What is XML?

- eXtensible Markup Language, using user-defined **tags**
- XML is a simplified subset of SGML
- Can also be used to define document markup vocabularies (e.g. XHTML)
  - These can have a strictly defined structure (DTD)
- Retains the powerful features of SGML (extensibility, structure, validation)
- Ignores the complex features of SGML and is therefore easier to use and implement
- XML documents look similar to HTML documents
- Separates structure and presentation (like SGML)
- Use references on following slide for syntax and fully worked out consistent examples.

References

- XML
  - Home Page: http://www.w3.org/XML/
  - Tutorial: http://www.w3schools.com/xml/default.asp

- XML Processing
  - Tutorial: http://www.w3schools.com/XML/dom_intro.asp
  - https://docs.oracle.com/javase/tutorial/jaxp/sax/parsing.html
  - Home Pages: http://sax.sourceforge.net/ http://www.w3.org/DOM/

Well Formed XML

```
<?xml version="1.0" ?>
<database>
<person age='34'>
    <name>
        <title> Mr </title>
        <firstname> John </firstname>
        <firstname> Paul </firstname>
        <surname> Murphy </surname>
    </name>
    <hobby> Football </hobby>
    <hobby> Racing </hobby>
</person>

<person >
    <name>
        <firstname> Mary </firstname>
        <surname> Donnelly </surname>
    </name>
</person>
</database>
```

To Be Well-Formed…….

- XML Declaration required
- At least one element
  - Exactly one root element
- Empty elements are written in one of two ways:
  - Closing tag (e.g. "
</br>")
  - Special start tag (e.g. "
")
- For non-empty elements, closing tags are required
- Attribute values must always be quoted
- Start tag must match closing tag (name & case)
- Correct nesting of elements

Physical parts of XML documents

- XML Declaration or Prolog
- Document Type Declaration
- Elements
- Attributes
- Entities
- Character Data Sections
- Processing Instructions
- Comments

- XML Namespaces

Valid XML

- Well-formed **plus** conforms to DTD or XML Schema
  - All elements and attributes are declared within a DTD/XML Schema
  - Elements and attributes match the declarations in the DTD/XML Schema

Express
UML in XML

Document Type Definition(DTD)
Element Declarations
Element Occurances
Entity Declarations
Attribute List Declarations
……

XML Schemas Definitions (XSD)
Simple Type/Complex Types
Structure
Attributes/Attribute Groups
Mixed Content
Empty Elements
……Lots more

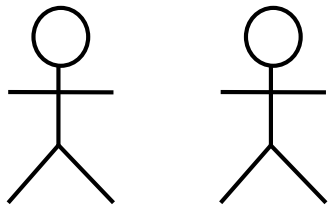
UML
Class Diagram
Use Case

Validated Using DTDs or XSD

BaseX
XML Database
and processor

XML
User Defined Tags
hierarchical Structure
Prolog
Document Type Declaration
Elements
Attributes
Entities
Cdata Sections
Processing Instructions
Comments

NameSpaces
Use
Syntax



Well-Formed

XPath

XQuery

XSL
XSLT

DOCUMENT TYPE DEFINITION (DTD)

What is a DTD?

- Document Type Definition,
- Defines structure/model of XML documents
  - Elements and Cardinality
  - Attributes
  - Aggregation
- Defines default ATTRIBUTE values
- Defines ENTITIES
- Stored in a plain text file and referenced by an XML document **(external)**
- Alternatively a DTD can be placed in the XML document itself **(internal)**

Example DTD

```
<?xml version="1.0" ?>
<database>
<person age='34'>
      <name>
            <title> Mr </title>
            <firstname> John </firstname>
            <firstname> Paul </firstname>
            <surname> Murphy </surname>
      </name>
      <hobby> Football </hobby>
      <hobby> Racing </hobby>
</person>

<person >
      <name>
            <firstname> Mary </firstname>
            <surname> Donnelly </surname>
      </name>
</person>
</database>
```

```
<!DOCTYPE database [

<!ELEMENT database (person*)>

<!ELEMENT person (name,hobby*)>
<!ATTLIST person age CDATA #IMPLIED>

<!ELEMENT name (title?, firstname+,
   surname)>

<!ELEMENT hobby (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT firstname (#PCDATA)>
<!ELEMENT surname (#PCDATA)>

]>
```

- Syntax for occurrences of elements in DTDs
  - ?: zero-or-one
  - +: one-or-more
  - *: zero-or-more

Element Type Declaration

- Define grouping of elements
  (", ")

