XML

Robert Grimm
New York University
The Whirlwind So Far

- **HTTP**
  - Persistent connections
  - (Style sheets)

- **Fast servers**
  - Event driven architectures

- **Clusters**
  - Availability metrics
  - Strategies for self-management, data replication, and load balancing

- **Caching**
  - Zipf-like popularity distributions
  - Effectiveness of cooperative caching
Content: XML
The Essence of XML

- External format for representing data
- Two simple properties
  - Self-describing
    - Possible to derive internal representation from external one
  - Round-tripping
    - When converting from internal to external to internal the two internal representations are equal
- Does XML have these properties? No!
  - “So, the essence of XML is this: the problem it solves is not hard, and it does not solve the problem well.”
XML
The Standards Soup

- Basic XML
  - XML 1.0
  - Namespaces in XML
  - XML Information Set
- Typing XML documents
  - DTDs (part of XML 1.0)
  - XML Schema
- Querying XML documents
  - XPath
  - XQuery
XML
Basic Ingredients

- **Elements**
  - `<foo/>, <foo></foo>, <foo> Something </foo>`

- **Attributes**
  - `<foo one="one" two="123" />`

- **Character data**
  - `<foo> Character data goes here </foo>`

- **Entity references**
  - `&lt; &amp; &gt; &quot; &apos;`
- Raw character data
  - `<![CDATA[ Some text here ]]`
- Comments
  - `<!-- This is a comment -->`
- Processing instructions
  - `<?robots index="yes" follow="no"?>`
An XML Document

- XML Declaration
  - `<?xml version="1.0" encoding="ASCII" standalone="yes"?>`

- One root element
  - All other elements must be nested, never overlap
  - All attribute values must be quoted
  - No element may have more than one attribute with a given name
  - Comments and processing instructions may not appear in tags
  - No unescaped < or & signs
Internationalization

- XML documents contain Unicode text
  - But they may still have different encodings
    - UCS-2, UTF-16, UTF-8, ISO-8859-1, Cp1252, MacRoman
    - Parsers look for #xFEFF, #xFFFE, #x3C3F786D
  - Element names may contain any letter
    - &lt;φου/&gt;
  - Character data may use character references
    - &amp;#1114; or &amp;#x45A to refer to Η
  - Elements may have an xml:lang attribute
    - &lt;foo xml:lang="el">λογος</foo>
Typing XML Documents
Take 1: DTDs

- A special syntax to define
  - Element nesting
  - Element occurrence constraints
  - Character data occurrence constraints
  - Permitted attributes
  - Attribute types and default values
  - More entities
Typing XML Documents
Take 2: XML Schema

- Why XML Schema?
  - Not a special syntax, just XML
  - More expressive
    - Precise control over element & attribute content
XML Schema
from 1,000,000 Feet

- Simple types
  - 19 of them, including booleans, integers, and strings

- Complex types
  - Atomic, list, and union types
  - Derivation by restriction
  - Derivation by extension

- Support for global and local declarations

- That’ it…
XML Schema Formalization Concepts

- Named types
- Structural types
- Validation
- Matching
- Erasure
- Relation
- Function
XML Namespaces

- Motivation
  - We want to mix different document types in the same document
    - E.g., XHTML document that also contains SVG and MathML

- The basic idea
  - Associate each element or attribute name with a namespace
  - Namespaces are identified by URIs
    - Essentially, URIs serve as opaque tokens
    - However, it is good practice to point to documentation
XML Namespaces (cont.)

- URIs are long, contain illegal characters (/,%,~)
  - Use qualified names (consisting of prefix + local part)
    - rdf:description, xlink:type, xsl:template
  - Bind prefixes to URIs
    - xmlns:rdf="http://www.w3.org/TR/REC-rdf-syntax#"
- Support default namespace
  - xmlns="http://www.w3.org/TR/REC-rdf-syntax#"
In general, writing parsers for external representations is painful.

Parsers for XML (may) reduce the tedium, check for:
- Well-formed content
  - Data adheres to XML syntax
- Valid content
  - Data adheres to some type declaration
    - Think DTD, XML Schema
Common XML Parser APIs

- **Document Object Model (DOM)**
  - Maintained by W3C
  - Tree-based
    - Exposes generic containers, allowing applications to traverse tree

- **Simple API for XML (SAX)**
  - Coordinated by David Megginson, hosted by SourceForge
  - Event-based
    - Exposes parsing events directly to application through callbacks

- **Why and when use one or the other API?**
SAX Setup

- Create a parser
  - XMLReader xr = XMLReaderFactory.createXMLReader();
- Configure parser
  - xr.setContentHandler(myContentHandler);
  - Configure features
    - http://xml.org/sax/features/namespaces
    - http://xml.org/sax/features/namespace-prefixes
- Parse XML document
  - xr.parse(new InputSource(in));
    - http://xml.apache.org/xerces2-j/samples-socket.html
SAX ContentHandler

- The methods
  - setDocumentLocator(locator)
  - startDocument(), endDocument()
  - characters(ch, start, len), ignorableWhitespace(…)
  - startElement(uri, localName, qName, atts)
    endElement(uri, localName, qName)
  - startPrefixMapping(prefix, uri)
    endPrefixMapping(prefix, uri)
  - skippedEntity(name)
  - processingInstruction(target, data)

- What’s missing from this API?
S-Expressions: A Much Simpler External Data Format

- Pair: record structure with two fields (car, cdr)
  - (1 . 2)
- List: empty, or pair whose cdr is a list
  - (), (1 2 3)
- Some basic Scheme types
  - Booleans
    - #t, #f
  - Strings
    - “This is a string”
  - Integers
    - 123
So, Why Is XML So Popular?

- Dare Obasanjo argues
  - Support for internationalization
  - Platform independence
  - Human-readable format
  - Extensibility
  - Large number of off-the-shelf tools

- What do you think?