Markup Languages
SGML, HTML, XML, XHTML

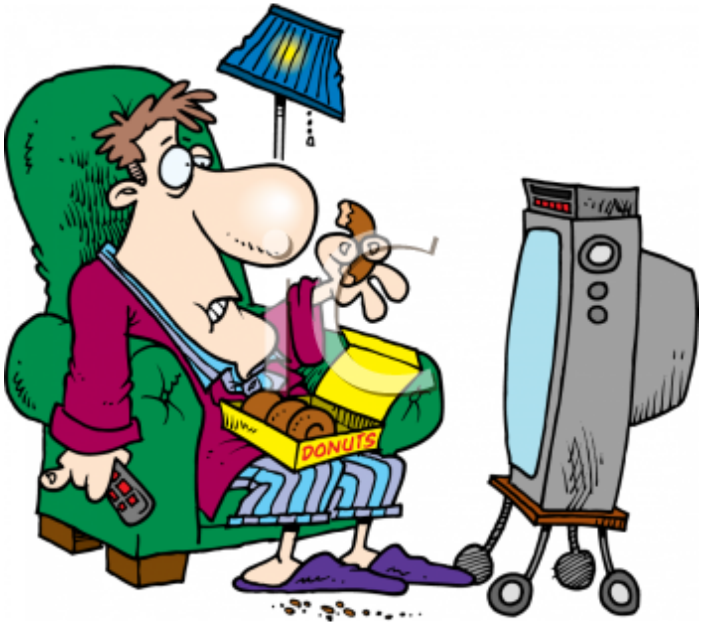
CS 431 – February 11, 2008

Carl Lagoze – Cornell University

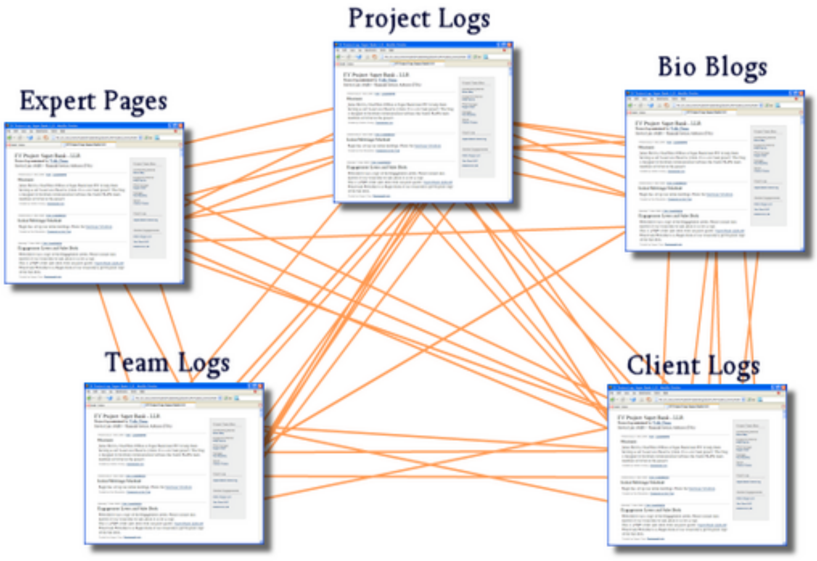
My philosophy on teaching technical tools

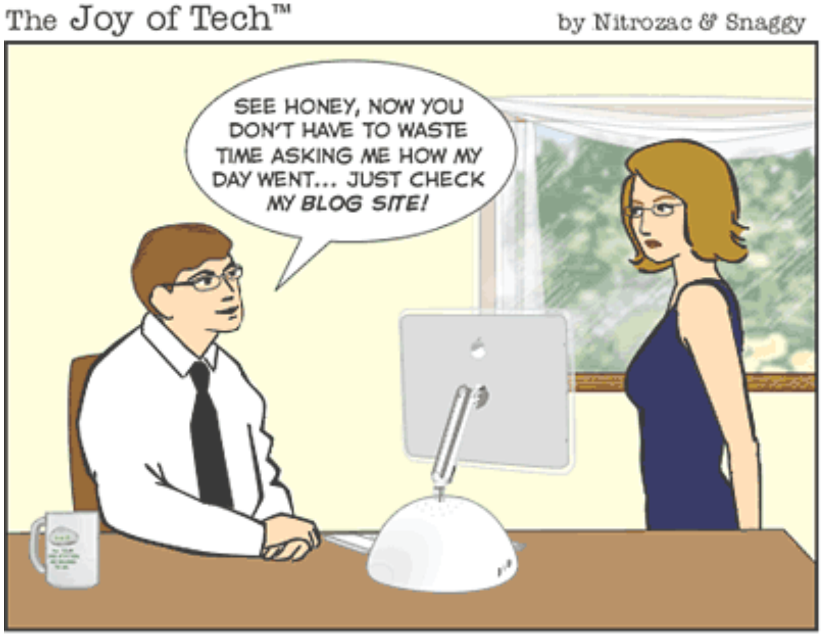
- I present the capabilities and underlying concepts
  - And give you examples
- You learn the details
- You take the time to practice

