Understanding XML

Aruna Panangipally

tasc CONSULTING

Technical Writing | Courseware | Training

Session Outline

- Why should a technical writer know XML?
- The Beginning
  - Understanding markup languages
  - Origins of XML
- The Semantics
  - Tags and elements
  - Element content
  - Attributes
  - Comments
  - Well-formed XML documents

Session Outline...

- Managing presentation
  - CSS
  - XSL-FO
- Enforcing document structure
  - DTD
  - Schema
- Single sourcing with XML
  - XML
  - XPath
  - XSLT

Why should a TW know XML?

- XML is emerging as THE information standard.
  - Many commercial products save information in the XML format.
  - Example: Microsoft Office
- Single-sourcing is the way to go for technical writers.
- XML enables single sourcing.

Origin of Markup Languages

- Flat files
- Databases
- Markup languages

Understanding Markup Languages

- Markup conveys information about the content it is applied to.
- For markup to be commonly understood, it is necessary to define rules that declare:
  - Identifiers to be used as markup.
  - Role/action of each markup identifier.
- A collection of such rules becomes a **markup language**.

Early Markup Languages

- Generalized Markup Language (GML)
  - Was unveiled in 1969
  - Was the brainchild of three IBM researchers: Charles Goldfarb, Edward Mosher and Raymond Lorie.
  - Was designed such that various types of documents could be created from a single source.
  - Enabled text editing, formatting, and information retrieval subsystems to share documents.
  - **Did not allow for defining document structure which became its biggest drawback.**

Early Markup Languages

- Standardized Generalized Markup Language (SGML)
  - Created by the ANSI committee on Computer Languages for the Processing of Text whose goal was to define a portable document format.
  - Adopted as a standard by ISO in 1986.
  - Is a specification that defines how to create markup. The authors are, in effect, creating individual markup languages that best suits their requirements.
  - **Is very complex and requires fairly high bandwidth usage.**

HTML - The Popular SGML Application

- Hypertext Markup Language (HTML)
  - Defined by Tim Berners-Lee.
  - Defined a set of SGML tags that could be used to format text and provide links from one document to another.
  - Became the de facto presentation language for the web.

Origins of XML

- People wanted to share data across the Internet.
- Existing markup languages were not suitable.
  - HTML focused on presentation and has a limited tag set.
  - SGML was too complex to implement and required fairly high bandwidth.

The Extensible Markup Language (XML)

- Created by a committee setup in 1996 to create a new language that:
  - Incorporates the SGML features of extensibility, structure and validity
  - Could be used over the Internet.
- Is a subset of SGML optimized for delivering information over the web.
- Enables authors to define descriptive tags that best meet the requirements of their document(s).
- Enables authors to define a structure for the document and validate the document against the structure.

The XML Family

- XPath
- XSL
  - XSLT
  - XSL-FO
- XLink
- XPointer
- More coming….

The Semantics

- Naming Conventions
- Tag
- Element
- Attributes
- Comments
- Processing Instructions

XML Naming Conventions

- Valid names begin either with a letter or an underscore followed by any combination of letters, numbers, periods, underscores, or hyphens.
- Names cannot contain spaces.
- Names cannot begin with any form of the reserved word XML.
  - Names starting with **XML**, **xml**, **Xml** or any other variation are not legal.
- Names using the colon (:) character are reserved for XML namespaces.

Spot the Invalid Names

- Which of the following are invalid XML names and why?
  - `BookName`
  - `Book Name`
  - `-BookName`

Tag

- Every XML document is made up of one or more tags that describe the data.
- The tags in XML always come in pairs, that is, there is a start tag and an end tag.
- The format of a tag pair is:

```
<TagName> EnclosedData</TagName>
```

- An empty element can be represented as:

```
<TagName/>
```

Element

- The start tag, end tag and the data enclosed between these tags constitute an **element**.

```
<Qualification>Graduate</Qualification>
```

- The data between the start tag and end tag is called the **element content**.

Types of Element Content

- Simple
- Element
- Mixed
- Empty

Types of Element Content

```
<Strings>
<Concatenate>
   <FirstString>Hello</FirstString>
   <SecondString>World</SecondString>
   <MoreStrings>
        Other Strings will be included here.
        <OtherStrings></OtherStrings>
   </MoreStrings>
   <AdditionalInfo></AdditionalInfo>
</Concatenate>
</Strings>
```

Wellformed Elements

- Tag names must follow the XML naming conventions.
- No space is allowed between the leading < and tag name.
  - Space is allowed between the tag name and the closing >.
- Tag names are case sensitive.
- Every starting tag must have a closing tag.
  - The empty element is an exception.

Wellformed Elements

- Elements cannot overlap.
- Every XML document must have a unique root element.
- Adopt the empty element syntax: `<TagName/>`.

Spot the Mistakes

```
<Organization>
  <Name>TASC Consulting</Name>
  <Board>
     < Director>Aruna Panangipally</Director>
     <Director >Suvarna Pandit</Director>
  </board>
  <Location>Mumbai, India</ Location>
</Organization>
```

Attributes

- Attributes are attached to an element and provide additional information about the element or its content.

```
<TagName Attribute_1="value“
    Attribute_2="value“
    ……..
    Attribute_N="value">

    ElementContent

</TagName>
```

Attributes

```
<books>
   <type>Fiction</type>
   <title>
       Money Changers
   </title>
   <author>
       Arthur Hailey
   </author>
   <inStock>Y</inStock>
</books>
```

```
<books type="Fiction"
   inStock="Y">
   <title>
       Money Changers
   </title>
   <author>
       Arthur Hailey
   </author>
</books>
```

Attributes

- Attribute names must follow the XML naming conventions.
- An element can have as many attributes as required. However, a given attribute can be used only once with a tag.
- Attributes are added as a part of the element's start tag but not to the end tag.
- Attribute values can be enclosed in single or double quotes.

