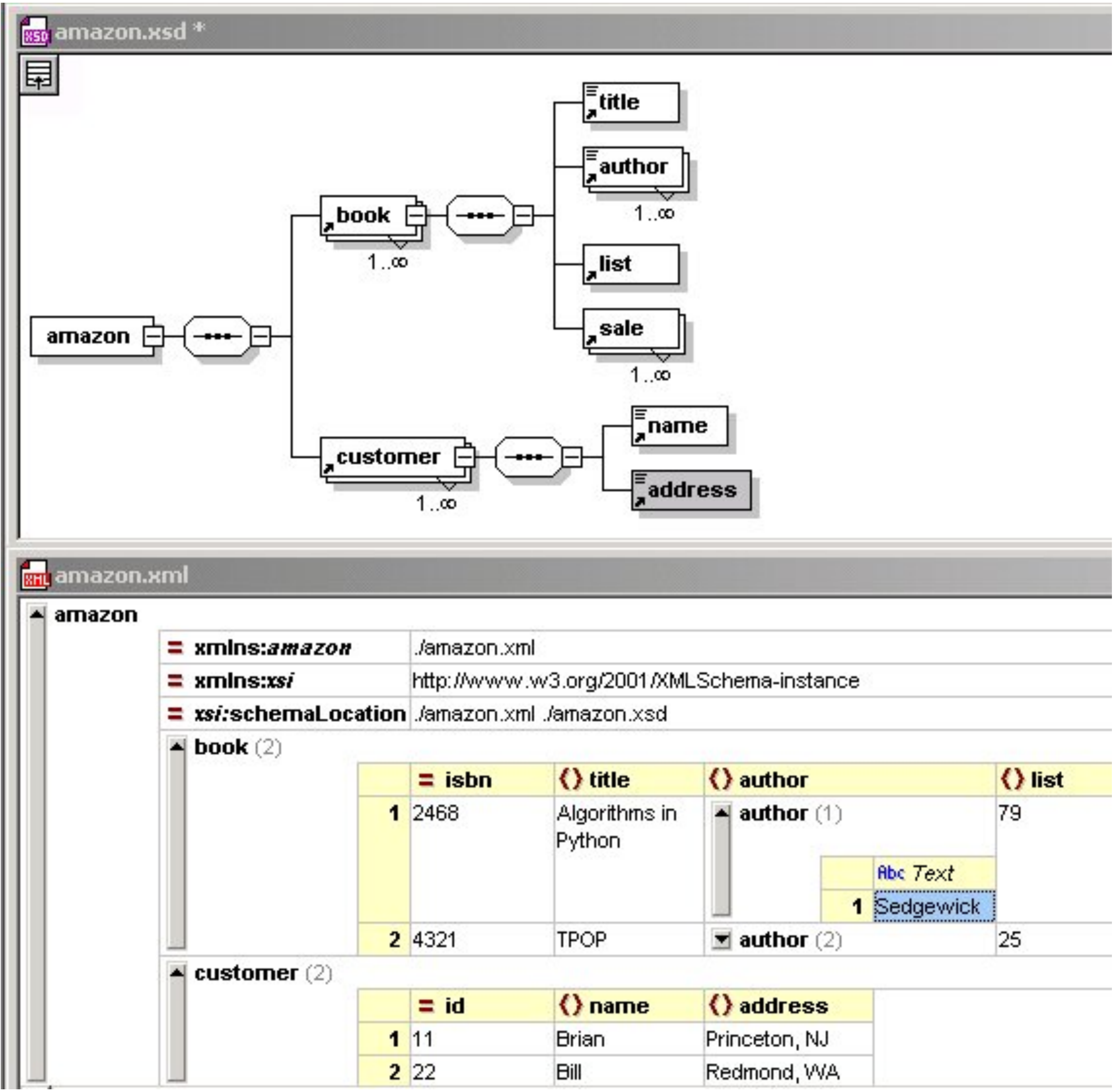
- **"well formed": it is a valid tree structure**
  - properly nested
  - syntactically correct
  - everything properly quoted
  - nothing about semantics or relationships among elements
- **"valid": well formed AND satisfies rules about what is legal**

- **DTD: document type definition**
  - (comparatively) simple pattern specification
  - not very powerful (no data types)
  - not written in XML syntax (needs separate tools)
- **Schema**
  - (comparatively) complicated specification
  - much stronger language for expressing structure

    sequencing and counting of complex types
  - built-in basic types like integer, double, string
  - can attach validation constraints to basic types

    ranges of integers, patterns of strings, etc.
  - written in XML, can apply all XML tools to it

XML tools / XMLSpy



amazon

= xmlns:*amazon*	./amazon.xml
= xmlns:*xsi*	http://www.w3.org/2001/XMLSchema-instance
= *xsi*:schemaLocation	./amazon.xml ./amazon.xsd

book (2)

	= isbn	() title	() author	() list
1	2468	Algorithms in Python	author (1) Abc *Text* 1 Sedgewick	79
2	4321	TPOP	author (2)	25

customer (2)

	= id	() name	() address
1	11	Brian	Princeton, NJ
2	22	Bill	Redmond, WA

XML processing by program

- **two basic kinds of parsers**

- **DOM (Document Object Model)**
  - read entire XML document into memory
  - create a tree
  - provide methods for walking/processing the tree