- Define sequence of elements
  "," means followed-by (Sequence)
  "|" means logical (Choice)

```
<!ELEMENT doc
   (title, author, editor,
   chapter, appendix)>

<!ELEMENT title (#PCDATA)>

<!ELEMENT author
   (name | synonym)>

<!ELEMENT image EMPTY>

<!ELEMENT paragraph
   (#PCDATA | bold | italic)*>
```

Element Type Declaration

- Define occurrences of elements
  - ?: zero-or-one
  - +: one-or-more
  - *: zero-or-more

```
<!ELEMENT doc
   (title, author+, editor?,
   chapter+, appendix*)>

<!ELEMENT chapter
   (title,
    (section+ | paragraph+))>

<!ELEMENT list
   (item?, item?, item)>

<!ENTITY % list "ordered |
   unordered | definition">

<!ELEMENT paragraph
   (#PCDATA | %list;)*>
```

Entity Declaration

- Internal entities
  - Built-in
- External entities
  - References to a file (text, images etc.)
- Parameter entities
  - Used inside DTDs

```
<!ENTITY author
   "Norman Walsh, Sun Corp.">
```

```
<!ENTITY copyright
   SYSTEM "copyright.xml">
```

```
<!ENTITY % part
   "(title?, (paragraph |
   section)*)">
```

Attribute List Declaration

- Define type of attribute
  - CDATA
  - ID
  - ENTITY
  - ……
- Define default values of attributes
  - #REQUIRED
  - #IMPLIED
  - #FIXED
  - A list of values with default selection

```
<!ATTLIST person
   ssn ID #IMPLIED>

<!ATTLIST adult
   age CDATA #REQUIRED>

<!ATTLIST mml
   version ‘1.0’ #FIXED>

<!ATTLIST person
    sex (m | f) #REQUIRED>

<!ATTLIST day
   temperature (l | m | h) "l">
```

Simple DTD Example

```
<!DOCTYPE doc[
<!ENTITY % part "(title?, (paragraph | section)*)">

<!ELEMENT doc (title, author+, chapter+, appendix*)>
<!ATTLIST doc type (book | article) "book“
              isbn CDATA #REQUIRED>

<!ELEMENT title (#PCDATA)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT chapter %part;>
<!ELEMENT appendix %part;>
<!ELEMENT section %part;>
<!ELEMENT paragraph (#PCDATA | url | ol)*>
<!ATTLIST paragraph type CDATA #IMPLIED>
<!ELEMENT ol (item+)>
<!ELEMENT item (paragraph+)>
<!ELEMENT url (#PCDATA)>
]>
```

```
<?xml version="1.0"?>
<!DOCTYPE catalog SYSTEM "books.dtd">
<catalog>
   <book id='bk101' type='softback'>
      <author>Gambardella, Matthew</author>
      <title>XML Developer's Guide</title>
      <genre>Computer</genre>
<price>44.95</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at creating
applications with XML.
</description>
</book>
<book id='bk102' type='hardback'>
      <author nationality='irish'>Jenkins,
Fred</author>
      <title>XML Technology Guide</title>
         <price>50.00</price>
      <publish_date>2000-10-01</publish_date>
      <description>An in-depth look at using XML
technologies.</description>
         <stocked_by>Easons</stocked_by>
         <stocked_by>Amazon</stocked_by>
   </book >
</catalog>
```

SUGGEST A DTD

XML NAMESPACES & XML SCHEMA

What are XML Namespaces?

- W3C recommendation (January 1999)
- Each XML vocabulary is considered to own a namespace in which all elements (and attributes) are unique
- **A single document can use elements and attributes from multiple namespaces**
  - A prefix is declared for each namespace used within a document.
  - The namespace is identified using a URI (Uniform Resource Identifier)
- An element or attribute can be associated with a namespace by placing the namespace prefix before its name (i.e. '*prefix*:*name*')
  - Elements (and attributes) belonging to the default namespace do not require a prefix

Example: XML Namespaces

St. James's Hospital

```
<!ELEMENT Patient (Name, DOB)>

<!ELEMENT Name   (First, Last)>
<!ELEMENT First  (#PCDATA)>
<!ELEMENT Last   (#PCDATA)>
<!ELEMENT DOB    (#PCDATA)>
```

Airport Pharmacy

```
<!ELEMENT Drug
     ((Name|Substance), Code)>

<!ELEMENT Name       (#PCDATA)>
<!ELEMENT Substance  (#PCDATA)>
<!ELEMENT Code       (#PCDATA)>
```

```
<?xml version='1.0'?>

<Accident Report
  xmlns:sjh="http://hospital/sjh"
  xmlns:dub=http://airport/dub >

 <sjh:Patient>
  <sjh:Name>
   <sjh:First>Mike</sjh:First>
   <sjh:Last>Murphy</sjh:Last>
  </sjh:Name>
  <sjh:DOB>12/12/1950</sjh:DOB>
 </sjh:Patient>

 <dub:Drug>
  <dub:Name>Nurofen</dub:Name>
  <dub:Code>IE-975-2</dub:Code>
 </dub:Drug>

   [...]
</Accident Report>
```

What are XML Schemas?

