Week 1

Semistructured Data and XML

Objectives

In this chapter, you will learn:

- What semi-structured data is.
- The main language elements of XML.
- The difference between well-formed and valid XML documents.
- How Document Type Definitions (DTDs) can be used to define the valid syntax of an XML document.

Semi-structured Data

- Data that has some structure, but may not be regular or complete
- Does not conform to a fixed schema (schema-less or self-describing)
- Information about a schema is contained within the data itself

https://app4me.online/easy/WFm03F_vp

https://app4me.online/easy/UtGr0Q1zx

XML

- eXtensible Markup Language (XML)

- A metalanguage (a language for describing other languages) that enables designers to create their own customized tags to provide functionality not available with HTML

- XML is a restricted version of SGML (Standard Generalized Markup Language)

- Provides a similar function to SGML but is less complex.

- XML is designed to complement for HTML by enabling different kinds of data to be exchanged over the web.

SGML

- Is a system for defining structured document types and markup languages to represent instances of those document types

- SGML allows a document to be logically separated into two:

  - The structure of the document called Document Type Definition (DTD)

  - Text

- SGML provides a powerful document management system, but has not been widely adopted due to complexity.

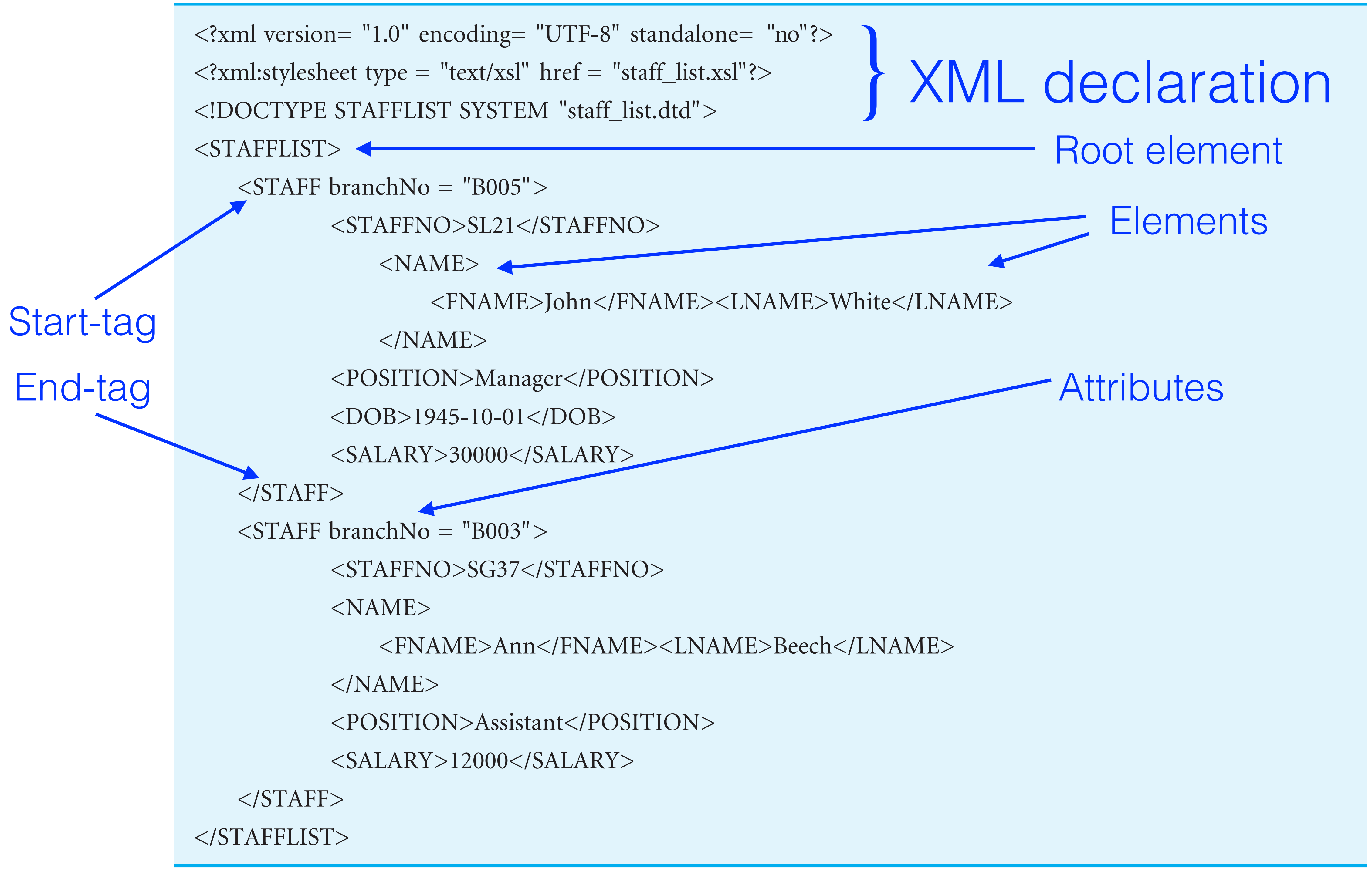
Advantages of XML

- Simplicity
- Open standard and platform/vendor-independent
- Extensibility
- Reuse
- Separation of content and presentation
- Improved load balancing

Advantages of XML

- Support for the integration of data from multiple sources
- Ability to describe data from a wide variety of applications
- More advanced search engines
- New opportunities

