Markup Languages
SGML, HTML, XML, XHTML

CS 431 - February 13, 2006
Carl Lagoze - Cornell University
Problem

• Richness of text
  - Elements: letters, numbers, symbols, case
  - Structure: words, sentences, paragraphs, headings, tables
  - Appearance: fonts, design, layout
  - Multimedia integration: graphics, audio, math
  - Internationalization: characters, direction (up, down, right, left), diacritics

• Its not all text
Text vs. Data

• Something for humans to read
• Something for machines to process
• There are different types of humans
• Goal in information infrastructure should be as much automation as possible
• Works vs. manifestations
• Parts vs. wholes
• Preservation: information or appearance?
Who controls the appearance of text?

• The author/creator of the document
• Rendering software (e.g. browser)
  - Mapping from markup to appearance
• The user
  - Window size
  - Fonts and size
Important special cases

- **User has special requirements**
  - Physical abilities
  - Age/education level
  - Preference/mood

- **Client has special capabilities**
  - Form factor (mobile device)
  - Network connectivity
Page Description Language

- Postscript, PDF
- Author/creator imprints rendering instructions in document
  - Where and how elements appear on the page in pixels
Markup languages

- SGML, XML
- Represent structure of text
- Must be combined with style instructions for rendering on screen, page, device
Markup and style sheets

Marked-up document

style sheet

rendering software

formatted document

document content & structure

rendering instructions
Multiple renderings from same marked-up documents

marked-up document

rendering software

document content & structure

rendering software

style sheet 1

style sheet 2

PC display

print
A short history of markup (b.w.)

- **Def.:** A method of conveying information *(metadata)* about a document
- Special characters used by proofreaders, typesetters
- **Standard Generalized Markup Language**
  - Standardized (ISO) in 1986
  - Powerful, complex markup language widely used by government and publishers
  - Also used in the exchange of technical information in manufacturing (Boeing design descriptions)
  - Functional overkill limited widespread implementation and use
HTML - Markup for the masses

• Core technology of web (along with URLs, HTTP)
• Simple fixed tag set
• Highly tolerant
  – Tag start/close
    • <p>blatz<p>scrog
    • <p>blatz</p><p>scrog</p>
  – Capitalization
• 7-bit ASCII based
• Tags express both appearance and structure
  – <title>This is structure</title>
  – What do <b>bold</b> or <i>italics</i> mean?
What is wrong with HTML?

• Fixed tag set
  - Extension has been difficult and chaotic?
    • Pages that can be rendered by IE and not other browsers
  - Prevents localization

• 7-bit ASCII
  - What about kanji, arabic, math, chemistry, etc?

• Tolerance
  - Non-specific syntax - can’t be expressed in formal manner like BNF
  - Parsing is difficult, non-deterministic. Leads to “screen scraping”

• Non-structural markup
  - Prevents clean distinction of meaning from appearance
eXtensible Markup Language

• Subset of SGML improving ease of implementation
• Meta-language that allows defining markup languages
  - No defined tags
  - Meta tools for definition of purpose specific tags
    • DTDs, Schema
• Syntax is defined using formal BNF
  - Documents can be parsed, manipulated, stored, transformed, stored in databases....
• Unicode character set
• W3C Recommendation (1998)
XML Suite

• XML syntax – “well-formedness”
• XML namespaces – global semantic partitions
• XML schema – semantic definitions, “validity”
• XSLT – language for transforming XML documents
  – One application is stylesheets
• XPATH – specifying individual information items in XML documents
• Xpointer – syntax for stating address information in a link to an XML document.
• Xlink – specifying link semantics, types and behaviors of links
Basic XML building blocks

• **One or more elements**
  - Opening tag `<tag>`
  - Empty element
    - `<picture></picture>`
    - `<picture />`
  - Non-empty element
    - Simple (CDATA) value
      - `<author>Paul Smith</author>`
    - Complex value
      - `<author><name>Smith</name><age>48</age></author>`

• **One or more attributes per element**
  - `<title lang="fr">Les Miserables</title>`
XML - sample instance document

<?xml version="1.0" encoding="UTF-8"?>
<!--This is the beginning of the XML data-->
<Book>
  <author age="48">Kevin Davies</author>
  <title>Cracking the Genome</title>
  <price>20.00</price>
</Book>
XML - well formed-ness

