

# 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, Atom, SVG, MathML, Schema, ...
  - validation: Schema, DTD
  - parsers: SAX, DOM
  - processing XML documents: XPath, XSLT, XQuery
  - web services based on XML: SOAP, WSDL, UDDI, ...
- **alternatives** (subset of a huge number)
  - JSON, YAML, HDF5, ASN.1, ...
- **sources** (subset of a huge number)
  - [www.w3.org](http://www.w3.org) (official)
  - [www.xml.com](http://www.xml.com) (O'Reilly)

## Markup languages

- **"mark up" documents with human-readable tags**
  - content is separate from description of content
  - not limited to describing visual appearance
- **SGML and XML are meta-languages for markup**
  - languages for describing grammar and vocabularies of other languages
  - 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 is compatible with SGML**
  - a simplified, inter-operable form

# XML: eXtensible Markup Language

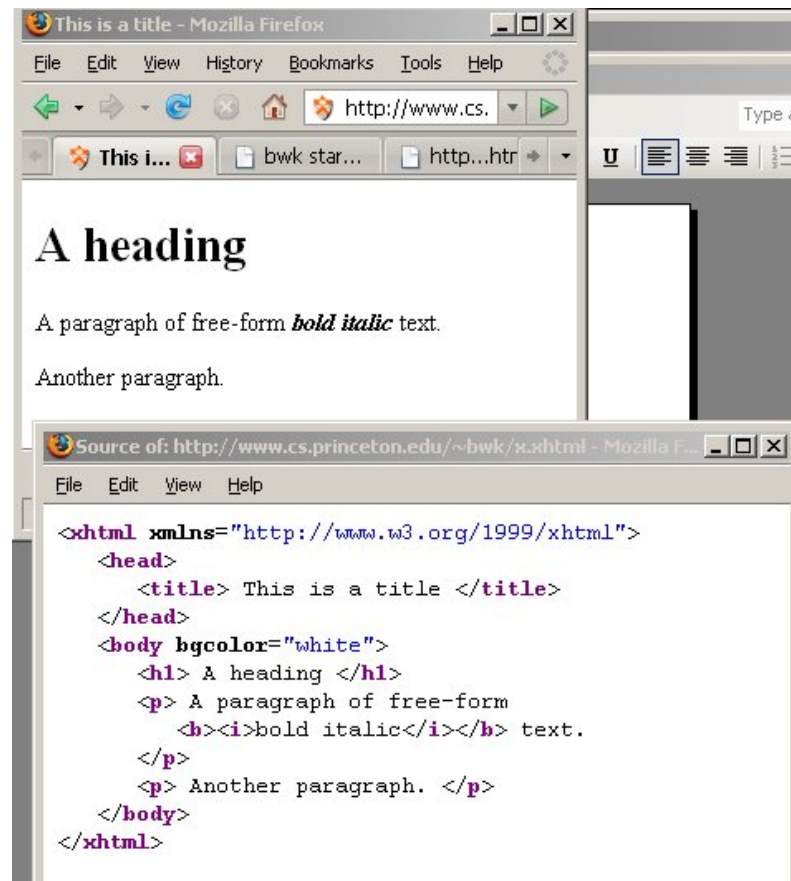
- **an extensible way to describe any kind of data**
- **a notation for describing trees (only)**
- **each internal node in the tree is an element**
- **leaf nodes are either attributes or text**
- **"well formed": the instance is a tree**
  - everything balanced, terminated, quoted, etc.
- **"valid": satisfies syntactic rules given in a DTD or schema**
  - valid tags & attribs, 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 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="white">
    <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



## 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)
- SMIL (markup for multi-media presentations)
- ...

- **namespaces**

- mechanism for handling name collisions between vocabularies

```
<ns:tag> ... </ns:tag>
```

```
<ns2:tag> ... </ns2:tag>
```

## RSS: Really Simple Syndication

### Princeton Computer Science Events

---

Events happening in the department

[Monday, April 19, 2010: David Sontag on "Approximate Inference"](#)