XML



```
<?xml version= "1.0" encoding= "UTF-8" standalone= "no"?>
<?xml:stylesheet type = "text/xsl" href = "staff_list.xsl"?>
<!DOCTYPE STAFFLIST SYSTEM "staff_list.dtd">
<STAFFLIST>
    <STAFF branchNo = "B005">
            <STAFFNO>SL21</STAFFNO>
                <NAME>
                    <FNAME>John</FNAME><LNAME>White</LNAME>
                </NAME>
            <POSITION>Manager</POSITION>
            <DOB>1945-10-01</DOB>
            <SALARY>30000</SALARY>
    </STAFF>
    <STAFF branchNo = "B003">
            <STAFFNO>SG37</STAFFNO>
            <NAME>
                <FNAME>Ann</FNAME><LNAME>Beech</LNAME>
            </NAME>
            <POSITION>Assistant</POSITION>
            <SALARY>12000</SALARY>
    </STAFF>
</STAFFLIST>
```

XML Elements

- Elements or tags are most common form of markup.
- First element must be a root element, which can contain other (sub)elements.
- XML document must have only one root element.
- Element begin with start-tag and end with end-tag.
- XML elements are case-sensitive
- An element can be empty and can be abbreviated to <EMPTYELEMENT/>.
- Elements must be properly nested.

XML Attributes

- Attributes are name-value pairs that contain descriptive information about an element
- Attribute is placed inside start-tag after corresponding element name with the attribute value enclosed in quotes.

```
<STAFF branchNo = "B005">
```

XML Other sections

- XML declaration
- Entity references
- Comments <!-- This is a comment -->
- CDATA
- Processing instructions

Entity References

"An alternative name for a series of characters"

- Entities serve three main purposes:
  - as shortcut to often-repeated text or include the content of external files

```
<!ENTITY writer "Donald Duck.">
<!ENTITY copyright "Copyright W3Schools.">

XML example:

<author>&writer;&copyright;</author>
```

