I. XML Basics

University of California Extension
Sunnyvale, June 10, 1999

Jon Bosak
Sun Microsystems

The Fundamentals

What is XML?

- Extensible Markup Language
- An activity of the World Wide Web Consortium (W3C) organized and led by Sun Microsystems
- Objective: move the Web to its next stage of evolution by adapting existing ISO standards for markup, linking, and formatting

Primary effects:

1. Will create new data-centric Web applications
2. Will fundamentally change publishing on the web and publishing in general

What made XML necessary?

Two aspects of Web evolution demanded a technology beyond HTML.

- Internationalized electronic publishing
  - Platform-independent
  - Language-independent
  - Media-independent
- New data-centric Web applications
  - Database exchange
  - Distribution of processing to clients
  - Client-side manipulation of views into the data
  - Customization of information by intelligent agents
  - Management of document collections

What's wrong with HTML?

- HTML was optimized for easy learning
  - One tag set for all applications
  - Predefined semantics for each tag
  - Predefined data structures
  - No formal validation
- HTML trades power for ease of use
- HTML is well suited to simple applications, but poorly suited to more demanding applications
  - Large or complex collections of data
  - Data that must be used in different ways
  - Data with a long life cycle
  - Data intended to drive scripts or Java applets

What does XML provide?

XML provides key features needed for a new generation of Web applications:

- **Extensibility:** Users can define new tags as needed
- **Structure:** Hierarchical data can be modeled to any level of complexity
- **Validation:** Data can be checked for structural correctness
- **Media independence:** The same content can be published in multiple media

Why did Sun invest in XML?

1. In industry, we knew from electronic publishing experience that HTML would not work for publishing in the general case.
2. We also knew that future Web applications would require a method of encoding that could drive arbitrarily complex distributed processes.
3. It was clear that if an open standard like XML was not created, HTML would be replaced by a more powerful **binary proprietary format.**

Strategically, we had to have XML in order to keep Web data open and portable. We needed XML to do for data what Java does for programs.

Current status

- The XML 1.0 Rec is being widely deployed
- XML is being widely adopted as a framework for the definition of domain-specific languages
- It is now generally agreed that Web content will be managed using standards based on XML

Key predictions:

1. XML will be the basis for future Web standards.
2. XML will become the universal format for data exchange in heterogenous environments.
3. XML will almost certainly become the basis for international publishing.
4. The combination of XML and XSL may replace all existing word processing and desktop publishing formats.

Key sources of information about XML

- **The W3C activity:**

  http://www.w3.org/XML/

- **Standards and drafts:**

  http://www.w3.org/TR/

- **Markup technology in general:**

  http://www.oasis-open.org/cover/

The XML Family of Standards

Meet the family

The XML family of languages moves the web to a new level of evolution suitable for electronic commerce and other industrial-strength applications.

- **XML** (Extensible Markup Language): A subset of SGML (ISO 8879) designed for easy implementation
  - Will replace HTML markup in industrial contexts
- **XLink/XPointer**: A set of standard hypertext mechanisms based on HyTime (ISO/IEC 10744) and the Text Encoding Initiative (TEI)
  - Will replace HTML linking in industrial contexts
- **XSL** (Extensible Stylesheet Language): A standard stylesheet language for structured information based on DSSSL (ISO/IEC 10179) and CSS
  - Will replace CSS in industrial contexts

XML itself

- A simplified subset of SGML (ISO 8879)
  - Very powerful -- no limits on namespace or structural depth
  - But easy to implement and small enough for Web browsers
- Not a language but a metalanguage
  - Designed to support the definition of an unlimited number of vertical-market languages for specific industries
  - All XML languages can be processed by a single lightweight parser built into every Web browser

XML tag languages

XML allows industries to design specific tag languages to solve specific problems.

Examples featured in Robin Cover's SGML/XML News page in one recent 30-day period (3/15 to 4/15, 1999):

- SVG (Scalable Vector Graphics)
- XMLNews (for the news industry)
- XCI (XML Court Interface)
- DocBk XML (for software documentation)
- XMI (XML Metadata Interface Format -- OMG)
- WAP (Wireless Application Protocol)
- SIF (Schools Interoperability Framework)

Key: An unlimited number of domain-specific tag languages can all be processed by a single parser.