From the “document web” …



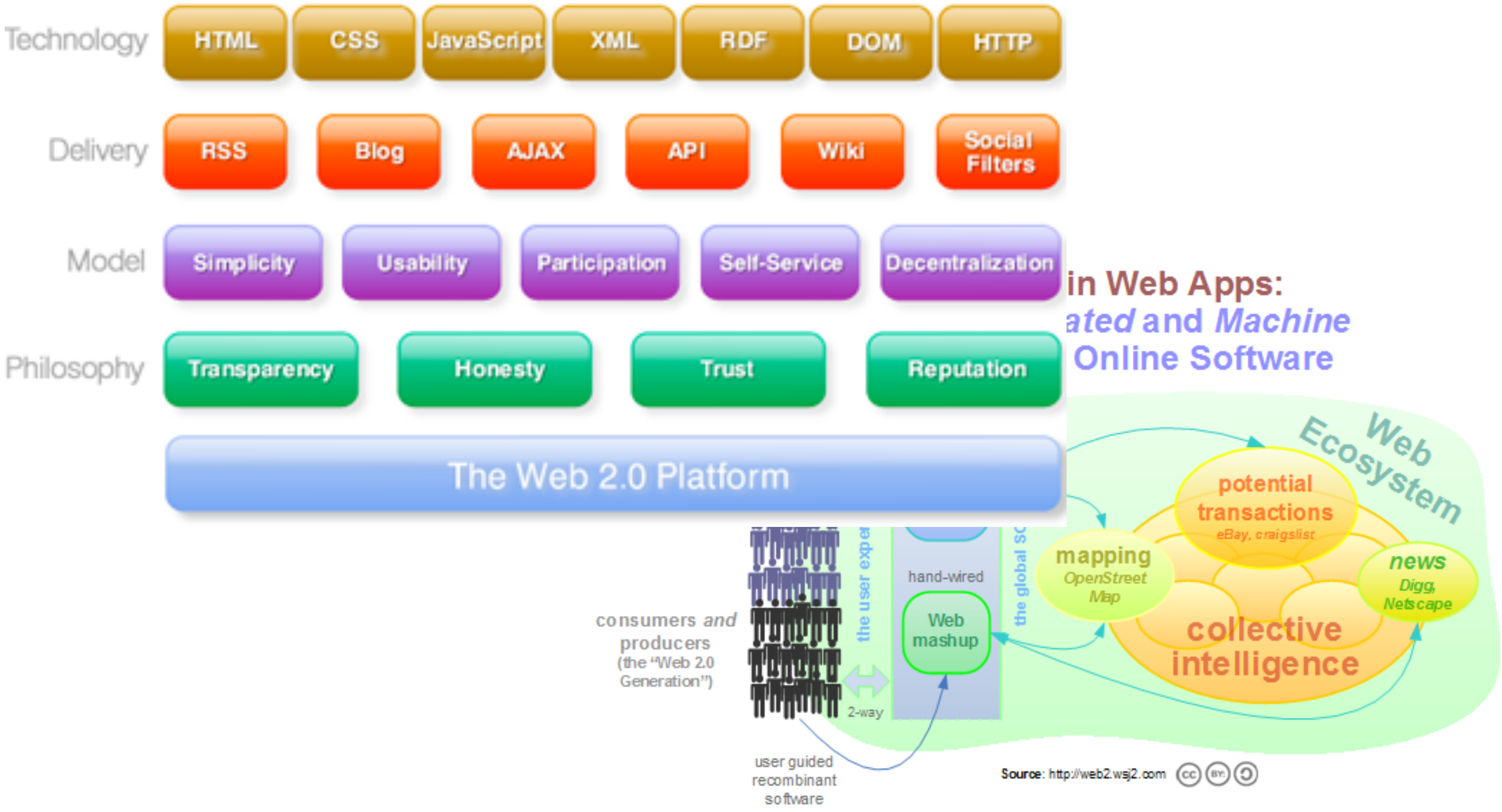
… to the “read/write” web ….