  - to insert arbitrary Unicode character into text

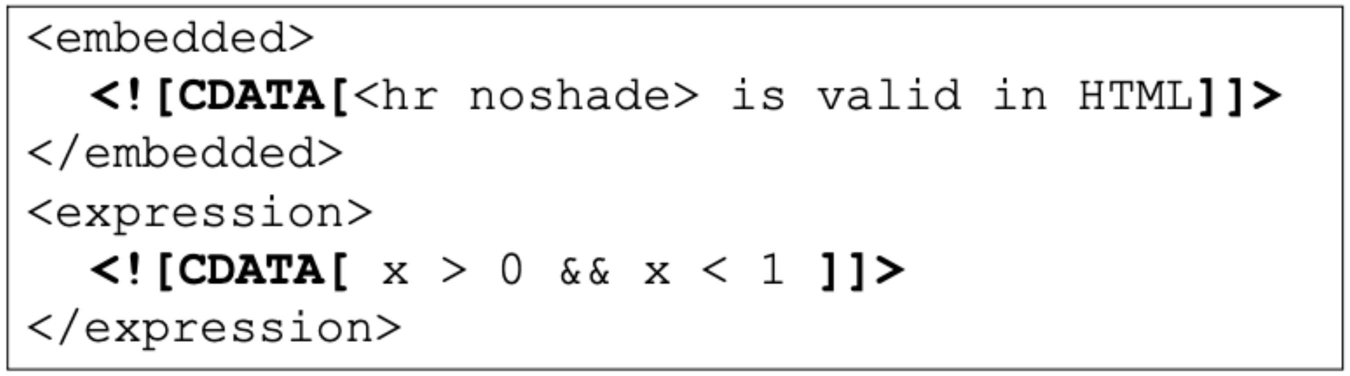
    &#60; represent <

  - to distinguish reserved characters from content

    &lt; represent <

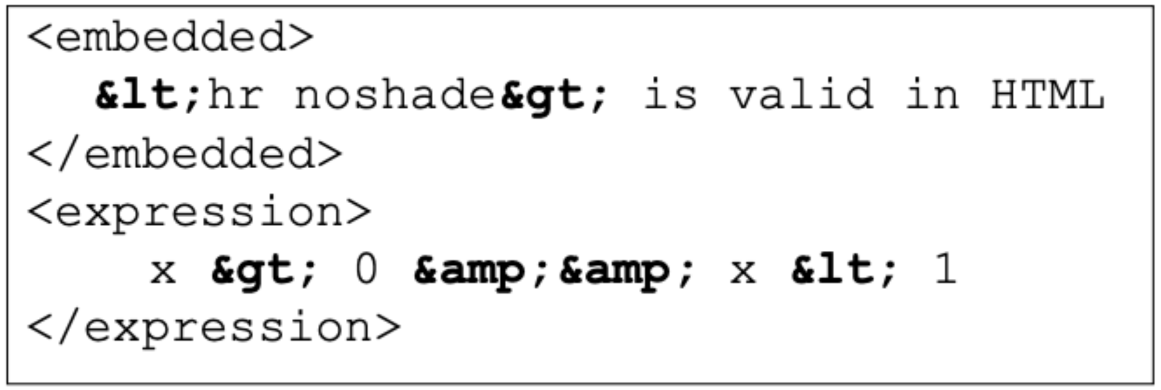
CDATA

- A CDATA section instructs the XML processor to ignore markup characters and pass the enclosed text directly to the application without interpretation.



```
<embedded>
  <![CDATA[<hr noshade> is valid in HTML]]>
</embedded>
<expression>
  <![CDATA[ x > 0 && x < 1 ]]>
</expression>
```

is the same as



```
<embedded>
  &lt;hr noshade&gt; is valid in HTML
</embedded>
<expression>
    x &gt; 0 &amp;&amp; x &lt; 1
</expression>
```

XML ordering

- In Semi-structure data model, collections are unordered.

- In XML, elements are ordered.

```
<NAME>                                  <NAME>
    <FNAME>John</FNAME>                     <LNAME>White</LNAME>
    <LNAME>White</LNAME>                    <FNAME>John</FNAME>
</NAME>                                 </NAME>
```

- However, attributes are unordered.

```
<NAME FNAME = "John" LNAME = "White"/>
<NAME LNAME = "White" FNAME = "John"/>
```

Document Type Definitions (DTDs)

- DTD defines the valid syntax of an XML document

1. List the element names
2. Show how elements can be nested
3. List attributes are available for each element type

- The grammar is specified using EBNF (Extended Backus-Naur Form)
- Although DTD is optional, it is recommended for document conformity

Types of DTD declarations

Four types of DTD declarations:

