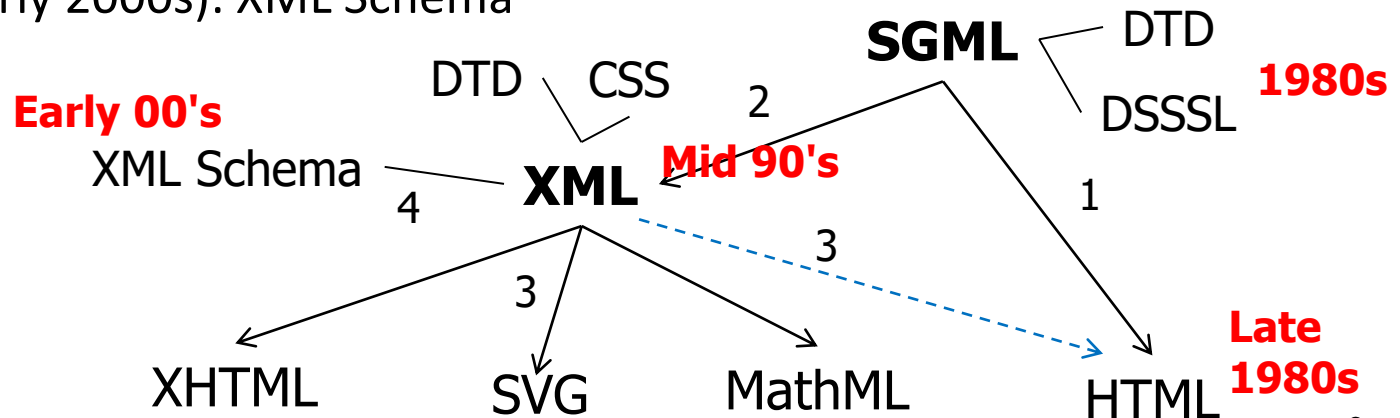


COMP 4021
Internet Computing

XML

History

- Mid 1980s: Information retrieval, library and publishing communities developed SGML (Standard Generalized Markup Language)
 - <tag>, DTD and DSSSL (father of CSS) are part of SGML technologies
 - Hypertext concepts were developed (Apple's HyperCard, linking of info **across** machines)
- Step 1 (Late 1980s): HTML taken some feature from SGML (e.g., tagging); DTD existed for HTML but not used very much
- Step 2 (Mid 1990s): XML as a simplified version of SGML
- Step 3 (Late 1990s till now): XHTML, SVG, MathML, etc., etc.
- Step 4 (Early 2000s): XML Schema



XML eXtensible Markup Language

- HTML is for markup of documents
 - XML **specifies** the structure/ content of a document
 - XML doesn't describe any visual appearance
- *It is **NOT** a language that allows you to add more commands and functions to make up a more powerful (and hence bigger) language*
- *It is a language that allows you to **define** a new language*
- You need a grammar to define a new language; XML allows you to define a grammar using Document Type Definition (DTD)

Tags, Elements and Attributes

Element: The "address" element contains four sub-elements, name, street, city and postal-code

Tag: start tag

```
<address>
```

Attribute:
Name=value
pairs inside
start tags

```
<name>
```

```
<title>Mrs.</title>
```

```
<first-name>
```

```
Mary
```

```
</first-name>
```

```
<last-name>
```

```
McGoon
```

```
</last-name>
```

```
</name>
```

```
<street>
```

```
1401 Main Street
```

```
</street>
```

```
<city state="NC">Anytown</city>
```

```
<postal-code>
```

```
34829
```

```
</postal-code>
```

```
</address>
```

Tag: end tag

Well-formed, Valid and Invalid XML

- Well-formed documents: Follow the XML syntax rules but don't have a DTD or schema
- Valid documents: Follow both the XML syntax rules and the rules defined in their DTD or schema
- Invalid documents: Don't follow the XML syntax rules or the DTD or schema, if available

XML Syntax Rules (I)

- The root element: An XML document must be contained in a single element called the root element

```
<?xml version="1.0"?>
<!-- A well-formed document -->
<greeting>
  Hello, World!
</greeting>
```

```
<?xml version="1.0"?>
<!-- An invalid document -->
<greeting>
  Hello, World!
</greeting>
<greeting>
  Hola, el Mundo!
</greeting>
```

- XML elements can't overlap.

```
<!-- NOT legal XML markup -->
<p>
  <b>I <i>really love</b> XML.</i>
</p>
```