The Joy of Tech™
by Nitrozac & Snaggy
SEE HONEY, NOW YOU DON'T HAVE TO WASTE TIME ASKING ME HOW MY DAY WENT... JUST CHECK MY BLOG SITE!

It wasn't too long afterwards, that Samantha
started blogging someone else.

… to the "programmable" web



Text vs. Data

- Something for humans to read
  - User has special requirements
    - Physical abilities
    - Age/education level
    - Preference/mood
- Something for machines to process
  - Goal in information infrastructure should be as much automation as possible
  - Client has special capabilities
    - Form factor (mobile device)
    - Network connectivity
- Structure
  - E.g. Parts and wholes
  - E.g. Relationships
- Semantics
  - Global and local concepts
- Preservation: information or appearance?

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

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

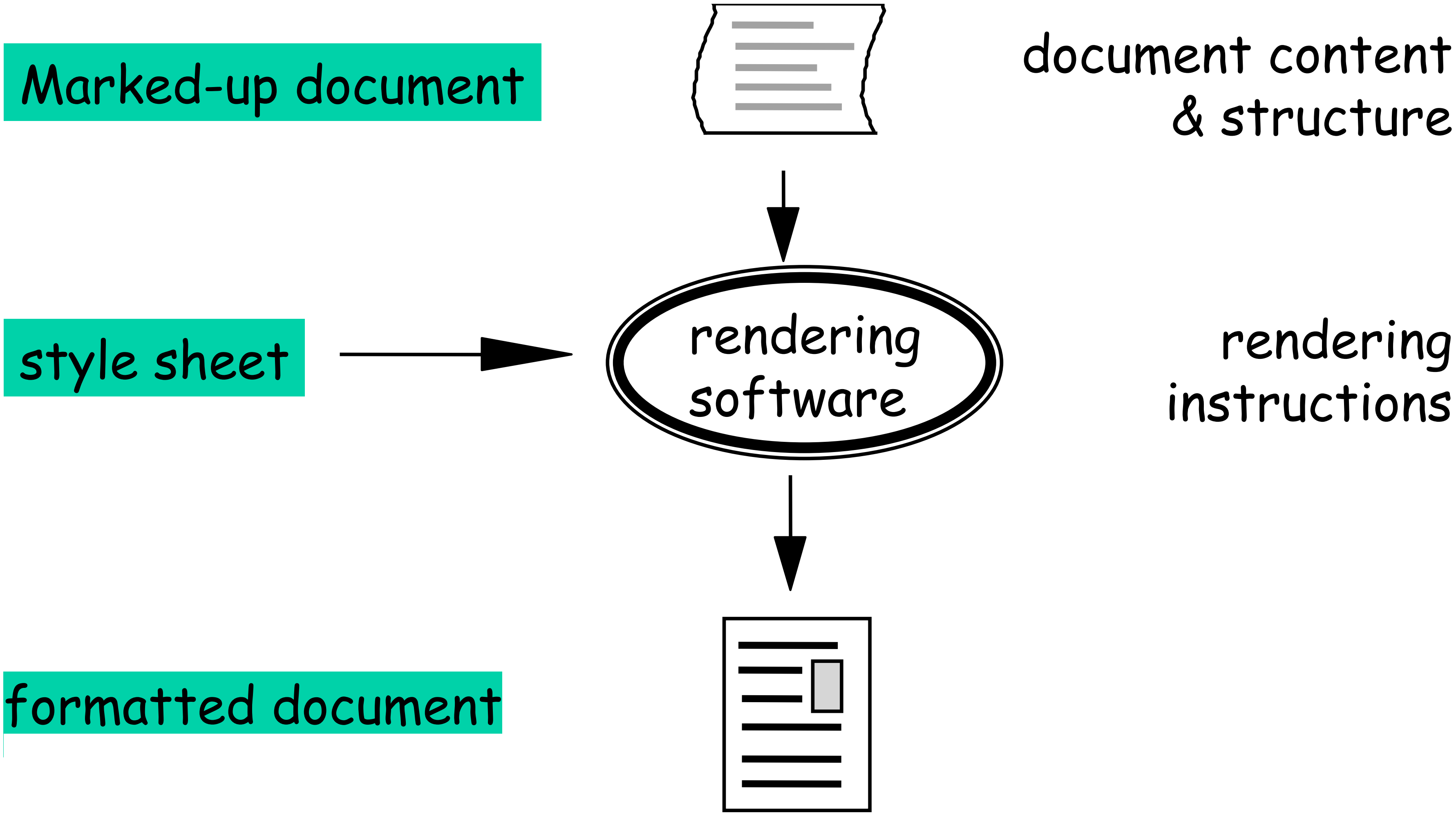
Page Description Language