- Element type declarations
- Attribute list declarations
- Entity declarations
- Notation declarations

DTD - Element type declarations

```
<!ELEMENT STAFFLIST (STAFF)*>
<!ELEMENT STAFF (NAME, POSITION, DOB?, SALARY)>
<!ELEMENT NAME (FNAME, LNAME)>
<!ELEMENT FNAME (#PCDATA)>
<!ELEMENT LNAME (#PCDATA)>
<!ELEMENT POSITION (#PCDATA)>
<!ELEMENT DOB (#PCDATA)>
<!ELEMENT SALARY (#PCDATA)>
<!ATTLIST STAFF branchNo CDATA #IMPLIED>
```

DTD - Element type declarations

<!ELEMENT ... >

- Identify the rules for elements that can occur in the XML document
- The options for repetition are:
  - asterisk (*)   indicates zero or more
  - plus (+).   indicates one or more
  - question mark (?) indicates either zero or exactly one
- Name with no qualifying punctuation must occur exactly once.
- Commas between element names indicate they must occur in succession. If commas omitted, elements can occur in any order.
- #PCDATA   indicates parsable character data.
- CDATA   indicates character data, containing any text. The string will not be parsed by   the XML processor and simply passed directly to the application.

DTD - Attribute list declarations

<!ATTLIST ... >

- Identify:
  - which elements may have attributes
  - what attributes they may have
  - what values attributes may hold
  - optional defaults
- There are some possible attribute types:
  - CDATA character data, contain any text
  - ID.   used to identify individual elements in a document
  - IDREF/ IDREFS
  - List of names   the values that attribute can hold (enumerated type)

Example : Attribute list declaration

Syntax:

```
<!ATTLIST elm-name  att-name  att-type  default-decl >
```

```
<!ATTLIST STAFF branchNo  CDATA #IMPLIED>
```

- The branchNo value is a string (CDATA) and is optional (#IMPLIED or #REQUIRED)

```
<!ATTLIST SEX   gender (M | F )  "M">
```

- The SEX element has an attribute gender containing either the value M or F and must always have the default value M.

DTD - Entity and notation declaration

- Entity declarations associate a name with some fragment of content, such as a piece of regular text, a piece of the DTD, or a reference to an external file containing text or binary data.

```
<!ENTITY DH "DreamHome Estate Agents">
```

- Notation declarations identify external binary data, which is simply passed by the XML processor to the application.

```
<!ENTITY  dreamHomeLogo  SYSTEM "dremhome.jpg"  NDATA  JPEGFormat>

<!NOTATION JPEGFormat   SYSTEM. "http://www.jpeg.org">
```

DTD - Element identify, IDs and ID references

- ID   allows a unique key to be associated with an element
- IDREF allows an element to refer to another element with the designated key
- IDREFS allows an element to refer to multiple elements

```
<STAFF staffNo = "SL21">
    <NAME>
        <FNAME>John</FNAME><LNAME>White</LNAME>
    </NAME>
</STAFF>
<STAFF staffNo = "SL41">
    <NAME>
        <FNAME>Julie</FNAME><LNAME>Lee</LNAME>
    </NAME>
</STAFF>
<BRANCH staff = "SL21 SL41">
    <BRANCHNO>B005</BRANCHNO>
</BRANCH>
```

```
<!ATTLIST STAFF staffNo ID #REQUIRED>
<!ATTLIST BRANCH staff IDREFS #IMPLIED>
```

DTDs - Document Validity

- Two levels of document processing
  - Well-formed
  - Valid
- XML document that conforms to structural and notational rules of XML is considered **well-formed**
  - XML document starts with <?xml version “1.0”?>
  - All elements must be within one root element
  - Elements must be nested in a tree structure without any overlap
  - All non-empty elements must have a start-tag and an end-tag

DTDs - Document Validity

- Non-validating processor ensures an XML document is **well-formed** before passing information on to application.

- A validating processor will not only check that an XML document is well-formed but that is also conforms to a DTD, in which case XML document is considered **valid.**