- W3C Recommendation, 2 May 2001
  - Part 0: Primer
  - Part 1: Structures
  - Part 2: Datatypes
- DTDs use a non-XML syntax and have a number of limitations
  - no namespace support
  - lack of data-types
- XML Schemas are an alternative to DTDs
- Supports simple/complex data-types
- https://www.w3schools.com/xml/xml_schema.asp

Why use XML Schemas?

- Uses an XML syntax
- Supports simple and complex data-types such as user-defined types
- An XML document and its contents can be validated against a Schema
- Can validate documents containing multiple namespaces
- Schemas are more powerful than DTDs and will eventually replace DTDs

Named Types – simple (can contain “only text”)

DTD

```
<!ELEMENT birthday(#PCDATA)>
```

XML Schema

```
<xsd:element name=” birthday" type="xsd:date"/>
```

XML doc. Instance

```
<birthday>01 March 2001</birthday>
```

Named Types – complex(can contain elements and attributes)

DTD

```
<!ELEMENT student_name (firstname, lastname)>
```

XML Schema

```
<xsd:complexType name="namePerson">
   <xsd:sequence>
     <xsd:element name="firstname" type="xsd:string"/>
     <xsd:element name="lastname" type="xsd:string/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name = "student_name" type="namePerson"/>
```

XML doc. Instance

```
<student_name>
   <firstname>Michael</firstname>
   <lastname>Porter</lastname>
</student_name>
```

Simple Type - Restriction

XML Schema

```
<simpleType name='celsiusBodyTemp'>
  <restriction base='decimal'>
    <totalDigits value='4'/>
    <fractionDigits value='1'/>
    <minInclusive value='36.4'/>
    <maxInclusive value='40.5'/>
  </restriction>
</simpleType>
<xsd:element name="temp" type="celsiusBodyTemp"/>
```

XML doc. Instance

```
<temp>37.2</temp>
```

Simple Type - Enumeration

XML Schema

```
<xsd:simpleType name="weekday">
   <xsd:restriction base="xsd:string">
     <xsd:enumeration value="Sunday"/>
     <xsd:enumeration value="Monday"/>
     <xsd:enumeration value="Tuesday"/>
     [...]
   </xsd:restriction>
</xsd:simpleType>
<xsd:element name="delivery" type="weekday"/>
```

XML doc. Instance

```
<delivery>Tuesday</delivery>
```

Complex Type - Cardinalities

DTD

```
<!ENTITY % fullname "title?, firstname*, lastname">
<!ELEMENT student_name (%fullname;)>
```

XML Schema

```
<xsd:complexType name="fullname">
   <xsd:sequence>
     <xsd:element name="title" minOccurs="0"/>
     <xsd:element name="firstname" minOccurs="0"
                  maxOccurs="unbounded"/>
     <xsd:element name="lastname"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name=" student_name" type="fullname"/>
```

XML doc. Instance

```
<student_name>
   <firstname>Michael</firstname>
   <firstname>Jason</firstname>
   <lastname>Porter</lastname>
</student_name>
```

Complex Type – Derived Type by extension

DTD

```
<!ENTITY % name "title?, firstname*, lastname">
<!ELEMENT student_name (%name;, maidenname?)>
```

XML Schema

```
<xsd:complexType name="fullnameExt">
   <xsd:complexContent>
     <xsd:extension base="fullname">
       <xsd:sequence>
         <xsd:element name="maidenname" minOccurs="0"/>
       </xsd:sequence>
     </xsd:extension>
   </xsd:complexContent>
</xsd:complexType>
<xsd:element name=" student_name" type="fullnameExt"/>
```

XML doc. Instance

```
<student_name>
   <firstname>Jane</firstname>
   <lastname>Porter</lastname>
   <maidenname>Hughes</maidenname>
</student_name>
```

Complex Type – Derived Type by Restriction

XML Schema

```
<xsd:complexType name="simpleName">
   <xsd:complexContent>
     <xsd:restriction base="fullname">
       <xsd:sequence>
         <xsd:element name="title" maxOccurs="0"/>
         <xsd:element name="firstname" minOccurs="1"/>
         <xsd:element name="lastname"/>
       </xsd:sequence>
     </xsd:restriction>
   </xsd:complexContent>
</xsd:complexType>
```

