

OVERVIEW OF XML TECHNOLOGIES

HappyJack Tech Talk
April 2, 2008

OVERVIEW

- What is XML?
- What is it good for?
- Key XML technologies
 - Specifying dialects
 - Parsing
 - Translating

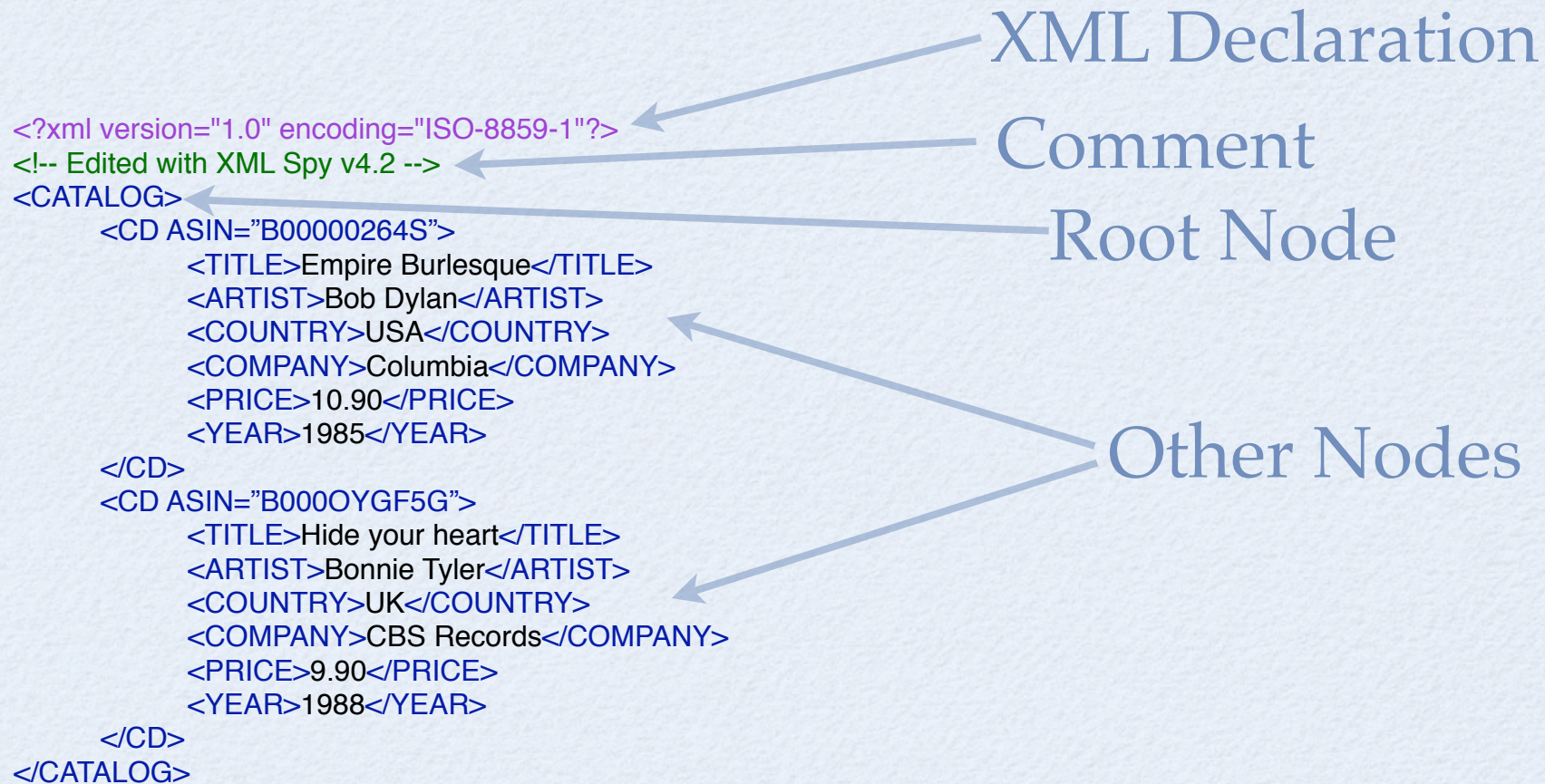
WHAT IS XML?