XML Syntax Rules (II)

- **End tags** are required; note how empty elements are handled

```
<!-- NOT legal XML markup -->
<p>Yada yada yada...
<p>Yada yada yada...
<p>...
```

```
<!-- Two equivalent break elements -->
<br></br>
<br />
```

```
<!-- Two equivalent image elements -->
</img>

```

- Elements are **case sensitive** (convention is to use lower case as much as possible)

```
<!-- NOT legal XML markup -->
<h1>Elements are
  case sensitive</H1>
```

```
<!-- legal XML markup -->
<h1>Elements are
  case sensitive</h1>
```

XML Syntax Rules (III)

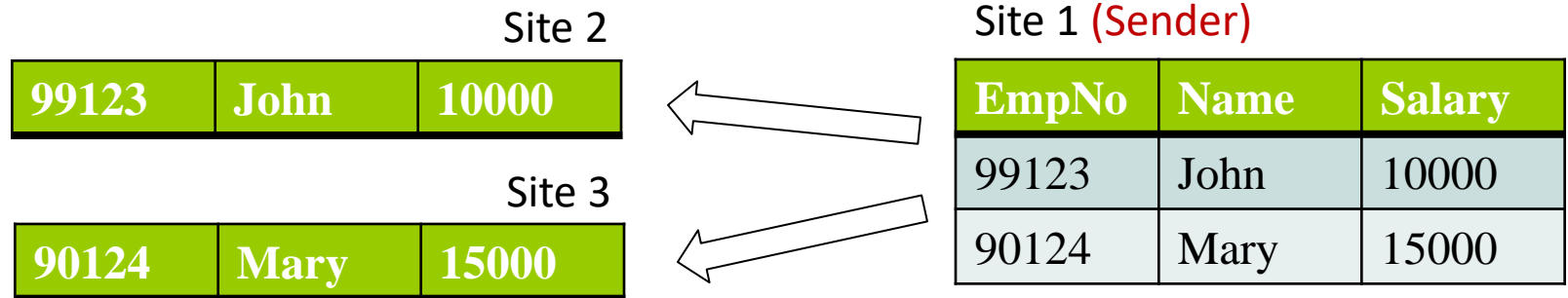
- An attribute, if specified, must have a value
- Attribute values must be double or single quoted

```
<!-- NOT legal XML markup -->  
<ol compact>  
  
<!-- legal XML markup -->  
<ol compact="yes">
```

- Parameter values are enclosed in speech marks
I.e. <circle id="face_outline" ... />
- E.g., <td nowrap>...</td> OK in html4; Not OK in html5