XML doc. Instance

```
<name>
   <firstname>Jane</firstname>
   <lastname>Porter</lastname>
</name>
```

Structure - Sequence

DTD

```
<!ELEMENT student_name (title?, firstname*, lastname)>
```

XML Schema

```
<xsd:complexType name="fullname">
   <xsd:sequence>
     <xsd:element name="title" minOccurs="0"/>
     <xsd:element name="firstname" minOccurs="0"
                  maxOccurs="unbounded"/>
     <xsd:element name="lastname"/>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name=" student_name" type="fullname"/>
```

XML doc. Instance

```
<student_name>
   <firstname>Michael</firstname>
   <firstname>Jason</firstname>
   <lastname>Porter</lastname>
</student_name>
```

Structure - Choice

DTD

```
<!ELEMENT pay (product, number, (cash | cheque))>
```

XML Schema

```
<xsd:complexType name="payment">
   <xsd:sequence>
     <xsd:element ref="product"/>
     <xsd:element ref="number"/>
     <xsd:choice>
       <xsd:element ref="cash"/>
       <xsd:element ref="cheque"/>
     </xsd:choice>
   </xsd:sequence>
</xsd:complexType>
<xsd:element name="pay" type="payment"/>
```

XML doc. Inst.

```
<pay>
   <product>Ericsson Telefon MD110</product>
   <number>1544-198-J</number>
   <cash>EUR150</cash>
</pay>
```

Attributes-

DTD

```
<!ELEMENT greeting (#PCDATA)>
<!ATTLIST greeting language CDATA "English">
```

XML Schema

```
<xsd:element name="greeting">
   <xsd:complexType>
     <xsd:simpleContent>
       <xsd:extension base="xsd:string">
         <xsd:attribute name="language" type="xsd:string"/>
       </xsd:extension>
     </xsd:simpleContent>
   </xsd:complexType>
</xsd:element>
```

XML doc. Instance

```
<greeting language="German">Hello!</greeting>
```

Attribute Groups

DTD

```
<!ELEMENT img EMPTY>
<!ATTLIST img src CDATA #REQUIRED
          width CDATA #IMPLIED
          height CDATA #IMPLIED>
```

XML Schema

```
<xsd:attributeGroup name="imgAttributes">
   <xsd:attribute name="src" type="xsd:string" use="required"/>
   <xsd:attribute name="width" type="xsd:integer"/>
   <xsd:attribute name="height" type="xsd:integer"/>
</xsd:attributeGroup>

<xsd:element name="img">
   <xsd:complexType>
     <xsd:attributeGroup ref="imgAttributes"/>
   <xsd:complexType>
</xsd:element>
```

XML Inst.

```
<img src="XMLmanager.gif" width="60"/>
```

Mixed Content

DTD

```
<!ELEMENT p (#PCDATA | b | i)*>
<!ELEMENT b (#PCDATA)>
```

XML Schema

```
<xsd:complexType name="bolditalicText" mixed="true">
   <xsd:choice minOccurs="0" maxOccurs="unbounded"/>
     <xsd:element ref="b" />
     <xsd:element ref="i" />
   </xsd:choice>
</xsd:complexType>

<xsd:element name="p" type="bolditalicText"/>
```

XML doc. Instance

```
<p>This is <b>bold</b> and <i>italic</i> text</p>
```

Empty Element

DTD

```
<!ELEMENT img EMPTY>
<!ATTLIST src CDATA #REQUIRED>
```

XML Schema

```
<xsd:element name="img">
   <xsd:complexType>
     <xsd:attribute name="src" type="xsd:string"/>
   </xsd:complexType>
</xsd:element>
```

XML doc. Instance

```
<img src="XMLmanager.gif"/>
```

XML Schema Example

```
<?xml version="1.0" encoding="utf-8"?>

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:element name="book">

    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="title" type="xsd:string"/>
        <xsd:element name="author" type="xsd:string"/>
        <xsd:element name="character" type="xsd:string"
                     minOccurs="0" maxOccurs="unbounded">
        </xsd:element>
      </xsd:sequence>

      <xsd:attribute name="isbn" type="xsd:string"/>
    </xsd:complexType>

  </xsd:element>
</xsd:schema>
```

Summary

- XML Vocabularies are defined using
  - DTD
  - XSD
- DTDs/XSDs used to validate XML documents
- XSD – more powerful than DTDs
  - Supports simple and complex data-types such as user-defined types
  - Can validate documents containing multiple namespaces