- Every XML document must have a declaration
- Every opening tag must have a closing tag.
- Tags can not overlap (well-nested)
- XML documents can only have 1 root element
- Attribute values must be in quotation marks (single or double) - Only one value per attribute.
XML - well formed-ness

- reserved characters should be encoded

```xml
<   &lt;
&   &amp;
]]>  ]]&amp;
>   &gt;
"   &quot;
.   &apos;
```
XML - well formed-ness

- **element names** must obey XML naming conventions:
  - start with **letter** or **underscore**
  - can contain **letters**, **numbers**, **hyphens**, **periods**, **underscores**
  - no spaces in names!
  - no leading space after `<`
  - **colon** can only be used to separate namespace of the element from the element name
  - **case-sensitive**
  - can not start with `xml`, `XML`, `xML`, ...
XML - well formed-ness

White Spaces: space, tab, line feed, carriage return

• in HTML: must explicitly write white spaces as &nsbsp; because HTML processors strip off white spaces
• not so in XML:
  • space in CDATA stays
  • tab in CDATA stays
  • multiple new line characters transformed into a single one
XML as semi-structured data

Unstructured data

Semi-structured data

Structured data

<table>
<thead>
<tr>
<th>Name</th>
<th>Occupation</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carl</td>
<td>Lagoze</td>
<td>Ithaca</td>
</tr>
<tr>
<td>George</td>
<td>Bush</td>
<td>Washington</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Location</th>
<th>State</th>
<th>Zip Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ithaca</td>
<td>NY</td>
<td>27000</td>
</tr>
<tr>
<td>Washington</td>
<td>DC</td>
<td>650000</td>
</tr>
</tbody>
</table>
XML data representation

```xml
<?xml version="1.0" encoding="UTF-8"?>
<invoice>
  <customer>
    <name>Carl Lagoze</name>
    <address>Ithaca</address>
  </customer>
  <product>
    <code>x022</code>
    <quantity>2</quantity>
  </product>
</invoice>
```
Document Object Model (DOM)

- W3C standard interface for accessing and manipulating an XML document
- Represents document as a tree with typed nodes
  - Document
  - Element
  - Attribute
  - Text
  - Comment
- DOM parser reads an XML document and builds a tree from it
DOM Interface Features

- Class structure for entities in XML documents
- Construct tree nodes of various types
  - E.g. construct element
- Create nesting structure (linkages) among nodes
  - E.g. appendChild
- Traverse trees
  - E.g. getFirstChild, getNextSibling
- Specialized sub-classes for HTML
Simple DOM Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <title lang="en">XML Basics</title>
</book>
```
DOM support in multiple languages

• Java
  - JAXP (Sun)
  - Xerces (Apache)

• Perl
  - XML::parser module
Simple API for XML (SAX)

- Event-based interface
- Does not build an internal representation in memory
- Available with most XML parsers
- Main SAX events
  - startDocument, endDocument
  - startElement, endElement
  - characters
Simple SAX Example

Document

```xml
<?xml version="1.0" encoding="UTF-8"?>
<books>
  <book>War and Peace</book>
</books>
```

Events

```
startDocument()
startElement("books")
startElement("book")
characters("War and Peace")
endElement("book")
endElement("books")
endDocument()
```
Why use SAX?

- Memory efficient
- Data structure independent (not tied to trees)
- Care only about a small part of the document
- Simplicity
- Speed
Why use DOM?

- Random access through document
- Document persistence for searches, etc.
- Read/Write
- Lexical information
  - Comments
  - Encodings
  - Attribute order
xHTML

• HTML “expressed” in XML
• Corrects defects in HTML
  - All tags closed
  - Proper nesting
  - Case sensitive (all tags lower case)
  - Strict well-formedness
• Defined by a DTD
  - Strict
  - Transitional
  - Frameset
  - `<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`
xHTML (cont.)

- All new HTML SHOULD be xHTML
- W3C validator
  - [http://validator.w3.org/](http://validator.w3.org/)
- Tidy
  - [http://sourceforge.net/projects/jtidy](http://sourceforge.net/projects/jtidy)