Attributes Vs. Elements

- Attributes take away the structuring information.
- An element can use multiple instances of a child element but only one instance of a given attribute.
- An attribute does not easily lend itself to enhancements.
  - We can add additional child elements and/or attributes to an element but encompassing new information in an attribute is difficult.
- Ideally, elements should store the actual data and attributes should store the information that describes elements.

Change Element(s) to Attributes

```
<mail_message>
   <priority>High</priority>
   <to>Chris Harris</to>
   <from>Anna Williams</from>
   <date>22/07/2000</date>
   <subject>Regarding the meeting </subject>
   <attachment>None</attachment>
   <message>
        The meeting is rescheduled for March 20, 2004.
   </message>
</mail_message>
```

Comments

- Comments:
  - Enable the inclusion of descriptive information about the document and its contents in an XML document.
  - Is not a part of the data in the XML document
  - Is targeted at the authors/editors of an XML document.

Comments

- Comments
  - Start with <!–
  - End with -->
- The format of a comment is:

```
<!–Text_of_the_Comment-->
```

- Comments cannot:
  - Be placed within a tag.
  - Use the double hyphen (--) within comment text.
  - Appear before the XML declaration.

Wellformed XML Documents

- A document that conforms to the XML rules for syntax and structure is said to be **wellformed**.
  - **The minimum criteria for a document to be considered as an XML document is that it be wellformed.**
- An XML document has two parts:
  - **Document Prolog** which contains the following:
    - The **mandatory** XML declaration.
    - The **optional** document type declaration (DTD) to be used.
    - **Optional** comments.
    - **Optional** processing instructions.
  - Document Body which contains the document content and:
    - Follows the prolog.
    - Starts with the root element.

XML Declaration

- Appears at the start of every wellformed XML document.
- Uses the format:

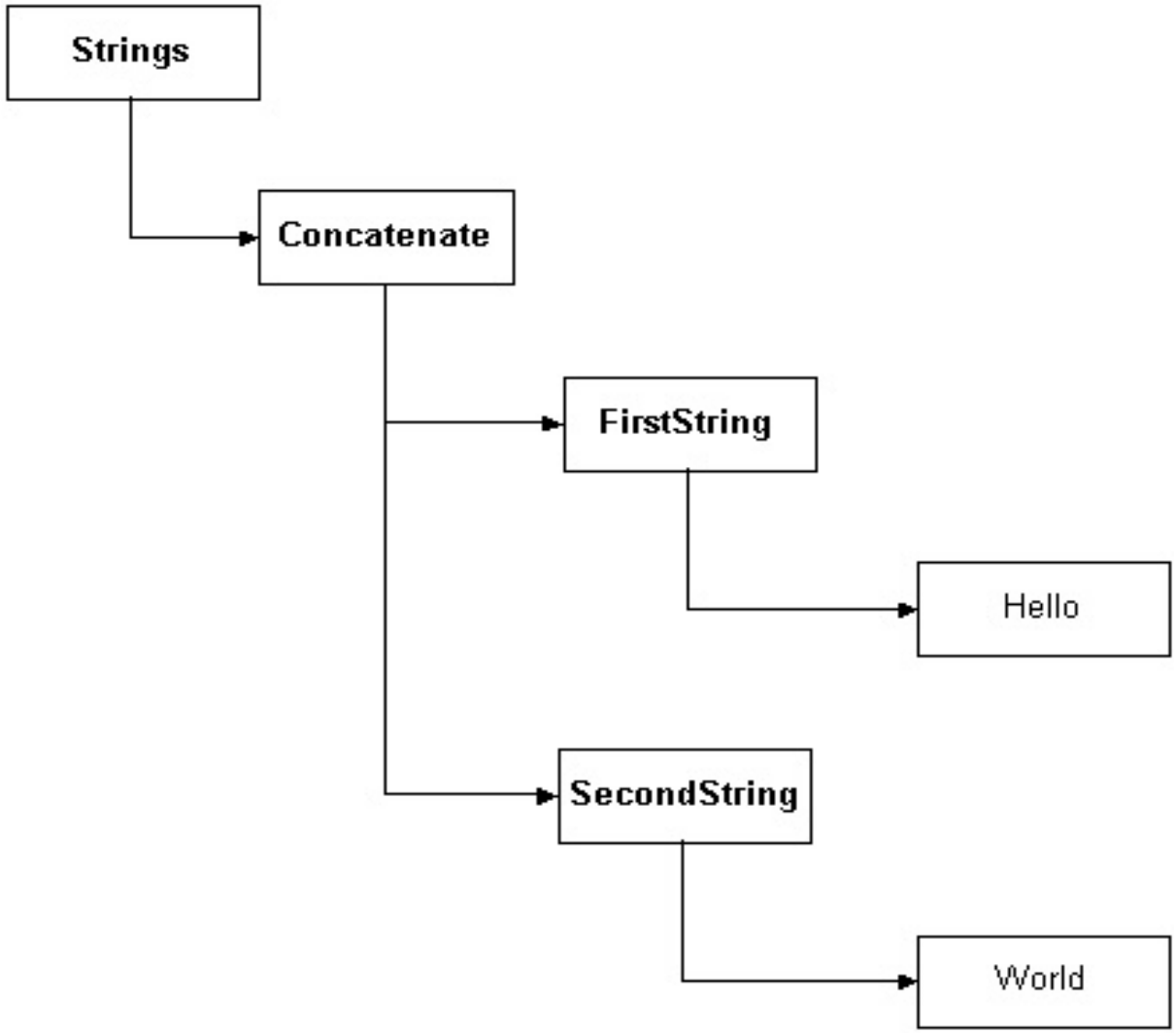
```
<?xml version="value" encoding="value" standalone=
"value"?>
```