- **SAX (Simple API for XML)**
  - read through XML document

    nothing stored implicitly
  - call user-defined method for each document element

    callbacks

- **other processing tools**
  - XSLT (extensible stylesheet language for XML transformations)
  - XPath (query/filter language for XML)
  - XQuery (query language for XML
  - …

DOM: document object model

- **standard "language-independent" interface for manipulating structured documents**

- **allows dynamic access and modification**

- **methods for traversing tree and accessing nodes**
  - does not define any semantics other than
    - walking the tree
    - accessing elements
    - adding or deleting elements

- **implementations in Java, C++, Python, ...**

- **not as language-independent as might appear**
  - have to change a fair amount to change languages

DOM reader in Java

```
import java.io.*;
import org.w3c.dom.*;
import javax.xml.parsers.*;

public class domreader {
   public static void main(String[] args) {
      domreader r = new domreader(args[0]);
}

public domreader(String f) {
   try {
      DocumentBuilderFactory dbf =
            DocumentBuilderFactory.newInstance();
      // dbf.setValidating(true);
      DocumentBuilder b = dbf.newDocumentBuilder();
      Document doc = b.parse(f);
      Element root = doc.getDocumentElement();
      print_node(root, "");
   } catch (Exception e) {
      e.printStackTrace();
   }
}
```

DOM reader, page 2

```
void print_node(Node n, String pfx) {
   if (n != null && n.getNodeType() == Node.ELEMENT_NODE) {
      Node cn = n.getFirstChild();
      String s = "";
      if (cn != null) s = ((CharacterData)cn).getData();
      s = s.trim();
      System.out.println(pfx + n.getNodeName() + " [" + s + "]");
      print_attrs(n, pfx + "   ");
      print_children(n, pfx);
   }
}
void print_children(Node n, String pfx) {
   NodeList nl = n.getChildNodes();
   for (int i = 0; i < nl.getLength(); i++)
      print_node(nl.item(i), pfx + "   ");
}
void print_attrs(Node n, String pfx) {
   NamedNodeMap nnm = n.getAttributes();
   if (nnm != null) {
      for (int j=0; j < nnm.getLength(); j++)
         System.out.println(pfx + nnm.item(j).getNodeName() +
                   "=" + nnm.item(j).getNodeValue());
   }
}
```

SAX reader in Java

```
import java.io.*; import java.util.*;
import org.xml.sax.*;
import org.xml.sax.helpers.*;
import javax.xml.parsers.*;
public class sax extends DefaultHandler {
  int depth = 0;
  List<String> path = new ArrayList<String>();
  public static void main(String[] args) {
    sax r = new sax(args[0]);
  }
  public sax(String f) {
    try {
      SAXParserFactory spf = SAXParserFactory.newInstance();
      spf.setValidating(true);
      SAXParser sp = spf.newSAXParser();
      sp.parse(new File(f), this);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
  public void startDocument() { depth = 0; }
  public void endDocument() {
    if (depth != 0) System.out.printf("error: depth = %d at end\n", depth);
  }
```

```
public void startElement(String nsURI, String localname, String qualname, Attributes
  depth++;
  if (localname.equals("")) localname = qualname;
  path.add(localname);
  if (depth > 0) System.out.printf("\n");
  System.out.printf("%d %s", depth, join(path, "/"));
  if (attr != null) {
    for (int i = 0; i < attr.getLength(); i++) {
      String s = attr.getLocalName(i);
      if (s.equals("")) s = attr.getQName(i);
      System.out.println(s + "=" + attr.getValue(i));
    }
  }
}
public void endElement(String nsURI, String localname, String qualname) {
  if (localname.equals("")) localname = qualname;
  depth--;
  path.remove(depth);
}
public void characters(char buf[], int offset, int len) {
  String s = new String(buf, offset, len);
  s = s.trim();
  if (s.length() > 0) System.out.printf(" %s", s);
}
String join(List<String> ls, String sep) {
  String s = "";
  for (int i = 0; i < ls.size()-1; i++) s += ls.get(i) + sep;
  s += ls.get(ls.size()-1);
  return s;
}
```

Tags vs Attributes

- **not always clear whether to use a tag or an attribute**

- **heuristics:**
  - tags should contain content
  - attributes should describe content
  - in general, favor tags over attributes

- **but attributes are more compact, perhaps easier to read**

```
<ProductName
      Code="198405" System="RxNorm"
      Strength="100" Unit="mg" Form="tablet">
   Ibuprofen
</ProductName>
```

Web services

- **Web service:**
  - interface that describes a set of operations
  - that are accessible by network
  - using XML or other standard protocols
- **SOAP (simple object access protocol)**
  - protocol for exchanging XML messages via HTTP
- **WSDL (web services description language)**
  - XML-based descriptions of web services
- **UDDI (universal description, discovery & integration)**
  - XML-based registry for public web services
- **RSS & Atom (XML-based syndication formats)**
  - simpler, lighter weight than SOAP
- **REST (representational state transfer)**
  - transfer information via HTTP
  - not XML, no additional layers

SOAP

- **"an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined datatypes, and a convention for representing remote procedure calls and responses." (W3C)**
- **communication protocol for invoking methods on servers, services, components and objects**
  - language independent "wire protocol"
  - COM, CORBA, etc., can use it
- **XML vocabulary for defining parameters, return values and exceptions**
- **uses HTTP to carry info**
  - interface & method names included in header
  - supposed to be checked by recipient
- **formalizes use of XML and HTTP for invoking remote methods**