Introduction to XML

# XML: basic elements

# XML

**"Trying to wrap your brain around XML is sort of like trying to put an octopus in a bottle. Every time you think you have it under control, a new tentacle shows up. XML has many tentacles, reaching out in all directions. "**

*(Dick Baldwin)*

```
<book>
    <chap>
        Text for Chapter 1
    </chap>
    <chap>
        Text for Chapter 2
    </chap>
</book>
```

# What is XML?

eXtensible Markup Language, or XML for short, is a new technology for web applications.

XML is a World Wide Web Consortium standard that lets you create your own tags.

**XML is not a single technology, but a group of related technologies that continually adds new members**

XML is a *lingua-franca* that simplifies business-to-business transactions on the web.

# Vendor independence in the data-formatting context

**"Other successful Internet technologies let people run their systems without having to take into account another company's own computer systems, notably:**
**TCP/IP for networking,**
**Java for programming,**
**Web browsers for content delivery.**
**XML fills the data formatting piece of the puzzle."**

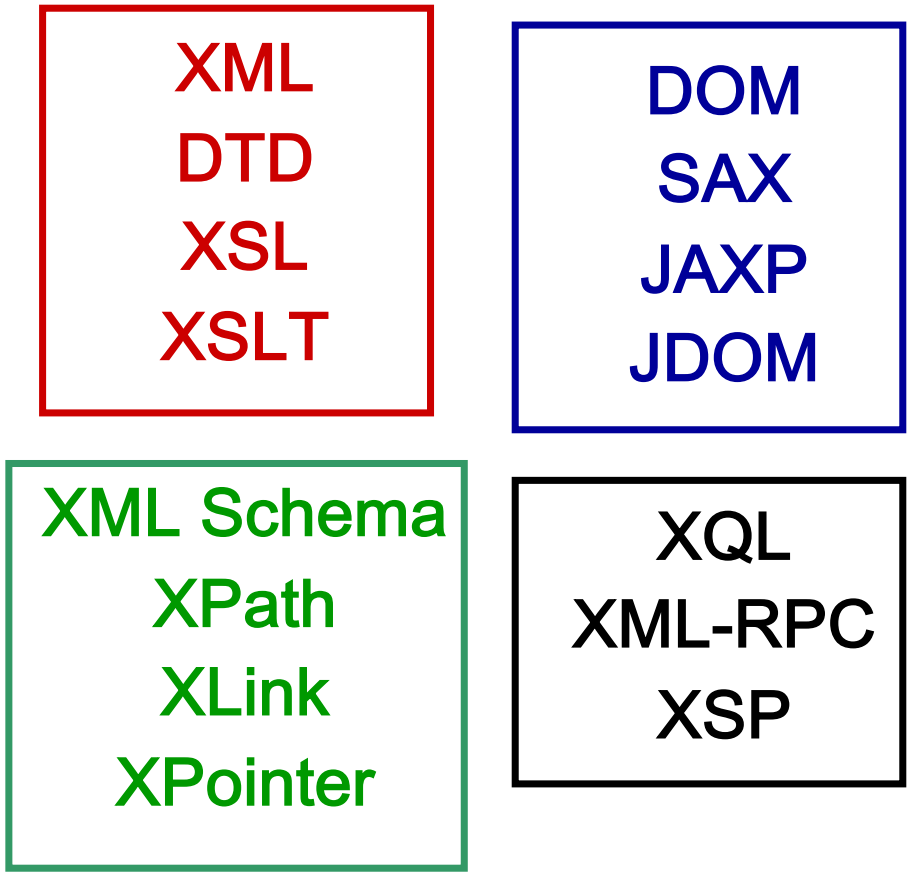
**"These technologies do not create dependencies. It means you can build solutions that are completely agnostic about the platforms and software that you use."**

**Phipps, IBM's chief XML and Java evangelist**

# XML Jargon

- Computer people are the world's worst at inventing new jargon.
- XML people seem to be the worst of the worst in this regard.

*(Dick Baldwin)*



# XML applications

Semantic Web

**RDF (Resource Description Framework), OWL, Topic Maps**

Web Services

**SOAP, UDDI, WSDL, XML-RPC**

Configuration files

# XML roots: SGML

## What is SGML

SGML is an ISO standard (ISO 8879:1986) which provides a formal notation for the definition of generalized markup languages. SGML is not a language in itself. Rather, it is a metalanguage that is used to define other languages.

# SGML: the three parts

An SGML document is really the combination of three parts. Let's refer to the parts as files (but they don't have to be separate physical files).

One file contains the content of the document (words, pictures, etc.). This is the part that the author wants to expose to the client.

A second file is the DTD that defines the accepted syntax.

A third file is a *stylesheet* that establishes how the content that conforms to the DTD is to be rendered on the output device. This is how the author wants the material to be presented to the client.