XML in isolation

- "Syntax, not semantics"
  - Tags have no predefined meaning
  - XML by itself conveys only content and structure, not presentation or behavior (unlike HTML)
- There are important applications for XML alone: interprocess communication, object serialization, metadata, database exchange
- But associating **presentation or behavior** with XML requires additional mechanisms
  - Downloadable programs, applets, or scripts designed for a specific tag set (grammar)
  - Tag-sensitive components (e.g., Java beans)
  - Industry agreements on the processing of specific grammars (example: HTML)
  - Stylesheets (XSL or CSS)

Classical XML

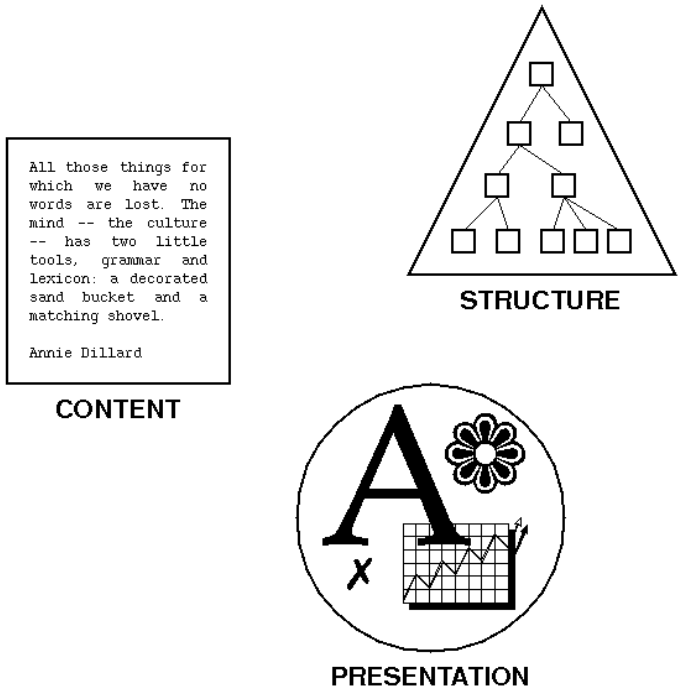
What's a document?

A document is data that you can **read.**

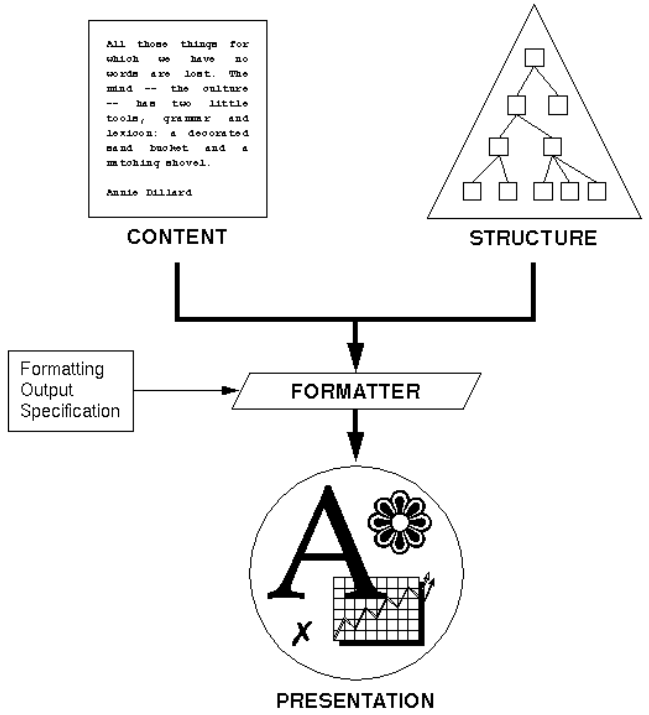
Documents are a **superset** of data.

The basic problem with documents is that we need to display them in lots of **different forms**. This is the problem that XML and SGML were originally designed to solve.

Basic document analysis



Structured publishing



XML allows you to specify the content and structure of a document in a way that lets you generate particular presentations as needed.

XML in one slide

- Legal XML documents are called **well-formed**
- A well-formed document describes a **logical tree**
- If a well-formed document conforms to an optional set of constraints (a DTD), it is also **valid**

A well-formed XML document:

```
<greeting type="friendly">Hello, world!</greeting>
```

A valid XML document:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE greeting [
  <!ELEMENT greeting (#PCDATA)>
  <!ATTLIST greeting type (friendly | unfriendly)
                                    "friendly" >
]>
<greeting>Hello, world!</greeting>
```


Proof of concept: this presentation

(These are links in the online version.)

- The XML source from which this presentation was produced
- The optional XML DTD used to validate the XML source
- The DSSSL style sheet for the HTML used in the online version
- The DSSSL style sheet for the RTF used in the printed version
- The Jade DSSSL engine used to produce both the HTML and RTF files
- An RTF version of this presentation produced by Jade
- A PostScript version of this presentation made from the RTF file
- A PDF version of this presentation made from the PS file

Lessons from the proof of concept

- Media-independent publishing works!
- HTML can handle the online version (for the moment), but not the print version
- The language for formatting specifications (stylesheets) must support **structural transformation** as well as formatting

Summary of classical XML

Separating content and structure from presentation and behavior makes possible

- Reusable information
- Media-independent publishing
- One-on-one marketing
- Intelligent downstream document processing
- Large-scale information management

Internationalization

XML and Unicode

- XML has been based on Unicode from Day One
  - There is nothing in an XML file but Unicode characters
  - Unicode is used for both content and markup (so you can mix languages, even in tag names)
- XML tools **must** support both the UTF-8 and UTF-16 encodings of Unicode
  - UTF-8: 1-5 bytes; Latin-1 is upward-compatible
  - UTF-16: 2 bytes; fixed overhead
- The widespread adoption of XML for data management and electronic commerce will probably make Unicode support universal

Example: an international bookstore