- 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

document content & structure

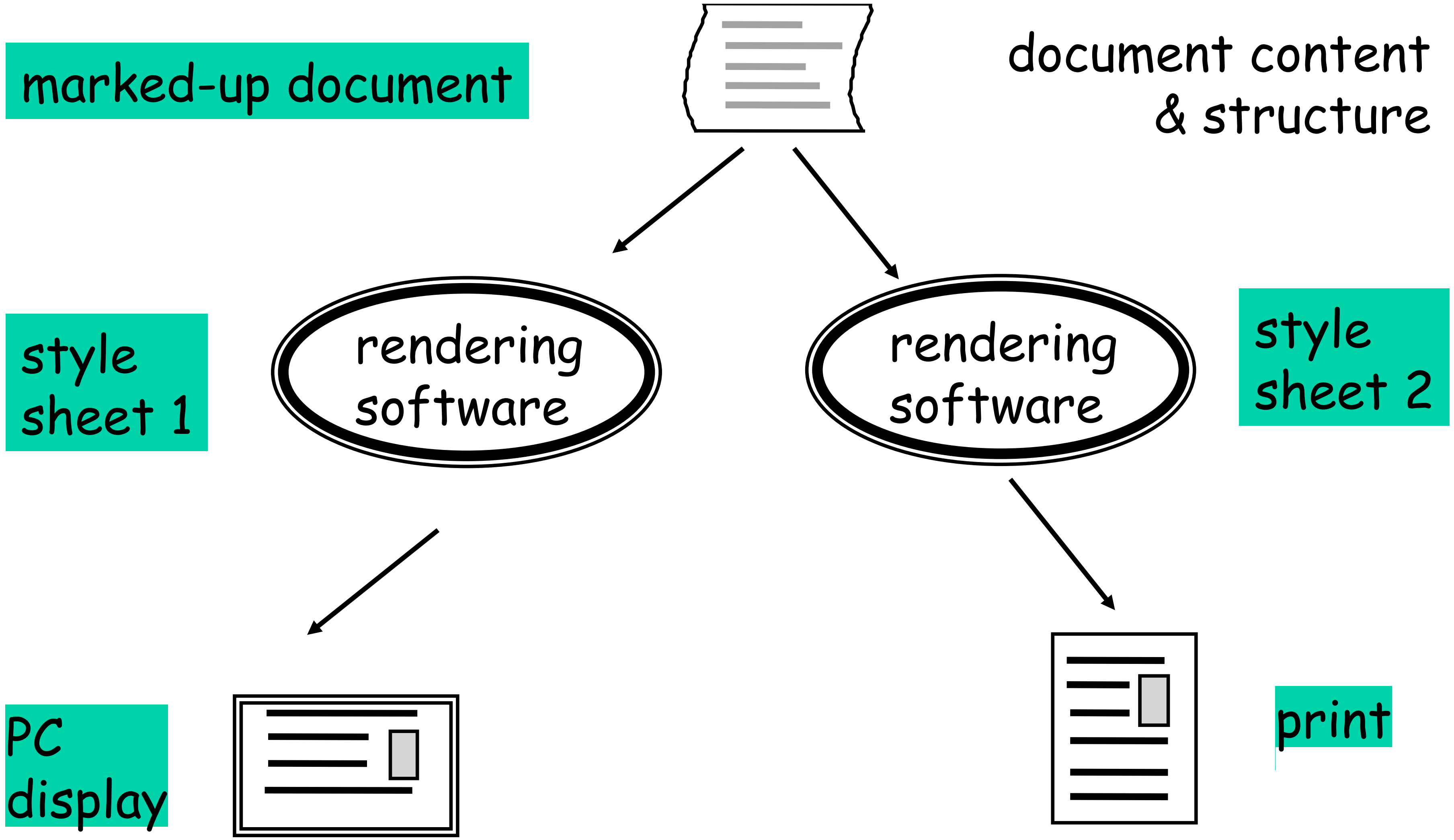
style sheet

rendering software

rendering instructions

formatted document

Multiple renderings from same marked-up documents



marked-up document

document content & structure

style sheet 1

rendering software

rendering software

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 URIs, 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?

Brief History of HTML

- HTML 1.0 – limited structural tags (title, h#...)
- HTML 2.0
  - 1997 RFC 1866
  - Basic HTML core feature set; tables, structuring/format tags
- HTML 3.2
  - January, 1997 W3C spec., attempt to restrain the browser wars
- HTML 4.0
  - 1998
  - Internationalisation
  - CSS
- XHTML 1.0
  - 2000, joint standard with HTML 4.01

Why not just use HTML

- Fixed tag set
- Domain-specific language
- Focus is on hypertext documents rather than representing semi-structured data