- The attribute `version` is mandatory while `encoding` and `standalone` are optional.

Your First XML Document

```
<?xml version="1.0"?>
   <mail_message>
        <priority>High</priority>
        <to>Chris Harris</to>
        <from>Anna Williams</from>
        <date>22/07/2000</date>
        <subject>Regarding the meeting </subject>
        <attachment/>
        <message>
                We will have to postpone the meeting until next week
        </message>
        <signature>Anna
                <email_id>anna@xyz.com</email_id>
        </signature>
   </mail_message>
```

The Tree Structure

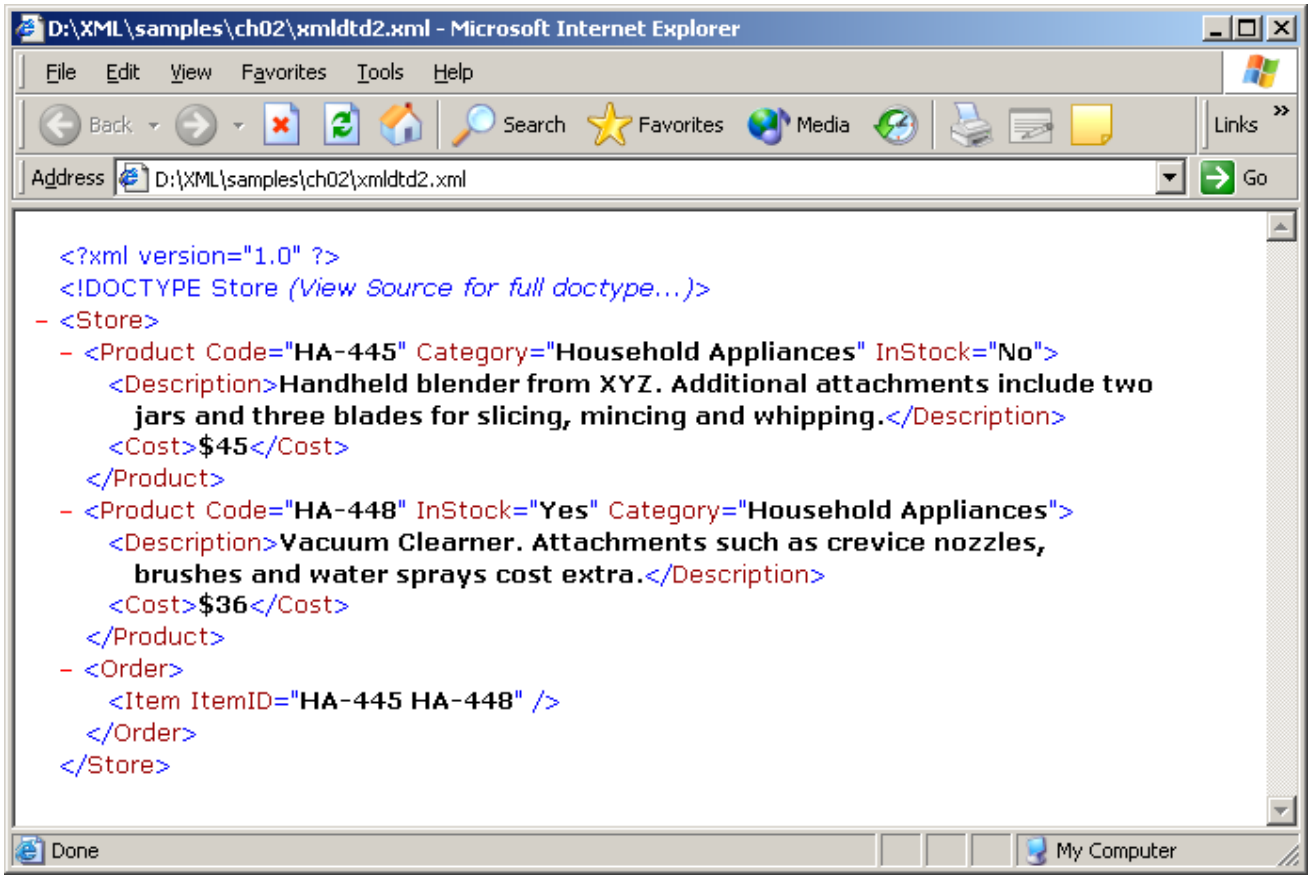


Create an XML Document

One World University					
Degree ="Literature"			**Degree** = "Physics"		
ID	**Name**		**ID**	**Name**	
	LastName	**FirstName**		**LastName**	**FirstName**
101	Smith	Jack	201	Monroe	Stanley
102	Thomas	Mary	202	Hall	Linda
103	Nick	Anna	203	Simpson	Beth

Managing Presentation

- An XML document is just structured data.



