Structured Documents: An Introduction

Michael B. Spring
Department of Information Science and Telecommunications
University of Pittsburgh
spring@imap.pitt.edu
http://www.sis.pitt.edu/~spring

Overview

- History of Structured Document
  - Early systems
  - The relationships between SGML, HTML, and XML
- Documents
  - Content
  - Tagged Content
  - Valid Tagged Content
- Document Type Descriptions
  - Elements
  - Attributes
  - Entities
- Simple Examples

History

- Primitive systems embedded procedural commands
  - Pub, runoff
  - Nroff, Troff
  - Script, Tex
  - GML, XICS (These systems crossed over)
- Scribe takes a new structured approach
  - Describe the document in terms of components
  - Separate content from structure from layout
- Commercial versions of GML, XICS, and the public domain Latex (based on Tex), turned to a structured approach

Procedural Markup(Peachtext)

\cpi12,propon,lm5,lw80,tm6
\bm6,bf3,cnp3,pi6,sp1,justc
\ctr\@Faculty Development Presentation
\ctr\January 26, 1984
~Introduction:I will cover three topics:

First, the reasons why we should be thinking about tv
Second, some of the things to keep in mind in working with video
Third, some ways to get started

~Reasons: We should be looking at video because:

The influence of Walter Annenberg and Mobil Oil
The emergence of TAGER and PECS
The growth of cable -- implications of over channeling
The increase in satellites -- implications of abundance
Microcomputer controlled videodiscs -- a marraige made in heaven

\np
~How to get started

Structural Markup(Scribe)

```
@make(report)
@begin(titlepage)
@title[COMPUTER CENTER REPORT]
@date[January 12, 1984]
@end(titlepage)
@chapter(DEPARTMENTAL LIBRARIES)
The library for Computer Science, CSL:, has been created, with a quota of 10,000 blocks.
Free space on SPL: was critical during the Fall term. It is currently at 106,000 for System
A and 122,000 for System B, and will decrease rapidly as the Winter term progresses.
@section(INFORMAL COURSES)
The schedule of informal courses for the Winter term has been announced.The courses being offered are
@begin(list)
Computing for the New User
Introduction to Graphics at Pitt
Interactive System 1022
@end(list)
Please see SYS:NEWS for details.
```

SGML and ODA

- SGML and ODA were developed and competed as structured document interchange standards.
- Both took an approach that separated content from layout and structure
- While ODA was the more complete standard, SGML, with publisher support emerged as a weak winner.
- When Berners-Lee was developing the WWW, he developed a generalized SGML DTD (Document Type Definition) calling it the HyperText Markup Language, or HTML.

The Truth about HTML

- HTML is much less than meets the eye.
- Berners-Lee chose to develop a document description based on the rules set out in SGML.
- SGML defines the rules by which a class of documents is defined.
  - Each DTD(Document Type Definition) has the potential of being the basis for 1000s of different documents.
  - SGML allows a virtually infinite number of DTD to be defined.
  - HTML is one single DTD that has been used to describe millions of different documents.

SGML and XML

- Unlike HTML, XML is a meta language like SGML that defines the rules for defining classes of documents.
- XML is both simplified SGML and extensions to SGML
  - The first draft of XML eliminated the computational expensive features of SGML and some features that were no longer needed.
  - XML, with time, added new features that allowed for better control of data types and more consistent processing of document descriptions

A Couple Definitions

- Content

  “Understand the forces moving e-business forward. Appreciate the impact of bit businesses versus atom business, national versus global markets.”

- Tags <xyz> = starttag </xyz> = endtag

  <LIST><ITEM></ITEM><ITEM></ITEM></LIST>

- Elements (tags + content)

  <LIST><ITEM>Understand the pressures that are moving e-business forward as a new mode of doing business</ITEM><ITEM>Appreciate the impact of bit businesses versus atom business, national versus global markets, and customer driven manufacturing on the conduct of business</ITEM></LIST>

More Formally

- An element in SGML (and XML) consists of:
  - A starttag
  - Content
  - An endtag
- a starttag is a name between angle brackets
  - It may include definition of one or more attributes
- an endtag is an element name between </ and >
- an empty element may be specified as an element name between < and />

A Document

Course on E-Business

by: Michael B. Spring, University of Pittsburgh

General Description

The Internet is providing new ways of communicating and of doing business. There are many facets to the developments and many technologies appearing and disappearing in the rush to develop this new area.