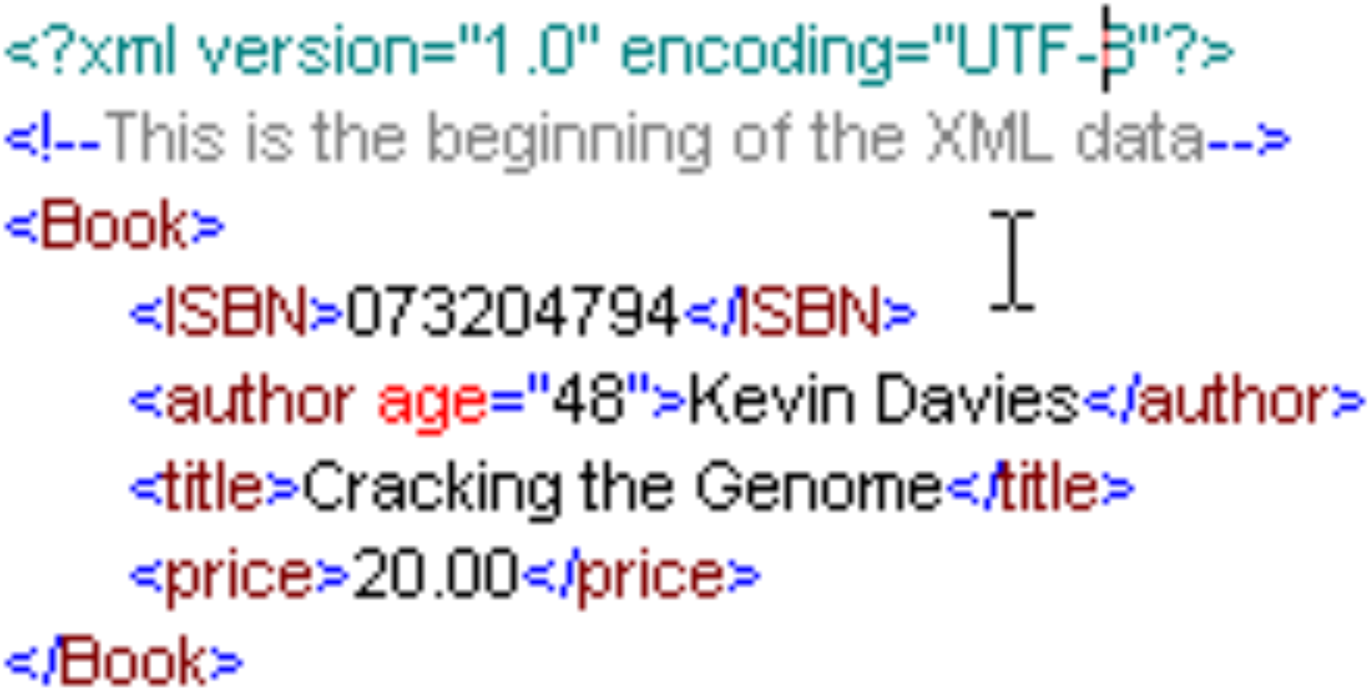
eXtensible Markup Language

- 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

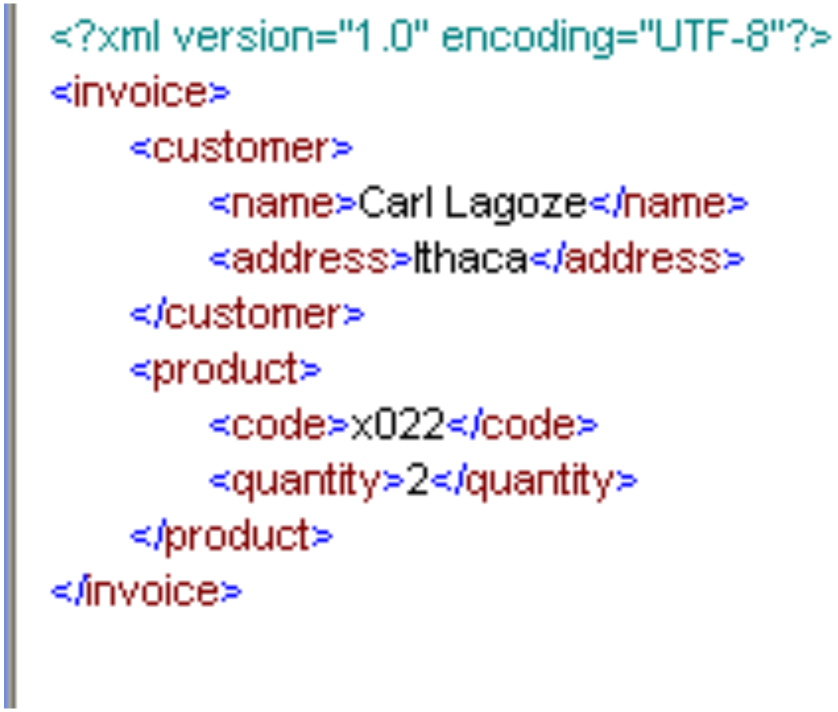
- XML syntax – “well-formedness”
- XML namespaces – global semantic partitions
- XML schema – semantic definitions, “validity”
- XSLT – language for transforming XML documents
  - One application is stylesheets
  - Distinct from CSS, which is rule-based styling language for HTML
- XPATH – specifying individual information items in XML documents
- XQUERY – generalized query language for XML-based databases
- Xpointer – syntax for stating address information in a link to an xml document.