**Date:** Monday, April 19, 2010

**Time:** 4:30 PM

**Guest:** David Sontag

**From:** MIT

**Title:** Approximate Inference in Graphical Models using LP Relaxations

```
<?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 Events</title>
    <link>http://www.cs.princeton.edu/</link>
    <description>Events happening in the department</description>
    <pubDate>Sun, 18 Apr 2010 14:16:53 -0400</pubDate>
    <generator>PUCS Custom-PHP</generator>
    <language>en-US</language>
  </channel>
  <item>
    <link>http://www.cs.princeton.edu/events/event/265</link>
    <title>Monday, April 19, 2010: David Sontag on "Approximate Inference in G
    <content:encoded><![CDATA[ <p><b>Date:</b> Monday, April 19, 2010<br/><b>!
  </item>
```

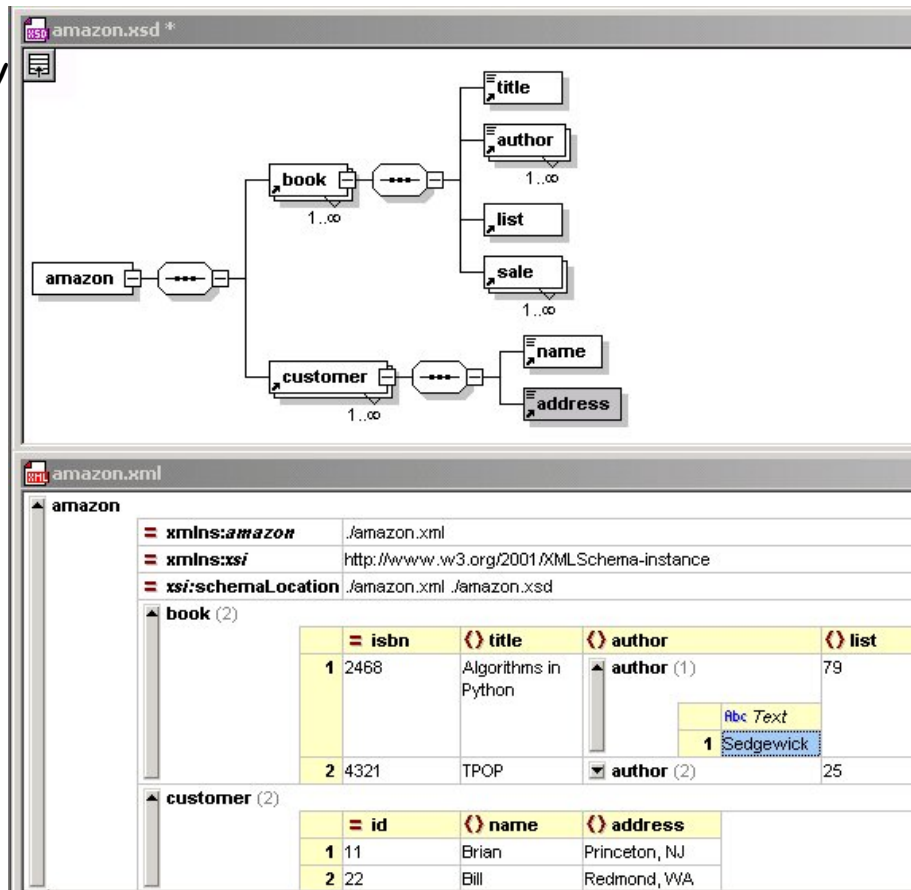
# XML describes trees

- **"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

## Example schema (a small part)

```
<?xml version="1.0" encoding="UTF-8"?>
<!--W3C Schema generated by XMLSPY -->
<xs:schema xmlns:xs=
  "http://www.w3.org/2001/XMLSchema">
  <xs:element name="amazon">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="book"
          maxOccurs="unbounded"/>
        <xs:element ref="customer"
          maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="amazon"
        type="xs:string" use="required"/>
    </xs:complexType>
  </xs:element>
  <xs:element name="book">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="title"/>
        <xs:element ref="author" maxOccurs="unbounded"/>
        <xs:element ref="list"/>
        <xs:element ref="sale" maxOccurs="unbounded"/>
      </xs:sequence>
      <xs:attribute name="isbn" use="required">
        <xs:simpleType>
          </xs:simpleType>
        </xs:attribute>
      </xs:complexType>
    </xs:element>
  </xs:schema>
```

## XML tools / XMLSpy



## 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++, VB, etc.**
- **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, Attribute
    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;
}
}

```

## 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**