The objectives of this course will be to:

Understand the pressures that are moving e-business forward as a new mode of doing business

Appreciate the impact of bit businesses versus atom business, national versus global markets, and customer driven manufacturing on the conduct of business

A Document with Tags

<MYDOC><INTRO>

<TITLE>Course on E-Business**</TITLE>**

<AUTHOR>Michael B. Spring**</AUTHOR>**

<INST>University of Pittsburgh**</INST></INTRO>**

<CHAP><CTITLE>General Description**</CTITLE>**

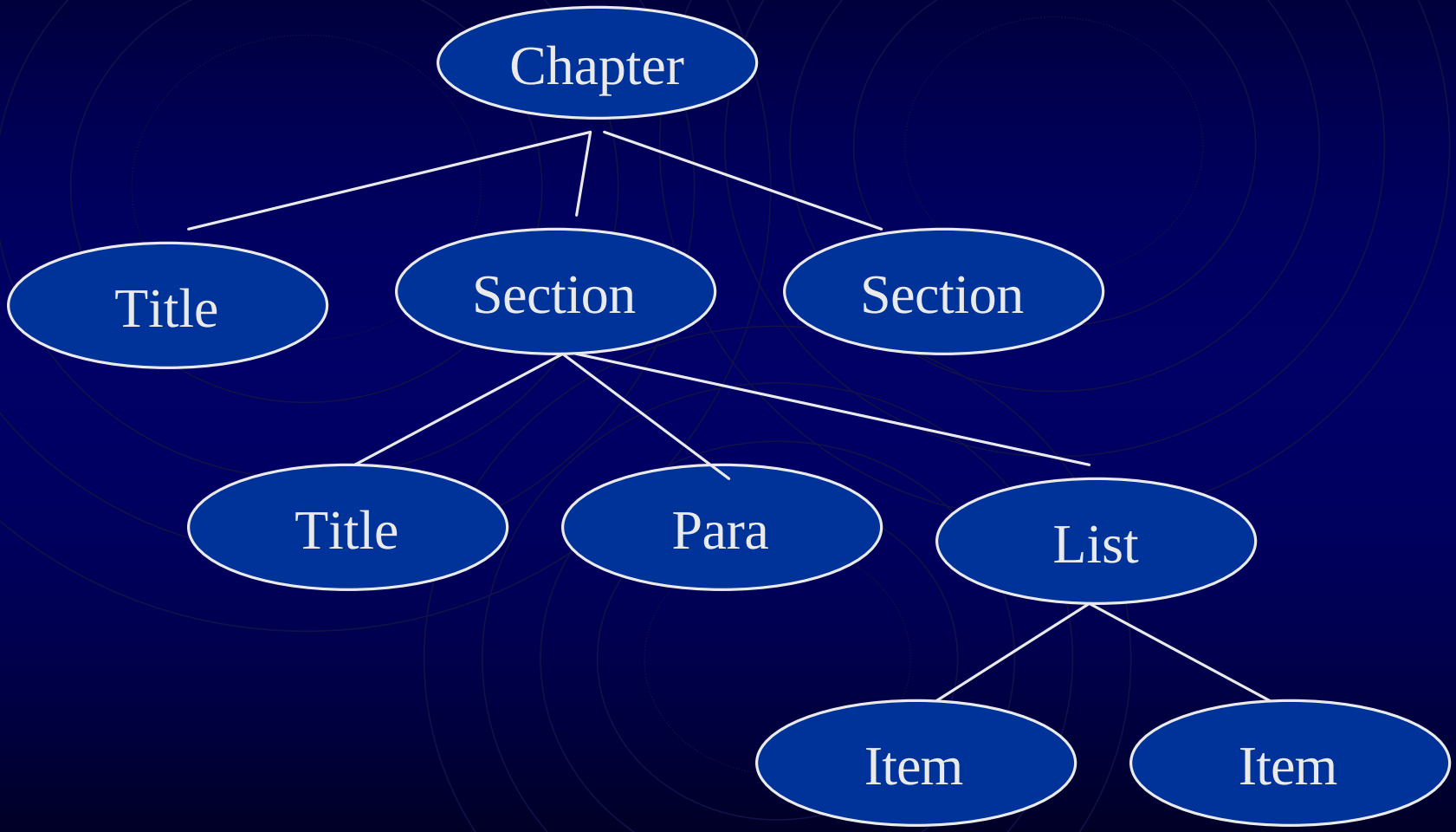
<PARA>The Internet is providing new ways of communicating and of doing business. There are many facets to the developments and many technologies appearing and disappearing in the rush to develop this new area**.</PARA>**

<PARA>The objectives of this course will be to**:</\PARA>**

<LIST><ITEM>Understand the pressures that are moving e-business forward as a new mode of doing business**</ITEM>**

<ITEM>Appreciate the impact of bit businesses versus atom business, national versus global markets, and customer driven manufacturing on the conduct of business**</ITEM></LiST></CHAP></MYDOC>**

A Document Visually



A Note about SGML

- SGML is being pushed in the background
- SGML was (is still being) used in corporate settings
  - SGML editors and tools were built and used to manage large document projects
  - SGML folks saw XML as a simple a display language
  - SGML documents were to be converted to XML for display
- SGML was dependent on two companion standards:
  - The Document Style Semantics and Specification Language (DSSSL) for presentation
  - The HyTime Language was developed to provide new forms of linking (HyTime was originally for multimedia synchronization)
- The development of companion standards for XML has been explosive

Well-formed and Valid

- SGML (and XML) documents include tags or copymarks that define elements
- Documents with tags that are correctly nested and written are called “well-formed”
  - The elements of a document must be nested – elements cannot overlap – and there are strict rules about naming of elements
- Documents whose elements are as specified in a DTD are called “valid”
  - Document Type Definitions (DTDs) specify the permissible elements in a document, the order of occurrence, and whether they can be repeated