- Xlink – specifying link semantics, types and behaviors of links

XML – sample instance document

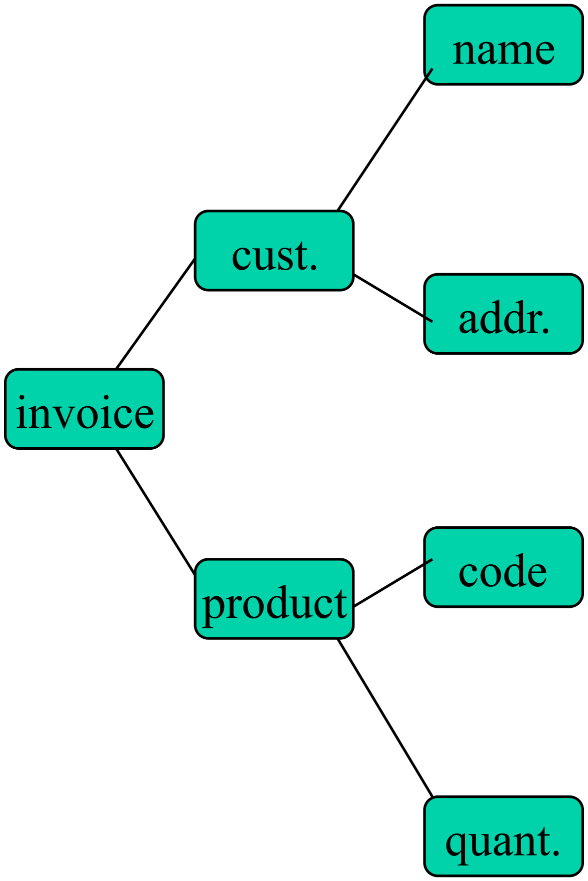


```
<?xml version="1.0" encoding="UTF-8"?>
<!--This is the beginning of the XML data-->
<Book>
    <ISBN>073204794</ISBN>
    <author age="48">Kevin Davies</author>
    <title>Cracking the Genome</title>
    <price>20.00</price>
</Book>
```

The XML tree



```
<?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>
```



XML as semi-structured data

The Networked Computer Science Technical Report Library

James R. Davis[*] Carl Lagoze[†]

Submitted to May 1996 IEEE Computer special issue on "Building Large-scale Digital Libraries".