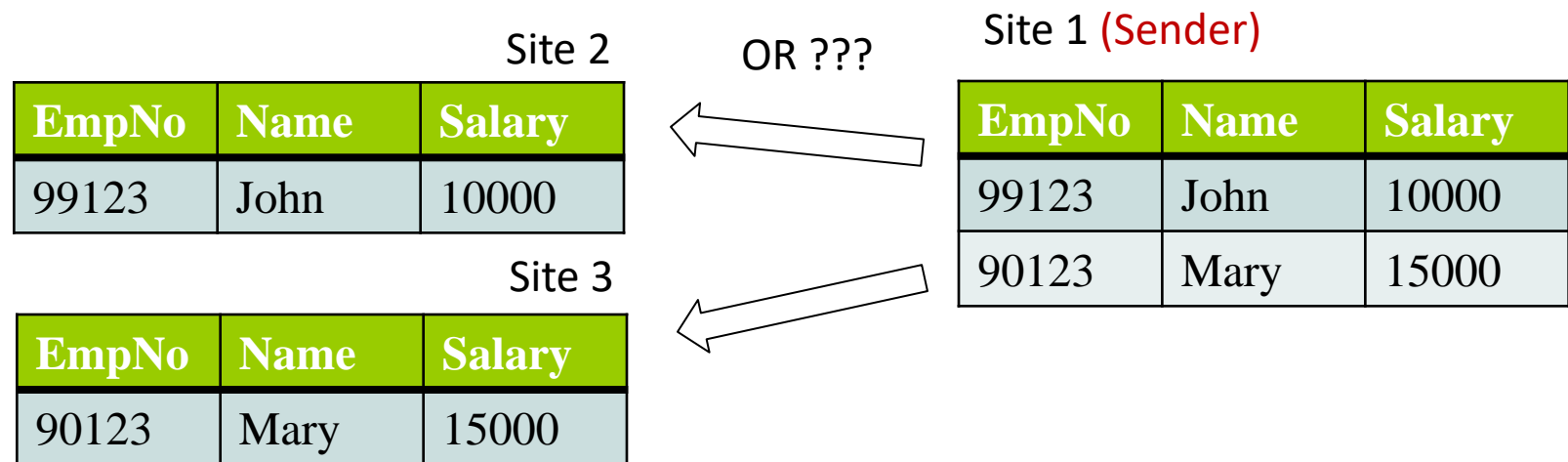
Why XML: Standard for Data Exchange

- XML is an standard for data exchange
- With DTD/XML Schema, an XML file can be validated
- XML data is self described



What do these values **mean**?

Why XML: Standard for Data Exchange



OR CSV, TXT, TSV, etc. ???

What if the data is binary?

If the table is stored in Oracle, can you simply send the table?

Why XML: Standard for Data Exchange

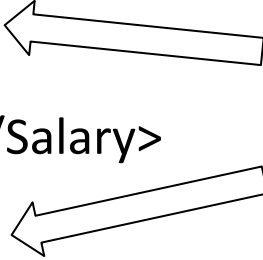
- XML data is **self described**

Site 2

```
<Employee>
<EmpNo>99123</EmpNo>
<Name>John</Name>
<Salary currency="JPY">10000</Salary>
</Employee>
```

Site 3

```
<Employee>
<EmpNo>90123</EmpNo>
<Name>Mary</Name>
<Salary currency="USD">15000</Salary>
</Employee>
```

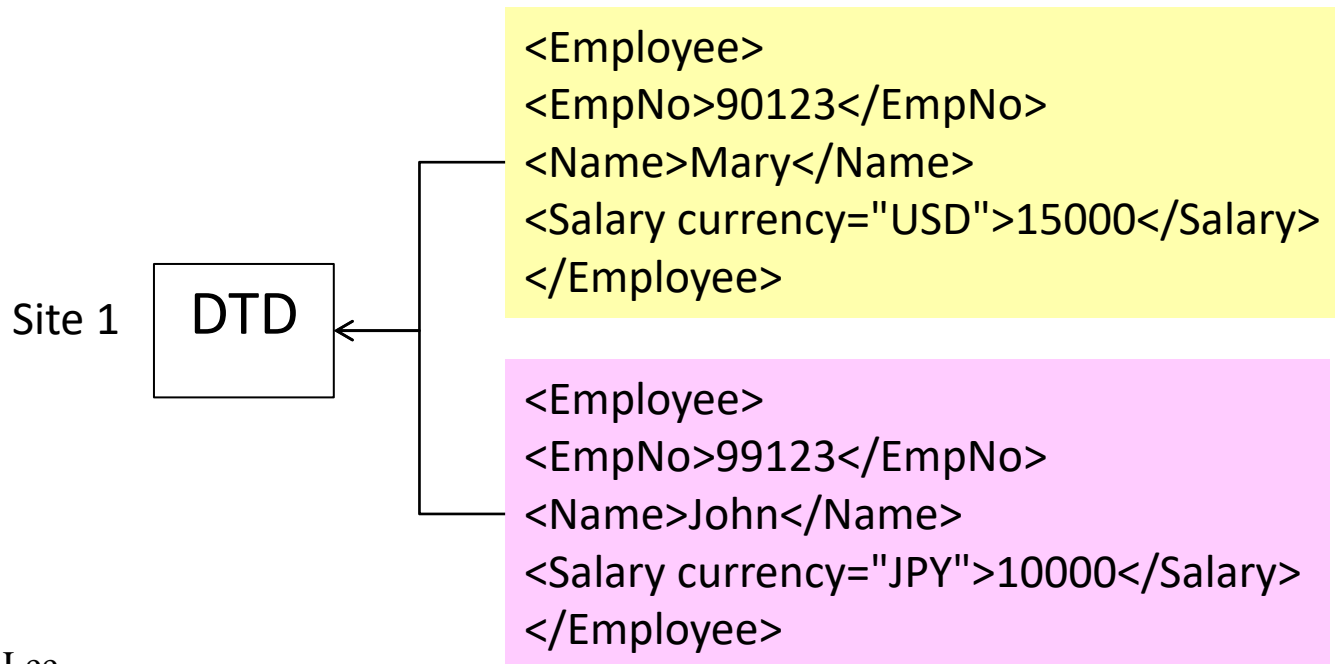


Site 1

EmpNo	Name	Salary
99123	John	10000
90123	Mary	15000

Why XML: Standard for Data Exchange

- XML data is Unicode based, thus supporting multiple languages in the same file
- By sharing the same DTD, a site can **validate** the XML data received from another site before using it