Managing Presentation

- The need of the hour is a mechanism that presents the data in an XML document in a user-friendly format.
- A style sheet is a collection of styles.
  - Each style defines how the data associated with a particular tag is displayed.
  - Even in HTML, only the tags are pre-defined. The look-n-feel of the tags are defined by the browsers that support HTML. As a result, the look of the HTML document varies with the browser it is viewed in.
- The two technologies that can be used to handle presentation of XML documents:
  - Cascading Style Sheets (CSS)
  - The XSL Formatting Objects (XSL-FO)

Using CSS to Manage Presentation

- A style sheet rule is defined as:

```
selector
{property_1:value;
property_2:value; … … …;
property_n:value}
```

- For example:

p{color:blue; font-family:arial}

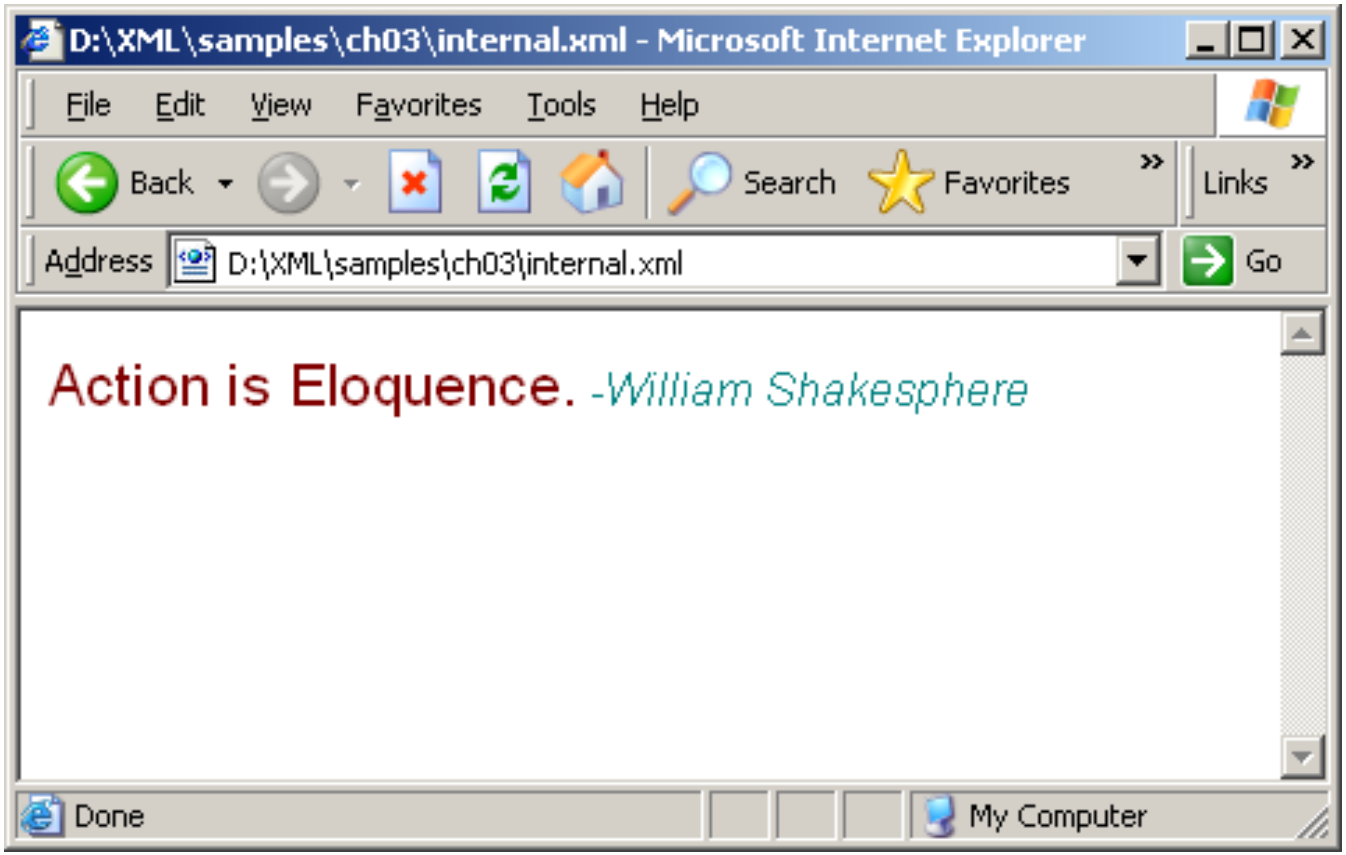
Or

p{font-family:arial}
p{color:blue}

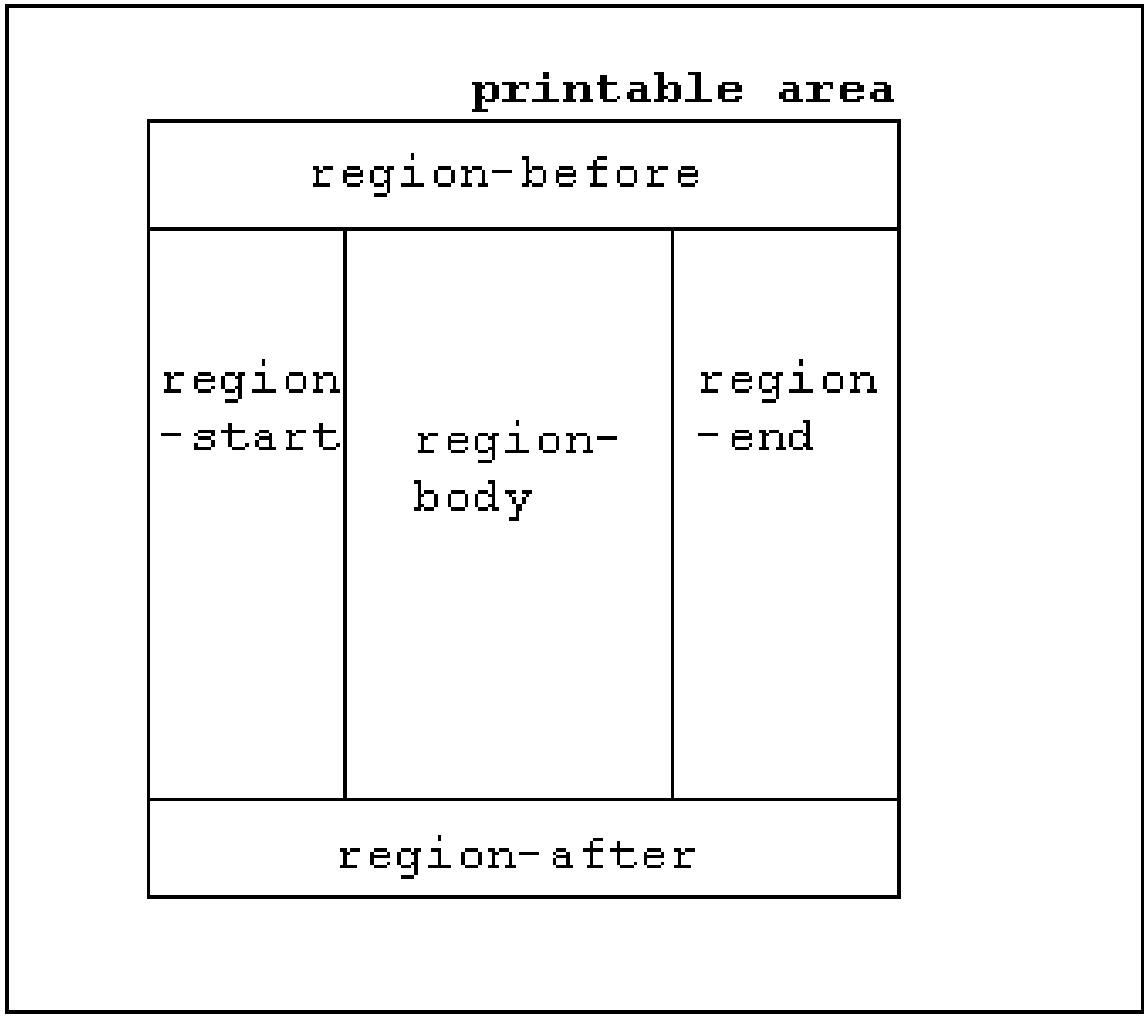
Using CSS to Manage Presentation

```
<?xml version="1.0"?>
<?xml-stylesheet type="text/css" href="#test"?>
<quotations>
   <stylerules id="test">
        quotations {display:block}
        para{font-family:arial}
        .quote{color:maroon; font-size:16pt}
        .name {color:teal; font-style:italic }
        stylerules {display:none}
   </stylerules>
   <para class="quote">Action is Eloquence.</para>
   <para class="name">-William Shakesphere</para>
</quotations>
```