Abstract

The Networked Computer Science Technical Report Library (NCSTRL[‡]) is a distributed digital library of research results from computer science departments and laboratories in the USA and abroad. NCSTRL benefits readers, authors, and departments. Researchers throughout the world can use familiar Internet tools (the World Wide Web) to search for, browse, read, and download technical reports from participating institutions. Authors - benefit by reaching a wider audience. Departments gain a clean, effective management system for distributing their technical reports and eliminate much of their current copying and mailing charges. This article describes the design of NCSTRL, its historical basis in earlier work, and the expected course of its development.

Keywords: protocols, document management, digital libraries

Introduction

A good library is a carefully selected collection of materials systematically organized in a manner accessible to users. Computer scientists have long desired to use the Internet as the basis of a library of computer science research. Until recently, sharing of research was usually done either by email or in anonymous FTP archives. Archives, while essential, lack many services required of a library. Over the past few years, three systems have appeared that bring us closer to having a true library of computer science research over the Internet.

The UCSTRI system[7] made FTP archives more useful by collecting indexes of their contents at a centrally searchable site. UCSTRI's weakness was that the descriptions of content on which it based its search were relatively crude and uncontrolled, usually being "README" files, which are intended for human eyes rather than automated search engines. Besides being unstructured, these files were rarely maintained with care. As a result, UCSTRI often returned imprecise results.

The next two systems, WATERS[5] and Dienst[4] improve on UCSTRI. WATERS mandates a uniform file format (an extended version of refer) for cataloging. This removes a lot of confusing

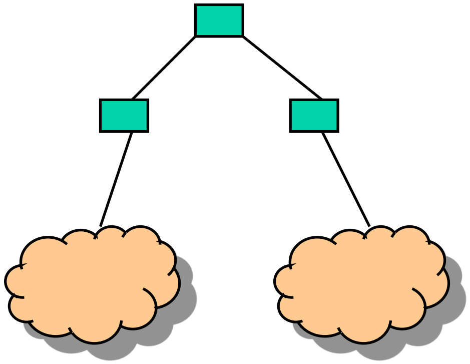
[*] Xerox Corporation, Design Research Institute, 102 Rhodes Hall, Cornell University, Ithaca NY 14853. davis@dri.cornell.edu

[†] Dept. of Computer Science, Cornell University, Ithaca NY 14853. lagoze@cs.cornell.edu

[‡] pronounced "ancestral" and hopefully connoting bountiful progeny

1

Unstructured data



Semi-structured data

Carl	Lagoze	Ithaca
George	Bush	Washington

Ithaca	NY	27000
Washington	DC	650000

Structured data

XML Tools

- http://www.oxygenxml.com/
- You can, of course, use a standard text editor (notepad, bbedit)
  - You could also wear a hair shirt.

Basic XML building blocks

- One or more elements
  - Opening tag <tag>
  - Empty element (terminal node in tree)
    - <picture></picture>
    - <picture />
  - Non-empty element
    - Simple (CDATA) value (only one child, a text node)
      - <author>Paul Smith</author>
    - Complex value (root of arbitrary sub-tree)
      - <author><name>Smith</name><age>48</age></author>
- One or more attributes per element
  - <title lang="fr">Les Miserables</title>

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

<	&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

xHTML as a special case of XML

- 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 (more on this later)
  - Defines the set of tags allowed and their nesting structure
- All new HTML (and ALL for this class) SHOULD be xHTML
- W3C validator
  - http://validator.w3.org/

