The Why and What of XML

•Extensible Markup Language.

•Extensible because it is not a fixed format like HTML

•XML is a “metalanguage”

–A language used to describe other languages using “markup”

•XML is a successor of SGML

–Standard Generalized Markup Language

What is SGML?

- International standard
- To define the structure of different types of electronic document.
- SGML is very large, powerful, and complex.
- XML is a lightweight cut-down version of SGML

Why XML?

- File format for the storage and transmission of text
- Platform independent
- Universally accepted standard
- Non-private
- Robust

Why XML? (cont.)

- W3C recommends it
- Marketplace support with lots of free/inexpensive tools

Well-formed and Valid XML

- Well-formed:
  - the XML declaration come first in every document
    - <?xml version="1.0" ?>
  - comments are not valid within a tag
  - comments may not contain two hyphens in a row, other than the beginning and end of the comment

Well-formed XML (Cont.)

- tags must have an end tag, or be closed within the singleton tag itself, for example
- all attributes of tags must be quoted
- every XML document must contain one element that completely contains all the other elements

Valid XML and Schemas

- Schema:
  - The rules that establish the format and structure of documents
- A Schema is used to check the validity of an XML document.

Types of Schemas

- Document Type Definition (DTDs)
- XML Schemas (XSDs)

DTDs

```
<!ELEMENT trainlog (session)+>

<!ELEMENT session (duration, distance, location, comments)>
<!ATTLIST session
  date CDATA #IMPLIED
  type (running | swimming | cycling) "running"
  heartrate CDATA #IMPLIED
>

<!ELEMENT duration (#PCDATA)>
<!ATTLIST duration
  units (seconds | minutes | hours) "minutes"
>

<!ELEMENT distance (#PCDATA)>
<!ATTLIST distance
  units (miles | kilometers | laps) "miles"
>

<!ELEMENT location (#PCDATA)>

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

XSDs

- XML Schema Definition Language
- An official W3C standard
- XSD is written in XML

XSD Data Types

- Simple Data Types
  - String types: xsd:string
  - Boolean types: xsd:boolean
  - Number types: xsd:integer, xsd:decimal, xsd:float, …
  - Date and Time types: xsd:time, xsd:date, …

  For Example:

  <xsd:element name="name" type="xsd:string/>

  <name>Vahak Matavosian</name>

XSD Data Types (cont.)

- Complex Data Types
  - Empty elements: no text or child (attributes)

```
<xsd:element name="automobile">
    <xsd:ComplexType>
     <xsd:ComplexContent>
       <xsd:extension base="xsd:anyType">
            <xsd:attribute name="vin" type="xsd:string"/>
            <xsd:attribute name="year" type="xsd:year"/>
            <xsd:attribute name="make" type="xsd:string"/>
            <xsd:attribute name="model" type="xsd:string"/>
       </xsd::extension>
     </xsd:ComplexContent>
    </xsd:ComplexType>
</xsd:element>

<automobile vin="SALH12345678" year="2004" make="Honda" model="Civic"/>
```

XSD Data Types (cont.)

- Element-Only elements: contain only elements with no text and child content
- Mixed Elements: contain both text and child
- Sequences And Choices:
  - A *sequence* is a list of child elements that must appear in a particular order
  - A *choice* is a list elements that must be used

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

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema">
  <xsd:element name="trainlog">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="session" type="sessionType" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:complexType name="sessionType">
    <xsd:sequence>
      <xsd:element name="duration" type="xsd:timeDuration"/>
      <xsd:element name="distance" type="distanceType"/>
      <xsd:element name="location" type="xsd:string"/>
      <xsd:element name="comments" type="xsd:string"/>
    </xsd:sequence>

    <xsd:attribute name="date" type="xsd:date" use="required"/>
    <xsd:attribute name="type" type="typeType" use="required"/>
    <xsd:attribute name="heartrate" type="xsd:positiveInteger"/>
  </xsd:complexType>
```

```
<xsd:complexType name="distanceType">
  <xsd:simpleContent>
    <xsd:extension base="xsd:decimal">
      <xsd:attribute name="units" type="unitsType" use="required"/>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>

<xsd:simpleType name="typeType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="running"/>
    <xsd:enumeration value="swimming"/>
    <xsd:enumeration value="cycling"/>
  </xsd:restriction>
</xsd:simpleType>

<xsd:simpleType name="unitsType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="miles"/>
    <xsd:enumeration value="kilometers"/>
    <xsd:enumeration value="laps"/>
  </xsd:restriction>
</xsd:simpleType>
```

Validating XML Documents

- Schemas allow us to:
  - Establish the elements and their attributes that can appear in a document
  - Determine whether the element is empty or not
  - Determine the number of sequence of child elements within an element

DTD vs. XSD

- DTDs:
  - Well supported
  - Easier to create
  - Use special language
- XSDs:
  - Use XML
  - Support data types

Processing XML Data

- Simple API for XML (SAX)
- Document Object Model (DOM)

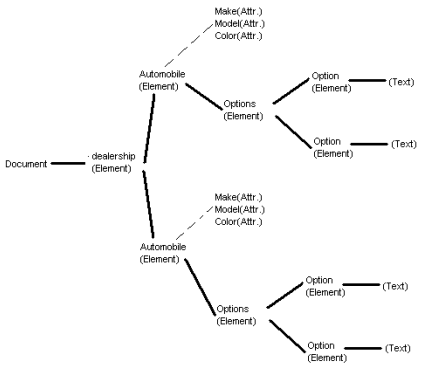
SAX

- Originally a Java-only API.
- is a “de facto” standard.
- Interface for event-based parsing of XML
- Linear
- Triggers certain events
- SAX in other languages:
  - http://www.saxproject.org/?selected=langs

DOM

- Standard from W3C
- DOM represents an XML document as a tree structure
- Everything is a *node*

```
<?xml-stylesheet href="mystyle.css" type="text/css"?>
<dealership>
  <automobile make="Buick" model="Century" color="blue">
    <options>
      <option>cruse control</option>
      <option>cd player</option>
    </options>
  </automobile>
  <automobile make="Ford" model="Thundrebird" color="red">
    <options>
      <option>convertible</option>
      <option>leather interior</option>
    </options>
  </automobile>
</dealership>
```



SAX vs. DOM

- interprets XML as a stream of events
- you supply event-handling callbacks
- SAX parser invokes your event-handlers as it parses
- doesn't build data model in memory
- serial access
- very fast, lightweight
- good choice when
  - no data model is needed, or
  - natural structure for data model is list, etc.

- W3C standard for representing structured documents
- interprets XML as a tree of nodes
- builds data model in memory
- enables random access to data
- therefore good for interactive apps
- more CPU- and memory-intensive
- good choice when data model has natural tree structure