Take Home Message

- XML is the foundation of Web languages
- XML is **a language for defining new languages** (including HTML, SVG, etc.)
- XML appears to be bulky but it is good for data exchange across distributed websites (see next set of slides)
- There are many ways to render XML
- XSL is a complete XML language specifically for manipulating XML data
 - No longer under development by W3C (last update Jan 2012; official announcement to stop further development made in Nov 2013)
 - XSL/XSLT are still supported by all major browsers

COMP 4021
Internet Computing

XML DTD / Schema

Document Type Definitions (DTD)

- DTD defines the structure of an XML document by imposing constraints
 - The set of legal elements
 - An attribute is required or not
 - A certain element can only exist within a specific element (e.g., <price> must be nested within <item>)
 - A certain element must exist or not (if must exist, exist once or more)
 -

```
<!DOCTYPE address
[
  <!ELEMENT address (name, street, city,
state, postal-code)>
  <!ELEMENT name (title?, first-name,
last-name)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT first-name (#PCDATA)>
  <!ELEMENT last-name (#PCDATA)>
  <!ELEMENT street (#PCDATA)>
  <!ELEMENT city (#PCDATA)>
  <!ELEMENT state (#PCDATA)>
  <!ELEMENT postal-code (#PCDATA)>
]>
```

PCDATA: Parsed Character Data

Constraints in DTD (I)

```
<!ELEMENT address (name, city, state)>
```

- The <address> element must contain a <name>, a <city>, and a <state> element, in that order. **All of the elements are required.**

```
<!ELEMENT name (title?, first-name, last-name)>
```

- The <name> element contains an optional <title> element, followed by a mandatory <first-name> and a <last-name> element; the question mark means **zero or one occurrence**

```
<!ELEMENT addressbook (address+)>
```

- An <addressbook> element contains **one or more <address>** elements; plus sign means an item must appear at least once

Constraints in DTD (II)

```
<!ELEMENT private-addresses (address*)>
```

- A <private-addresses> element contains zero or more <address> elements; the asterisk indicates zero or more occurrences

```
<!ELEMENT name (title?, first-name, (middle-initial | middle-name)?, last-name)>
```

- A <name> element contains an optional <title> element, followed by a <first-name> element, followed by zero or one of <middle-initial> or a <middle-name> element, followed by a <last-name> element; vertical bars indicate a list of choices

```
<!ELEMENT name ((title?, first-name, last-name) | (surname, mothers-name, given-name))>
```

- The <name> element can contain one of two sequences:
 - An optional <title>, followed by a <first-name> and a <last-name>
 - A <surname>, a <mothers-name>, and a <given-name>.

Defining Attributes in DTD

- Define which attributes are required
- Define default values for attributes
- List all of the valid values for a given attribute

Defines Element city and its attributes

```
<!ELEMENT city (#PCDATA)>  
<!ATTLIST city state CDATA #REQUIRED  
              postal-code CDATA #REQUIRED>
```

```
<!ELEMENT city (#PCDATA)>  
<!ATTLIST city state CDATA (AZ | CA | NV | OR | UT | WA) "CA">
```

Validating XML Data Against DTD

```
<address>
  <name>
    <title>Prof.</title>
    <first-name>Dik</first-name>
    <last-name>Lee</last-name>
  </name>
  <street>Queen's Road East</street>
  <city>Hong Kong</city>
  <state>N/A</state>
  <postal-code>12345</postal-code>
</address>
```

- Is it a valid XML record?

```
<!DOCTYPE address
[
  <!ELEMENT address (name, street, city,
state, postal-code)>
  <!ELEMENT name (title?, first-name,
last-name)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT first-name (#PCDATA)>
  <!ELEMENT last-name (#PCDATA)>
  <!ELEMENT street (#PCDATA)>
  <!ELEMENT city (#PCDATA)>
  <!ELEMENT state (#PCDATA)>
  <!ELEMENT postal-code (#PCDATA)>
]>
```

What about this?

```
<address>
  <name>
    <title>Prof.</title>
    <first-name>Dik</first-name>
    <last-name>Lee</last-name>
  </name>
  <street>Queen's Road East</street>
  <city capital="y">Hong Kong</city>
  <state>N/A</state>
  <postal-code>12345</postal-code>
</address>
```

- Is it a valid XML record?