Using CSS to Manage Presentation



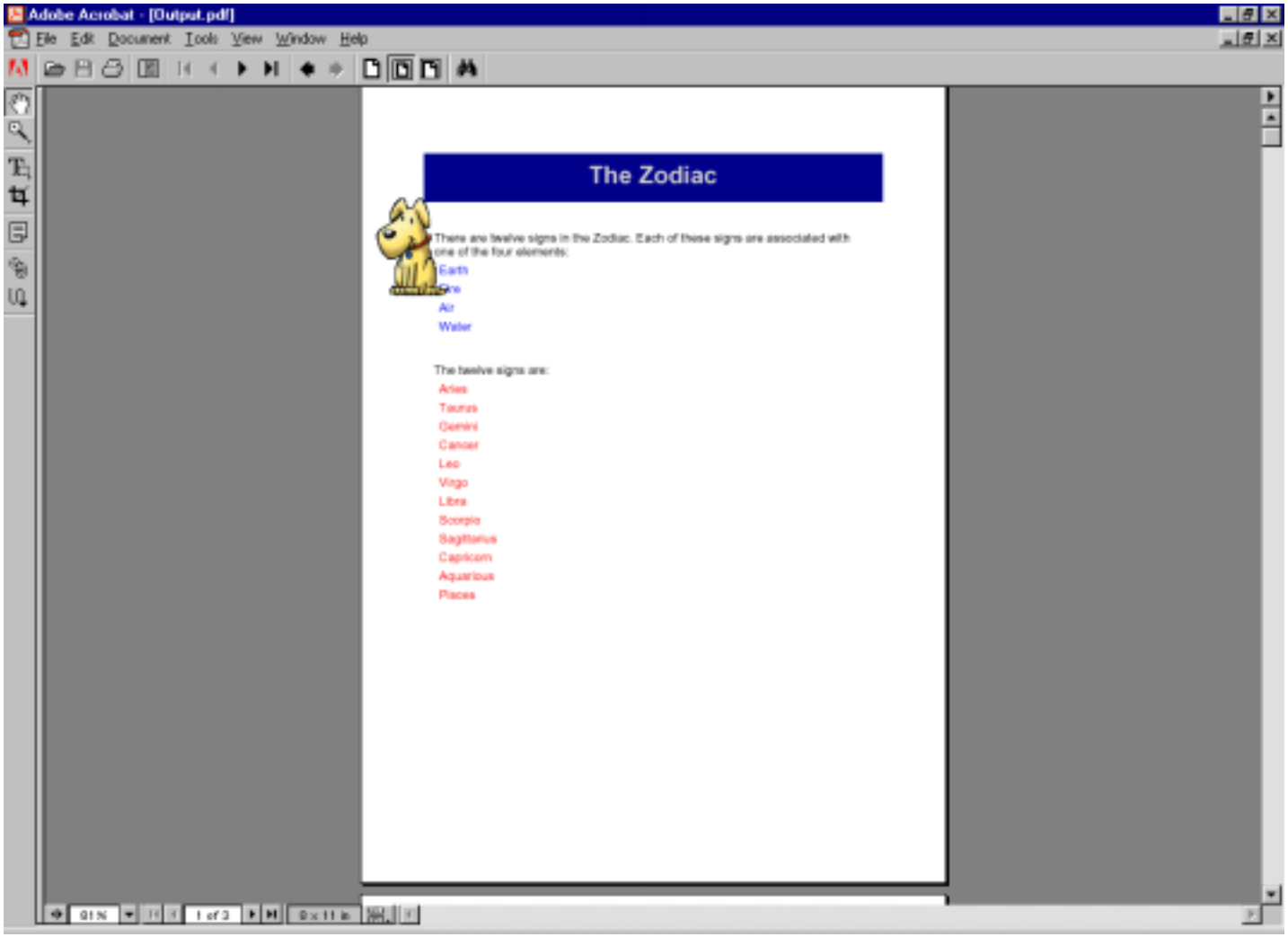
Using XSL-FO to Manage Presentation



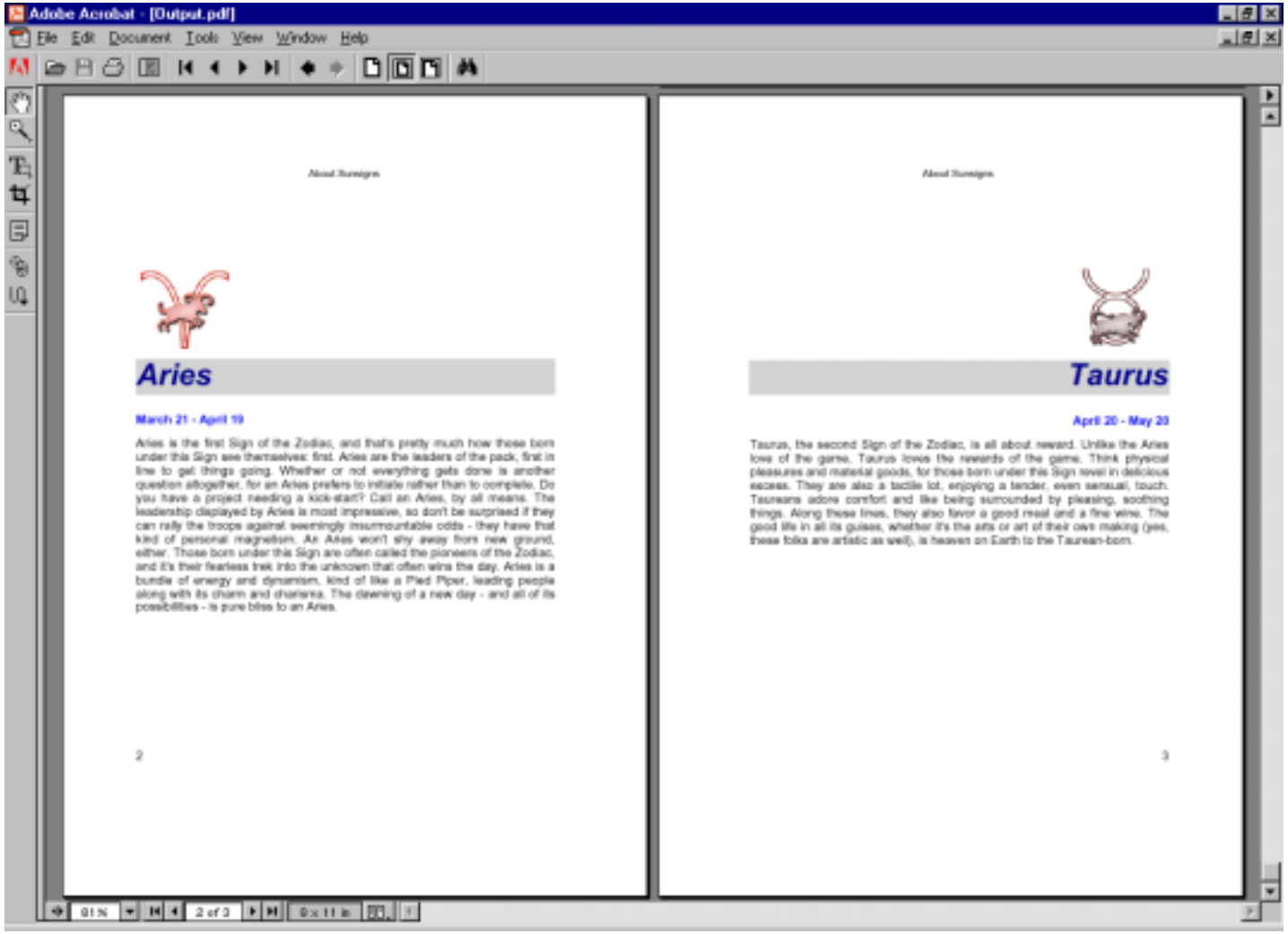
Using XSL-FO to Manage Presentation

```
<?xml version="1.0" encoding="UTF-8"?>
<fo:root xmlns:fo="http://www.w3.org/1999/XSL/Format">
   <fo:simple-page-master master-name="Cover">
       <fo:simple-page-master>
              <fo:region-body></fo:region-body>
              <fo:region-after></fo:region-after>
              <fo:region-before></fo:region-before>
              <fo:region-end></fo:region-end>
              <fo:region-start></fo:region-start>
       </fo:simple-page-master>
   </fo:layout-master-set>
</fo:root>
```