Parsing and Manipulating XML
XML Parsers

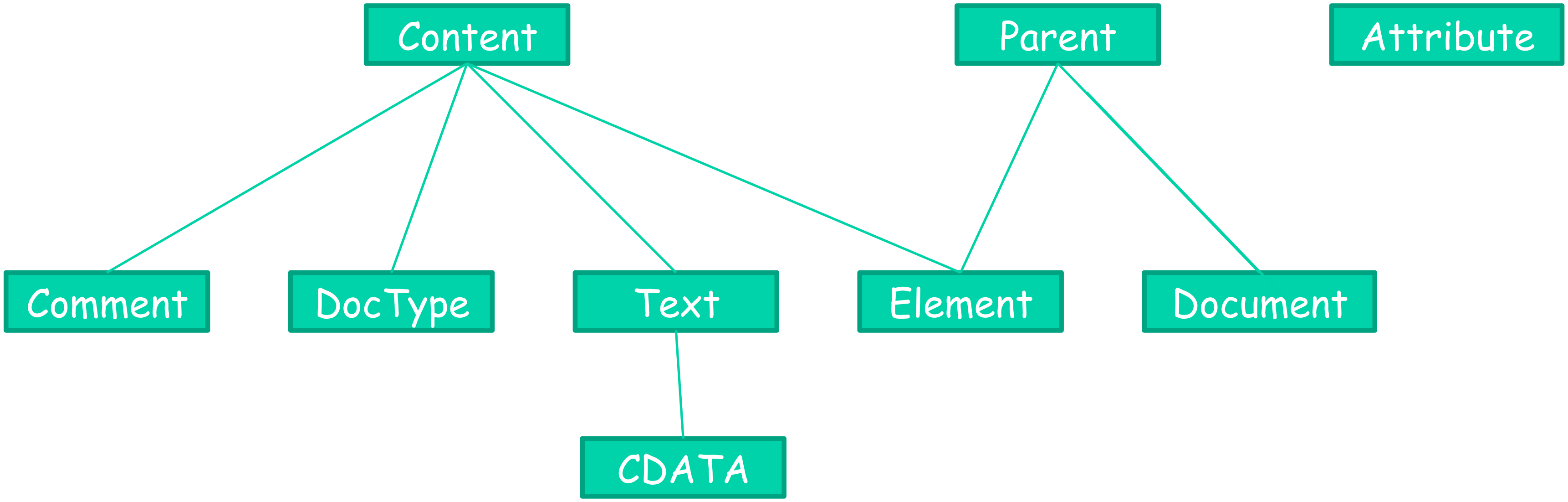
- Two types of parsers
  - Non-validating (only check well-formedness)
  - Validating
- Apache xerces is most popular for Java

Parsing & Manipulating XML
Document Object Model (DOM)

- W3C standard interface for accessing and manipulating an XML document
- Language-neutral API for manipulating/accessing XML documents
  - Bindings to multiple languages (C#, Java, Perl, Python)
  - JAXP is one Java implementation of DOM
- Basic tree model
  - General **node** interface captures general behavior
    - Child, parent, descendents, etc.
  - Specializations of node
    - Document (root)
    - Element, Text, Comment, Attribute, etc.
- Generality of DOM makes it a bit cumbersome

Parsing & Manipulating XML (JDOM) http://www.jdom.org/

- One example of a Java-specific XML tree API
  - (Another is dom4j - http://www.dom4j.org/)
  - 80-20 rule, common operations easy to perform, use DOM or dom4j for more complex.
- Tailored for Java rather than language neutral
  - Java elements described as a class hierarchy
  - Collections of elements and attributes represented as Java lists, traversed using Iterators



Parsing & Manipulating XML (JDOM)
http://www.jdom.org/

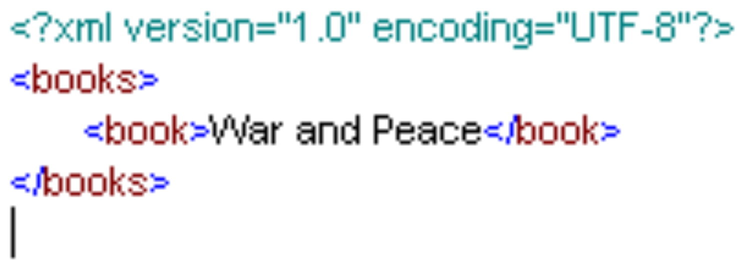
- Basic navigation functionality
  - Parent
  - Child (all, specific, filtered)
  - Descendents
  - Attributes (all, specific, filtered)
- Basic tree manipulation
  - Adding, replacing, removing contents and attributes)
  - Text modification
  - Maintains well-formedness

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 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 or JDOM?

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