Demo

```
<!DOCTYPE address
[
  <!ELEMENT address (name, street, city,
state, postal-code)>
  <!ELEMENT name (title?, first-name, last-
name)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT first-name (#PCDATA)>
  <!ELEMENT last-name (#PCDATA)>
  <!ELEMENT street (#PCDATA)>
  <!ELEMENT city (#PCDATA)>
  <!ELEMENT state (#PCDATA)>
  <!ELEMENT postal-code (#PCDATA)>
  <!ATTLIST city capital CDATA #REQUIRED>
]>
```

Invalid XML Record

```
<address>
  <name>
    <title>Prof.</title>
    <last-name>Lee</last-name>
    <first-name>Dik</first-name>
  </name>
  <street>Queen's Road East</street>
  <city capital=y>Hong Kong</city>
  <postal-code>12345</postal-code>
</address>
```

- Identify the mistakes

Demo

Dik Lun Lee

```
<!DOCTYPE address
[
  <!ELEMENT address (name, street, city,
state, postal-code)>
  <!ELEMENT name (title?, first-name, last-
name)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT first-name (#PCDATA)>
  <!ELEMENT last-name (#PCDATA)>
  <!ELEMENT street (#PCDATA)>
  <!ELEMENT city (#PCDATA)>
  <!ELEMENT state (#PCDATA)>
  <!ELEMENT postal-code (#PCDATA)>
  <!ATTLIST city capital CDATA #REQUIRED>
]>
```

XML Schema

- XML schemas are themselves XML documents
 - A schema can be processed just like any other document
 - You can convert an XML schema into a Web form complete with automatically generated JavaScript code to validate the input data
- XML schemas support **more data types** than DTDs
 - Most of the data types in a programming language are supported
- XML schemas are extensible
 - **User-defined and derived** data types are supported
- XML schemas have **more expressive power**
 - XML schemas can restrict a value to be no longer than 2 characters, or matching a regular expression, e.g., `[0-9]{5}(-[0-9]{4})?`

XML Schema Definition (XSD)

Example (I)

- A new data type is defined with the `<xsd:complexType>` element

```
<xsd:element name="address">
  <xsd:complexType>
    <xsd:sequence>
      <xsd:element ref="name"/>
      <xsd:element ref="street"/>
      <xsd:element ref="city"/>
      <xsd:element ref="state"/>
      <xsd:element ref="postal-code"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:element>
```

```
<xsd:element name="title" type="xsd:string"/>
<xsd:element name="first-Name" type="xsd:string"/>
<xsd:element name="last-Name" type="xsd:string"/>
<xsd:element name="street" type="xsd:string"/>
<xsd:element name="city" type="xsd:string"/>
```

XML Schema Example (II)

- Derived data type
- Strings restricted by regular expression

```
<xsd:element name="state">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:length value="2"/>
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>

<xsd:element name="postal-code">
  <xsd:simpleType>
    <xsd:restriction base="xsd:string">
      <xsd:pattern value="[0-9]{5}(-[0-9]{4})?" />
    </xsd:restriction>
  </xsd:simpleType>
</xsd:element>
```


XML vs JSON

- XML data can also be encoded in JSON, but JSON does have facility equivalent to DTD and XSD
- JSON is good for client and server maintained by the same developers and the structure/semantic is simple enough for developers to understand and write the code to manipulate the JSON document
 - Server defines a JSON document, which is then shipped to the client
 - Client has the JavaScript codes for processing and displaying the JSON document; OK when the client understands the structure/semantics
 - What if the JSON document is open to other clients? How do these clients understand the format and semantics of the JSON?
 - JSON is ubiquitous among developers
- XML is for enterprise data exchange between heterogeneous environment, DTD/Schema defines the data structure and semantics

Take Home Message

- XML DTD is more **document centric** while XML Schema is more **database centric**
- They both define constraints on XML elements so as to make the data more exchangeable and understandable
- Although you can create any XML that is syntactically correct, you must define the DTD or Schema to make it understandable and sharable to other applications