Using XSL-FO to Manage Presentation



Using XSL-FO to Manage Presentation



CSS Vs. XSL-FO

- CSS
  - Provides only basic formatting.
  - Suitable for creating online documents.
- XSL-FO:
  - Enables creation of documents in a variety of formats.
  - Enables complex formatting with features such as:
    - Use different page layouts within the same document.
    - Set page size and numbering.
    - Headers and footers.
    - Page breaks
- XSL-FO is an XML standard. CSS is not.

Defining the Document Structure

- XML enforces adherence to a defined document structure by associating one of the two with an XML document:
  - Document Type Definition (DTD)
  - XML Schema

Enforcing Document Structure Using a DTD

- DTDs are a legacy of SGML.
- The structure and semantics of a particular type of XML document can be defined in a document type definition (DTD) which specifies:
  - The tags that must or may appear in the XML document.
  - Multiplicity of tags.
  - Attributes that may be associated with the various tags.
  - Relationship between the tags.
- Thus, a DTD encapsulates the grammar required to regulate and structure a particular document type.

Enforcing Document Structure Using DTD

- When a DTD is associated with an XML document, it is compared with the DTD to ensure that it is structured as specified in the DTD.
  - The comparison process is called **validation** and is performed by the **validating parser**.
- An XML document that adheres to the rules defined by a document type definition (DTD) is said to be **valid**.

Understanding a DTD

```
<!DOCTYPE Movies [
   <!ELEMENT Movies (Movie*)>
   <!ELEMENT Movie ((Name|Title), LeadActor+, Description?)>
   <!ELEMENT Name (#PCDATA)>
   <!ELEMENT Title (#PCDATA)>
   <!ELEMENT LeadActor (#PCDATA)>
   <!ELEMENT Description (#PCDATA| ReleaseDate | Director)*>
   <!ELEMENT ReleaseDate (#PCDATA)>
   <!ELEMENT Director (#PCDATA)>
]>
```

Enforcing Document Structure Using Schemas

- Schemas are an XML standard for defining document structure.
- Schemas enable complex document structures by:
  - Supporting simple and complex data types.
  - Supporting inheritance that allows vocabularies to be extended.
  - Supporting namespace integration that allows using more than one schema in a XML document.
  - Enabling grouping of elements and attributes, which allows logical grouping of elements and attributes and increases their reusability.

A Schema Fragment

```
<xs:complexType name="MovieDetails">
   <xs:sequence>
       <xs:choice>
              <xs:element name="Title“
                     type="xs:string"/>
       <xs:element name="Name" type="xs:string"/>
       </xs:choice>
       <xs:element name="LeadActor"
              type="xs:string" maxOccurs="2"/> <xs:element
       name="Description“
              type="MovieDescription“ minOccurs="0"/>
       </xs:sequence>
</xs:complexType>
```

DTD Vs. Schemas

- DTD uses non-XML syntax.
- Schemas support simple and complex data types.
- Schemas can be inherited enabling extension of vocabularies
- More than one schema can be associated with an XML document.
- XML schemas allows grouping of elements and attributes increasing their reusability.
- Schemas are complex and verbose.
- Schemas use XML syntax
- DTD treats all data as string.
- DTDs do not allow such extensibility.
- DTDs allow only one to one relationship between the document and DTD.
- DTDs are compact.