# HTML versus SGML

HTML implements some of the concepts derived from SGML but in effect the DTD and the Style Sheet are hard-coded into the browser software.

Because each browser manufacturer has some flexibility in implementing the intended style, the same document will sometimes look different when rendered with two different browsers. This is a (wanted) shortcoming of HTML.

Web page designers are constantly faced with the problem of designing workarounds to compensate for the deficiencies in some versions of some browsers being used to view the page.

# SGML - HTML

## What the world needs now is...

What the Web community needs is an approach where a standard browser is simply a rendering engine that validates a document according to a given DTD and renders it according to a given stylesheet.

## A package deal

The combination of the document, the DTD, and the stylesheet would constitute a package delivered by a server to the browser. The author of the document would provide the DTD and the stylesheet in addition to the data to be rendered. Then the author could be more confident that it would be rendered properly, especially for complex data.

# SGML – HTML - XML

## The two extremes

With HTML, the DTD and the stylesheet are essentially hard-coded into the browser.

With SGML, the processor requires both a DTD and a stylesheet.

## XML, the middle ground

With XML, the DTD is optional but the stylesheet (or some processing mechanism that substitutes for a stylesheet) is required.

# XML: element, content, and attribute

**What is an element?**

**An element is a sequence of characters that begins with a *start tag* and ends with an *end tag* and includes everything in between.**

```
<chap number="1">Text for Chapter 1</chap>
```

**What is the content?**

**The characters in between the tags (rendered in green in this presentation) constitute the CONTENT.**

# XML: element, content, and attribute

**An element may include optional attributes**

**The *start tag* may contain optional *attributes*. In this example, a single *attribute* provides the number value for the chapter.**

```
<chap number="1">Text for Chapter 1</chap>
```

**The characters rendered in blue in the above element constitute an attribute.**

**The term *attribute* is a commonly used term in computer science and usually has about the same meaning, regardless of whether the discussion revolves around XML, Java programming, or database management: Attributes belong to things, or things have attributes.**

# XML: tree structure

An XML document must have a root tag.

An XML document is an information unit that can be seen in two ways:

As a linear sequence of characters that contain characters data and markup.

As an abstract data structure that is a tree of nodes.

# XML: additional elements

An XML document can contain:

Processing Instructions (PI): **<? … ?>**

Comments **<!-- … -->**

When a XML document is analyzed, character data within comments or PIs are ignored.

The content of comments is ignored, the content of PIs is passed on to applications.

# XML: CDATA sections

**Note:** the element content that are going to be parsed are called **PCDATA**

An XML document can contain sections used to escape character strings that may contain elements that you do not want to be examined by your XML engine, e.g. special chars (<) or tags:

CDATA sections **<![CDATA[ … ]]>**

When a XML document is analyzed, character data within a CDATA section are not parsed, by they remain as part of the element content.

```
<java>
<![CDATA[
    if (arr[indexArr[4] ]>3) System.out.println("<HTML>");
]]>
</java>
```

Avoid having ]]> in your CDATA section!

# Well formed documents

**All XML documents must be well-formed**

**XML documents need not be valid, but all XML documents must be well-formed.**

**(HTML documents are not required to be well-formed)**

**There are several requirements for an XML document to be well-formed.**

# Well formed documents

Caution: XML is case sensitive

**Start and end tags are required**

**To be well-formed, all elements that can contain character data must have both start and end tags.**

**(Empty elements have a different requirement: see later.)**

**For purposes of this explanation, let's just say that the *content* that we discussed earlier comprises character data.**

**Elements must nest properly**

**If one element contains another element, the entire second element must be defined inside the start and end tags of the first element.**

# Well formed documents

**Dealing with empty elements**

**We can deal with empty elements by writing them in either of the following two ways:**

```
<book></book>
<book/>
```

**You will recognize the first format as simply writing a start tag followed immediately by an end tag with nothing in between.**

**The second format is preferable**

**Empty element can contain attributes**

**Note that an empty element can contain one or more attributes inside the start tag:**

```
<book author="eckel" price="$39.95" />
```

# Well formed documents

**No markup characters are allowed**

**For a document to be well-formed, it must not have some characters (entities) in the text data: < > “ ‘ &.**

**If you need for your text to include the < character you can represent it using < or < or &#x3C instead.**

**All attribute values must be in quotes (apostrophes or double quotes).**

**You can surround the value with apostrophes (single quotes) if the attribute value contains a double quote. An attribute value that is surrounded by double quotes can contain apostrophes.**

# Logical structure of an XML document

XML declaration (optional, but if present MUST be the first element)

<?xml version=‘1.0’ encoding=‘utf-8’>

Optional DTD declaration

Optional comments and Processing Instructions

The root element’s start tag

All other elements, comments and PIs

The root element closing tag