- XML is (basically) a standard syntax for representing trees
- It is written in Unicode (ASCII for English documents), so you can actually read it
- It looks like HTML
 - Actually, both HTML and XML are instances of SGML

XML DOCUMENT

- An XML document must start with an XML declaration
- It can include comments, XML processing instructions, etc.
- It also has (exactly) one XML node (the root node), which can have other nodes as children
- Note: XML is case-sensitive (unlike HTML)

XML DOCUMENT



XML COMPONENTS

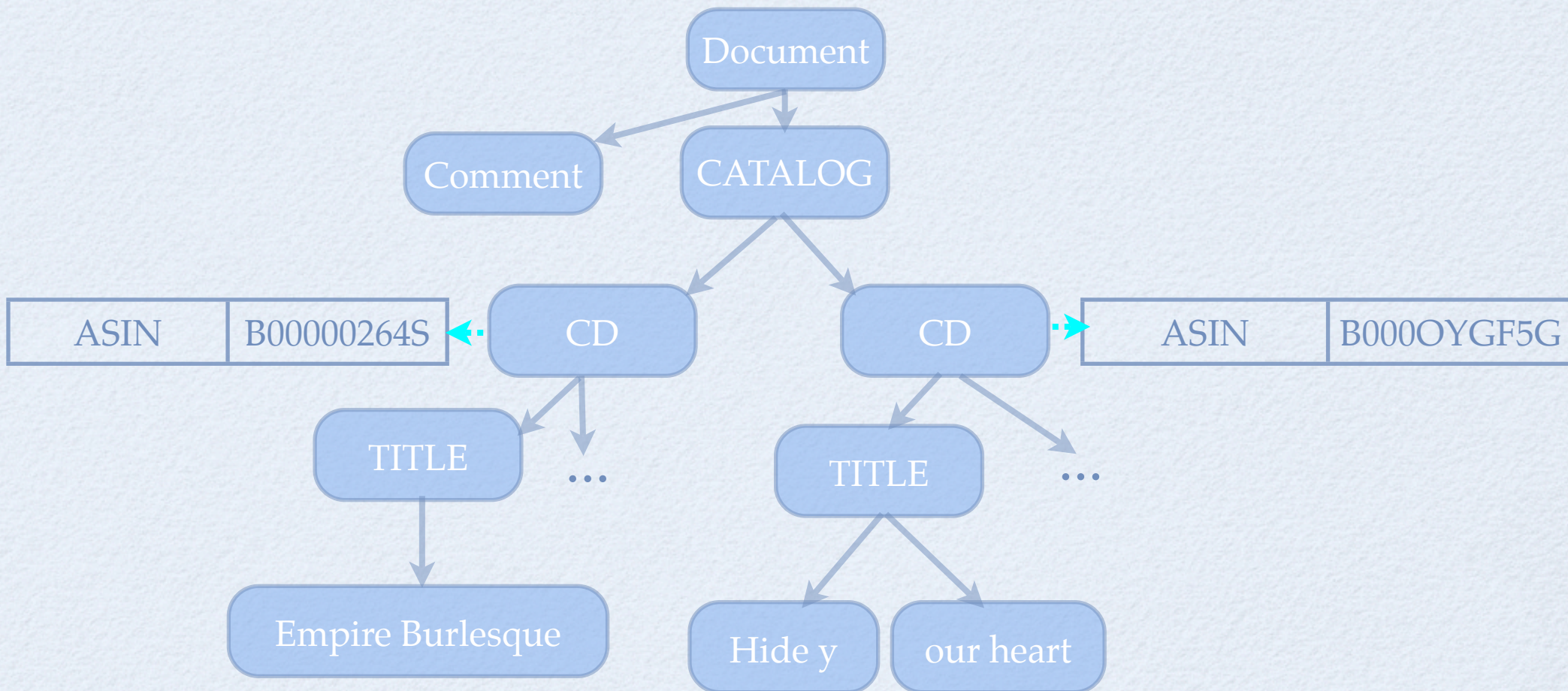
- **XML declaration** – specifies the version and character encoding (e.g., UTF-8 or ISO-8859-1)
- **Processing instructions** – provide arbitrary “extra” information about the document (e.g., printing options)
- **Comments** – who uses those, anyway?