XML as an Enabler of Single Sourcing

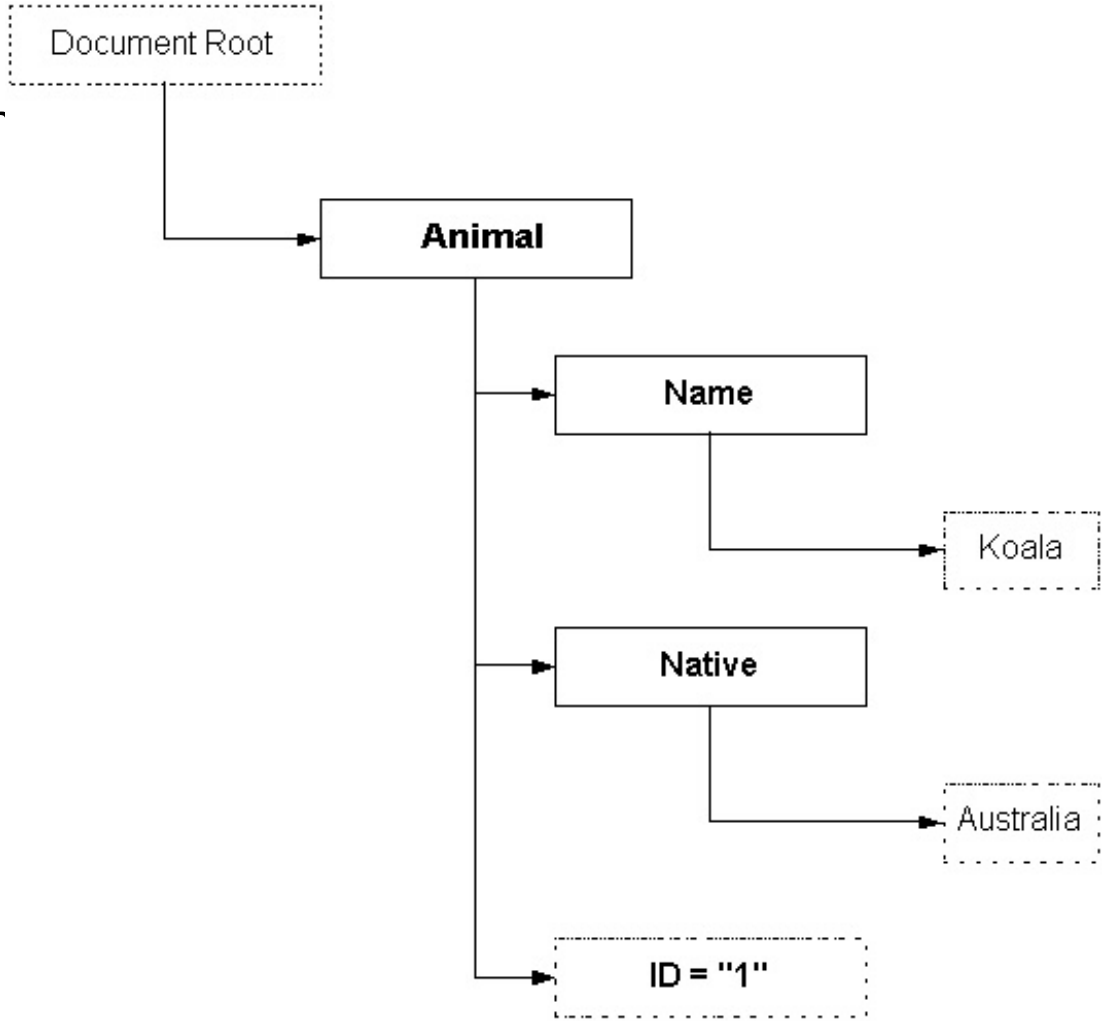
- What are the requirements for single sourcing?
  - Ability to define “named” data.
  - Ability to identify specific data.
  - Ability to extract identified data.
  - Ability to create a new deliverable using identified data.
  - Ability to create a deliverable in the required format.

XML as an Enabler of Single Sourcing

- The XML technologies that address the needs to single sourcing
  - XML
    - Ability to define “named” data.
  - XPath
    - Ability to identify specific data.
  - XSLT
    - Ability to extract identified data.
    - Ability to create a new deliverable using identified data.
    - Ability to create a deliverable in the required format.
  - XLink
    - Creating links

Understanding XPath

- The **XML Path language (XPath)** provides the addressir mechanism required to identify/access portions of the source document.
- **Examples**:
  - `Animal`
  - `Animal[3]`
  - `Animal[native='India']`
  - `text()`



Understanding XSLT

- XSLT is used to transform a given XML document into another XML document by:
  - Deciding what parts of the source are to be included in the result.
  - Manipulating (sort/duplicate/omit) source data as required.
  - Generating additional information on the basis of the data in the source.
  - Add additional information to the target (such as formatting specifications or additional content).

Understanding XSLT

We have animals from these countries:

1.Australia

2.China

3.India, Africa

4.Australia

5.Antarctica

Australian Animals	
ID	Name
1	Koala
4	Kangaroo

Chinese Animals	
ID	Name
2	China

There are 4 students with Grade A. The details are:

Name	Course
Marjory Thomas	Poetry of Robert Frost
Joanna Wooster	Poetry of Robert Frost
Rachel Summers	Organic Chemistry
Bryson Jones	Particle Physics

Panda is from **China**

A Recap

- What did you learn in this session? ☺

Resources

- www.w3schools.com provides an excellent low-level introduction to XML.
- www.XML.com for everything to do with XML.
- Others
  - www.xmlpitstop.com
  - www.devx.com
- Not in the least, TASC Consulting. ☺
  - Training programs
  - Comprehensive courseware
  - Experienced people

Questions?

Contact Us

TASC Consulting Pvt Ltd

D2/10 Khira Nagar,
SV Road, Santacruz(W),
Mumbai – 400054

URL: www.tascindia.com

e-mail: people@tascindia.com

Tel: +91 22 26616686, 26614732