Markup

- SGML and XML documents begin with a declaration
  - The SGML declaration was a complex structure allowing more than 20 processing instructions to be set.
    `<!SGML “ISO 8879:1986” …>`
  - An XML document begins with the declaration which is actually a processing instruction
    `<?xml version="1.0"?>`
- SGML and XML documents then specify the DTD or schema which it follows

  `<!DOCTYPE name [.....]>`
  - To be well formed, an XML document need not have a DTD

The DTD

- A DTD can be:
  - PRIVATE – specified in the document
  - SYSTEM – specified on the system
  - PUBLIC – specified in some public registry
- This presentation deals only with private
- The name you give to a document type must be the same as the root element
  - the DTD is specified via the doctype element
  - <!DOCTYPE name [.....]>

Content Modeling

- The DTD defines a model of the document content
- Within the [] of the <!DOCTYPE > declaration, the designer specifies the content of the document in terms of:
  - Elements
  - Attributes
  - Entities
  - Data types
  - Notations
- The most important are the element definitions

What the DTD does

- The document type definition defines the legitimate markup structure for a document.
- For each element, the DTD specifies
  - element content, if any
  - attributes of element
  - the allowable sub-elements including
    - ordering information
    - occurrence information

The DTD components

- <!ELEMENT – defines the content model for a given element
- <!ATTLIST – defines the attributes for a specified element, possible values, and defaults
- <!ENTITY – defines the entities that can be referred to in the document using entity references.
- <!NOTATION – defines, like entities means for handling non-SGML notations

Element Declaration

- groups
  - () parentheses define a group
- sequence connectors
  - , indicates in the specified order
  - | indicated a choice
  - & (SGML only)indicates elements may be in any order
- occurrence indicators
  - nothing indicates a single instance is required
  - ? Indicates optional
  - * indicates optional and repeatable
  - + indicates required and repeatable

More Detail

- Element names must consist of at least one letter
  - if more than two characters long, they may start with a _ or a :
  - letters, digits, hyphens, period, and underscores are allowed in the body of the name
  - spaces and tabs are not allowed

Attribute Defintion

- The second type of type declaration is attribute definition, it takes the general form
  - <!ATTLIST gi name value/range default>
  - <!ATTLIST memo status ("dft"| "fnl") "fnl">
- Given this in the DTD, in text we could see a value after = and in """"
  - <memo status = "draft">
- value range must either be a group, or a reserved word (see next slide)

Attribute Reserved Words

- The reserved words can be:
  - CDATA -- character data
  - NUMBER -- a number
  - NAME -- a name string
  - NMTOKENS -- names that can begin with a number
  - NUTOKENS -- names that begin with a number
  - ID -- must be a valid and unique name within the scope of the document; ID attributes should be named consistently -- some would say they should be called id
  - IDREF -- need not be unique but must match a value of an ID in the document.

Default Values

- Default values may be specified as one member of the set.
- They may also include the following:
  - #REQUIRED -- must be supplied
  - #IMPLIED -- is optional and will be supplied by the system if absent
  - #CURRENT -- is the most recent value
- This allows definitions like
  - `<!ATTLIST fig figtag ID #IMPLIED>`
  - and
  - `<!ATTLIST figref reffig IDREF #IMPLIED>`

Entity Definition

- <!ENTITY SIS "School of Information Sciences">
  - allows &SIS; in the text.
- Character references are like entity definitions
  - an entity reference for a character might be &
  - a character reference might be   or

Special Entity References

- if an entity is a processing instruction, the keyword PI is inserted in the definition between the entity and the string literal. for example
  - <!ENTITY dothis PI "newpage recto">
- if an entity is to be allowed in a parameter literal, the entity must be defined using a % indicator. for example
  - <!ENTITY % myref "some string">
  - this allows us to resolve <!ENTITY another "%myref; some other string">

Content

- SGML character data may be defined as CDATA or RCDATA or PCDATA.
  - CDATA is simply that -- Character data
  - RCDATA is character data that may contain entity references
  - PCDATA is character data that may be fully parsed

Partial DTD

```
<!DOCTYPE letter [
<!ELEMENT letter (adrs, sal, body, sig)>
<!ELEMENT adrs (name, str, city)>
<!ELEMENT name (first, last)>
<!ELEMENT body (p*)>
<!ELEMENT p (#PCDATA)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT sal (#PCDATA)>
<!ELEMENT sig (#PCDATA)>
<!ELEMENT str (#PCDATA)>
<!ELEMENT city (#PCDATA)>
]>
```

Partial Document

```
<letter>
   <adrs>
     <name>
                <first>Joe P.</first>
                <last>Smith</last>
     </name>
     <street>1 main street</street>
     <city>Pittsburgh PA 15213<city>
   </adrs>
   <sal>Dear Pat</sal>
   <body>
     <p>some text</p>
     <p>some more text</p>
     <p>yet more text</p>
   </body>
   <sig>Mike</sig>
</letter>
```