```
<?xml version="1.0"?>
<!DOCTYPE 書籍カタログ [
  <!ELEMENT 書籍カタログ (書籍)+ >
  <!ELEMENT 書籍 (書名,著者,出版社,(定価 | 在庫数)) >
  <!ATTLIST 書籍      xml:lang CDATA #REQUIRED >
  <!ELEMENT 書名      (#PCDATA) >
  <!ELEMENT 著者      (#PCDATA) >
  <!ELEMENT 出版社    (#PCDATA) >
  <!ELEMENT 定価      (#PCDATA) >
]>

<書籍カタログ>
  <書籍 xml:lang="JP">
    <書名>XML入門</書名>
    <著者>村田、門馬、荒井</著者>
    <出版社>日本経済新聞社</出版社>
    <定価>2800</定価>
  </書籍>
  <書籍 xml:lang="EN">
    <書名>Developing SGML DTDs</書名>
    <著者>E. Maler and J. el Andaloussi</著者>
    <出版社>Prentice Hall</出版社>
    <定価>50</定価>
  </書籍>
</書籍カタログ>
```


With stylesheet for Japanese

XML入門

著者 村田、門馬、荒井

出版社 日本経済新聞社

定価 2800円

Developing SGML DTDs

著者 E. Maler and J. el Andaloussi

出版社 Prentice Hall

定価 50ドル

With stylesheet for English

XML入門

Author: 村田、門馬、荒井

Publisher: 日本経済新聞社

Price: ¥2800

Developing SGML DTDs

Author: E. Maler and J. el Andaloussi

Publisher: Prentice Hall

Price: $50

Source files for the bookstore example

(These are links in the online version.)

- The UTF-16 XML source from which the different versions were produced
- The UTF-16 DSSSL style sheet used to produce the version for the reader of Japanese
- The UTF-16 DSSSL style sheet used to produce the version for the reader of English
- The Jade DSSSL engine used to produce RTF files from the source and the style sheets
- The UTF-16 RTF file for the reader of Japanese (font association done in Word 97)
- The UTF-16 RTF file for the reader of English (font association done in Word 97)

Lessons from the example

- The catalog example shows that the distinction between data exchange and publishing is ultimately an artificial one (the same source would also be used to create the printed catalog)
- The rendition in each case occurs **on the web client**
- The database owner can publish **a single data stream** to the entire world
- Consider the alternative:
  - Generation of a different HTML output stream for **every possible** user and target platform
  - Much greater load on the server
  - No user autonomy

Namespaces

The naming of names

- In electronic commerce, XML documents will be assembled on the fly from a wide variety of sources using different tag vocabularies (DTDs)
- Must prevent collisions between elements (or attributes) with the same name but different meanings
  - For example, the element <RING> would have very different meanings in a jewelry catalogue, a chemistry textbook, and a mathematical journal
- Must also allow re-use of common data elements (dates, currencies, measurements) across different XML tag languages
- Ultimately, we will need a system for associating **meanings** with XML components
- XML Namespaces (http://www.w3.org/TR/) is a small first step toward solving this problem

The concept of the XML namespace

- An XML *namespace* is a collection of XML element and/or attribute names that are guaranteed to be *unique*
- Basic trick: use DNS (Domain Name Service) to ensure uniqueness

DNS is the service that controls the ownership of domain names. It also provides the mechanism whereby names are resolved to actual resources, *but DNS resolution is not necessary to make XML namespaces work.*

URI + name=unique name

Here the element name "price" is not unique:

```
<x>
  <price units='Euro'>
    32.18
  </price>
</x>
```

Prefix the element name with a URI such as "http://ecommerce.org/schema"; now the name is unique (although verbose and syntactically illegal):

```
<x>
  <{http://ecommerce.org/schema}price units='Euro'>
    32.18
  </{http://ecommerce.org/schema}price>
</x>
```


The namespace prefix

By substituting a *namespace prefix* for the URI we get a structure that is both elegant and legal:

```
<x xmlns:edi='http://ecommerce.org/schema'>
  <edi:price units='Euro'>
    32.18
  </edi:price>
</x>
```

Namespace scoping ensures that "edi:" means the same as "{http://ecommerce.org/schema}" only upon and within the element <x> on which it is declared.

Important things to remember about namespaces

1. Namespace prefixes are just **temporary placeholders** for the current namespace URI.

   There are no standard prefixes!

2. A namespace URI does not necessarily point to a web resource (although it may).
3. If there is a resource, it is as likely to be a prose description as a machine-processable schema.
4. Namespace scoping is cool but complicated.
5. Namespaces make traditional DTD validation highly problematic if not downright useless. The solution to this lies in the XML schema work.
6. We need much more namespace implementation experience before this technique can be considered fully cooked.