XML COMPONENTS

- **Tags** – enclosed between `< ... >`
- **Elements** – include everything from the start tag to the end tag
- **Attributes** – `key="value"` pairs specified in the start tag of an element
- **Text nodes** – special type of node that contains (part of) text inside an element

DOCUMENTS AS TREES

- An XML document is really just a tree



USES OF XML

- XML documents fall into two broad categories
- **Document-centric** – similar to HTML, these documents have mostly text that is intended to be read by a human
- **Data-centric** – encode records, tables, i.e., programming objects

DOCUMENT-CENTRIC XML

- This was the original goal of SGML
- The idea is that the document contains “markup” so that the intent of words, phrases, etc., can be specified
 - e.g., “this phrase is a book title”
- A publisher can typeset the information in whatever style they want

DOCUMENT-CENTRIC XML

- The technical challenge is to *transform* the XML into some other format (e.g., PDF)
- Adobe's FrameMaker product has a nice front-end to SGML- or XML-based document formats (e.g., DocBook)
- The big problem with this style of XML is that authors aren't great at tagging all of their words appropriately!

DATA-CENTRIC XML

- Data-centric XML documents represent
 - a record
 - a table of records
 - an object (possibly pointing to other objects)
 - a tree
- XML by geeks, for geeks!

DATA-CENTRIC XML

- Data-centric XML can be used to
 - store information, e.g., config files in many programs (Apple preferences, Eclipse settings, Ant files, deployment descriptors, ...)
 - exchange information from one program to another, e.g., RSS, SOAP
- The CD catalog example was data-centric

XML DIALECTS

- XML allows you to use arbitrary tags, attribute names, etc.
- In both the document- and data-centric world, it is important for an application to know what sort of XML documents it can expect
 - E.g., what tag denotes book title?
 - E.g., where is the input argument?

XML DIALECTS

- An XML dialect lets you specify
 - what tags a document may have (e.g., `<p>`, `<body>`, `<table>`, ...)
 - what attributes elements can have (e.g., `<body>` can have an `onLoad` attribute)
 - what nesting structures are allowed (e.g., the top-level must be `<html>`, then `<body>`, etc.)

XML SCHEMAS

- You can specify the dialect using a number of different notations
 - DTD – the original solution for SGML, DTDs have their own wooly syntax. It's mostly outdated today, though some tools require it
 - XML Schema – a W3C standard for specifying dialects in an XML syntax. You'll need to know this works, though nobody likes it....

XML SCHEMAS

- More XML Schema notations....
 - Relax NG – a de facto standard with both an XML-based and a “compact” (non-XML) notation
- Most practitioners prefer Relax NG, though XML Schema is the standard, and XML Schema is the foundation for other standards (like SOAP)

MANY XML DIALECTS

- There are tons and tons of standard XML dialects out there!
- **DocBook** – used by O'Reilly and Sun for their documentation/books
- **MathML** – to describe mathematical formulas
- **SVG** – vector graphics format

MIXING DIALECTS

- You may want to mix more than one dialect in a single XML document
 - E.g., a book can contain math formulas or vector images!
- XML has a namespace mechanism that allows you to do just this
 - e.g., `<svg:rect>`

PARSING XML

- One of the appealing aspects of XML is that you can read XML from almost any programming language on the planet
- There are many XML parsers available

XML PARSERS

- Some languages have their own XML parsers, which take advantage of specific language features
- Some languages are even supporting XML as a native type, like String or float
- Java is planning to support native XML, including automatic casting to Java classes

STANDARD PARSERS

- The W3C defined two different standards for XML parsers to follow
- These are independent of the programming language used, so if you learn how to use them in Java, say, you can also use them in Python

STANDARD PARSERS

- **SAX** – parser uses an event-based model, so you can “subscribe” to events like “beginTag”, “endTag”, “beginText”, etc.
- **DOM** – parser builds a complete tree corresponding to the document and returns the tree
- In practice, many DOM parsers are implemented on top of SAX parsers....

DOM CONCEPTS

- A DOM tree is a memory representation of an XML document
- It has types that mirror the XML components we described earlier
 - e.g., Element, Attribute, ...
- You can use the DOM API to navigate the tree

DOM EXAMPLE

```
var cds = xmlDoc.getElementsByTagName("CD");
var output = "";

output = "<ul>\n";
for (var i=0; i < cds.length; i++) {
    var title="";
    var kids=cds[i].childNodes;
    for (var j=0; j<kids.length; j++) {
        if (kids[j].nodeType != 1)
            continue;
        if (kids[j].nodeName == "TITLE")
            title = getTextValue (kids[j]);
    }
    output += "<li>" + title + "</li>\n";
}
output += "<\ul>\n";
var div = document.getElementById ("HeadersDiv");
div.innerHTML = output;
```

DOM EXAMPLE

```
function getTextValue(node) {  
    var kids = node.childNodes;  
    var str = "";  
  
    for (var i=0; i<kids.length; i++) {  
        if (kids[i].nodeType == 3)  
            str += kids[i].nodeValue;  
    }  
  
    return str;  
}
```

DOM OUTPUT

- The DOM also allows you to modify the internal tree structure
- In JavaScript, this is often used to alter the HTML elements viewed on the page
- It can also be used to create (or modify) an XML tree, and then generate the XML document (as opposed to using print statements)

DOM OUTPUT

```
var cds = xmlDoc.getElementsByTagName("CD");
var output = document.createElement("ul");

for (var i=0; i < cds.length; i++) {
    var title="";
    var kids=cds[i].childNodes;
    for (var j=0; j<kids.length; j++) {
        if (kids[j].nodeType != 1)
            continue;
        if (kids[j].nodeName == "TITLE")
            title = getTextValue (kids[j]);
    }
    var li = document.createElement("li");
    li.appendChild(document.createTextNode(title));
    output.appendChild(li);
}
var div = document.getElementById ("HeadersDiv");
div.appendChild(output);
```

TRANSLATING XML

- XML documents can be translated
 - from one XML dialect to another
 - e.g., from DocBook to HTML
 - from XML to plain text
 - from XML to something else, like PDF

XSLT

- XSLT is the XML standard language for translating an XML document into either another XML document or plain text
- If you need to translate into a different form (e.g., PDF), you can use XSLT to translate into “nearly” the right form, and then a simple program to do the final translation
 - e.g., translate into XMLified PDF

XSLT

- XSLT uses **stylesheets** to define the desired transformation
- XSLT stylesheets are also XML documents!
- This is a general feature of the XML ecosystem
 - Almost **everything** is an XML document

DECLARATIVE XSLT

- XML stylesheets are written in a declarative style
- The stylesheet is made up of one or more templates
- Each template matches a portion of the input document
- The rule then describes the transformation

XSLT TEMPLATES

- For example, the following template describes what to do with a <CD> element

```
<xsl:template match="CD">  
  <li>  
    <xsl:value-of select="TITLE"/>  
  </li>  
</xsl:template>
```

```
<CD>  
  <TITLE>Empire Burlesque</TITLE>  
  <ARTIST>Bob Dylan</ARTIST>  
  <COUNTRY>USA</COUNTRY>  
  <COMPANY>Columbia</COMPANY>  
  <PRICE>10.90</PRICE>  
  <YEAR>1985</YEAR>  
</CD>
```

`Empire Burlesque`

XPATH

- Did you notice the strings “CD” and “TITLE” in the stylesheet?
- These expressions select a portion of the XML document
- The syntax is called XPath
- The basics are easy, but actually XPath is a very powerful language – beyond our scope

XQUERY

- XPath can be used to select snippets of XML based on complex “path queries”
- XQuery is another XML standard
- Based on XPath, XQuery allows you to build queries and get result sets from XML “databases”
- That’s waaay beyond our scope....

XSLT AND NAMESPACES

- Recall that an XSLT stylesheet is an XML document
- It contains a mixture of XSLT elements (e.g., `xsl:template`) and target elements (e.g., HTML elements, like ``)
- In other words, XSLT uses multiple dialects, so XML namespaces are required

XSLT NAMESPACES

- You declare the XML namespaces in the root element:

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
```

- This declares that you will use the XSL/Transform namespace (defined by an URL) with the prefix **xsl:**

THE STYLE SHEET

- The final stylesheet has two templates
- The first selects the top-level CATALOG entry and produces the `<html>...</html>` skeleton
- It also creates a `...` element to hold the individual CDs
- The second template converts a CD into a ``, and we saw it earlier

THE STYLESHEET

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="CATALOG">
    <html>
      <body>
        <h1>CD Catalog</h1>
        <ul>
          <xsl:apply-templates select="CD"/>
        </ul>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="CD">
    <li>
      <xsl:value-of select="TITLE"/>
    </li>
  </xsl:template>
</xsl:stylesheet>
```

CONCLUSIONS

- XML includes several technologies that are useful in many programming tasks
- The XML ecosystem is large and varied
- We've only scratched the surface!
- To learn more, check out O'Reilly's *Learning XML* book, then go from there...