XML: eXtensible Markup Language

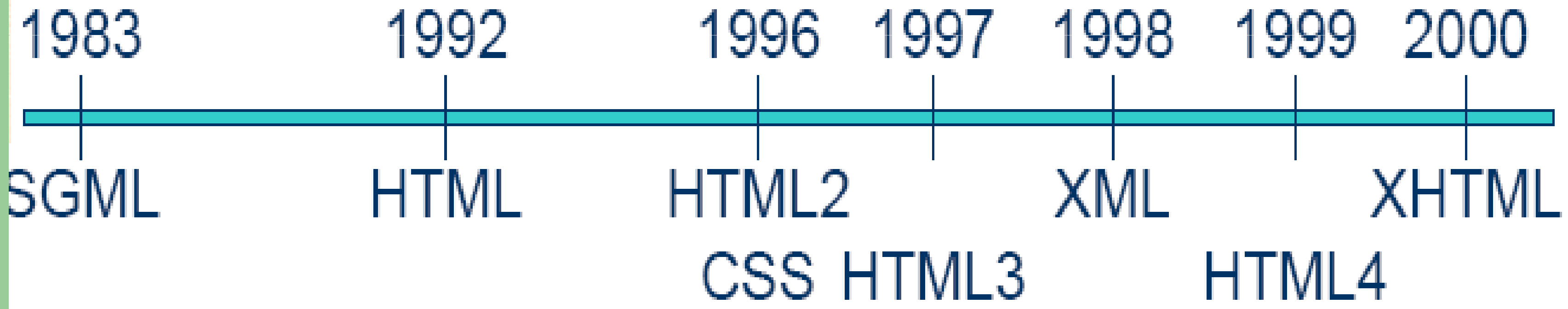
Anabel Fraga

Table of Contents

Historic Introduction (I)

- **XML was defined as an W3C standard in 1998. In 2000 version 1.0 was approved.**
- It is a **tagged language**, such as HTML or its precursor SGML is.
- It differentiates to SGML for its **simplicity**
- It differentiates to HTML for its **flexibility**: the number of tags that can be included in a XML document is unlimited.
- Equally to HTML, it is **portable** to any platform.

Historic Introduction (II)



Historic Introduction (III)

- Main objectives:
  - **Directly utilizable in Internet**
  - **Support for a wide variety of application for data transfer**
  - **Compatible with SGML**
  - Possible to create simple XML processors
  - Readable XML Documents and relatively easy to understand (depending on the definition)
  - Rapid language design
  - Simple, but perfectly formal
  - Easy to create XML Documents

XML vs. HTML

- HTML lacks a syntactic checker. **Pages with errors are displayed in the browsers**
- HTML **lacks a structure**
- HTML is not object oriented
- HTML mixes content and representation
- For all this:
  - **HTML can not be easily read by a machine**
  - **HTML will never be a standard for data interchange**
- XML covers all these with a language of extreme simplicity

XML Characteristics (I)

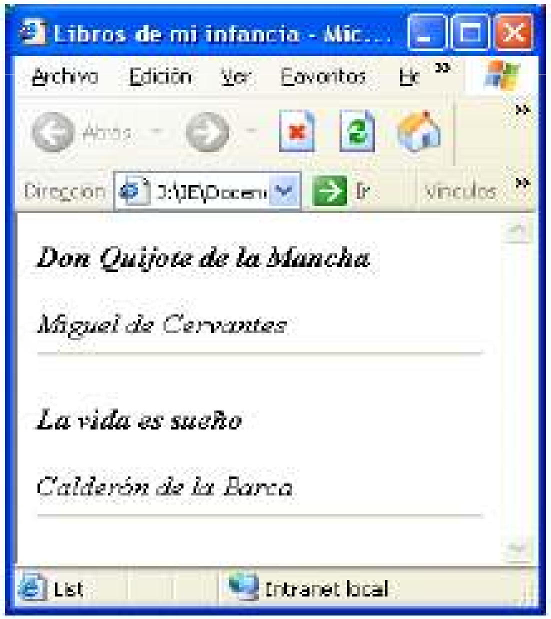
- It is a subset of the SGML language
- Similarly to SGML, it is used for representing data in a **structured** form (Hierarchical)
- It is based on an **obligatory and well defined grammar**. This facilitates the development of *parsers* and thus, its massive utilization
- The internal structure of an XML document is reflected in another document called **DTD (*Document Type Definition*)**
- In contrast to HTML, it drastically separates the **semantic** of the document, from its graphical representation

XML Characteristics (II)

- XML has been converted to a **standard for data interchange** not only for the Web
- It is easy to use, for both humans and machines, because it is based on a set of extensible semantic tags.
- **It is now in a state of maturity and absolute expansion**
- Thanks to its support of *Unicode*, all alphabets of the world are supported

HTML Document (I)

```
<HTML>
<HEAD><TITLE>Libros de mi
    infancia</TITLE></HEAD>
<BODY>
<P><I><B>Don Quijote de la
    Mancha</B>
<P><I>Miguel de Cervantes</I>
<HR>
<P><B>La vida es sueño</B>
<P><I>Calderón de la Barca</I>
<HR>
</BODY>
</HTML>
```



Don Quijote de la Mancha

Miguel de Cervantes

La vida es sueño

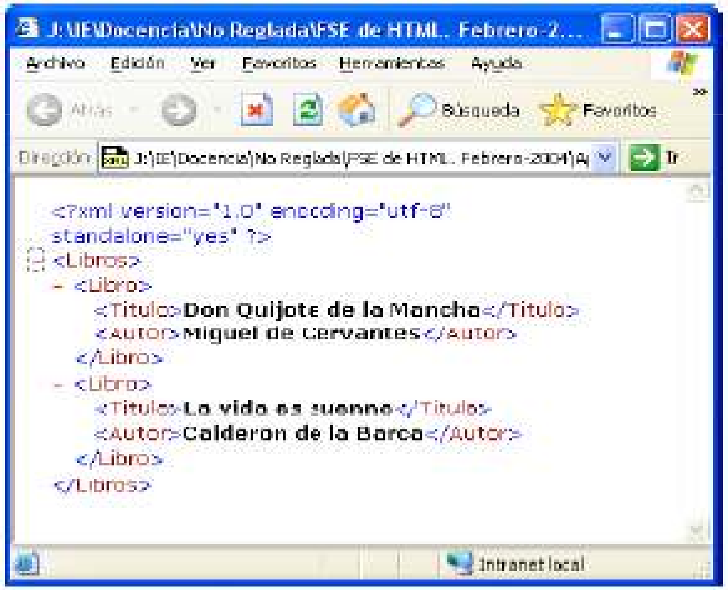
Calderón de la Barca

HTML Document (II)

- As it appears, the previous HTML document is correct, but:
  - There are tags that never close: <P>
  - Some tags are not well nested: the first <I> never closes
  - For a non human reader, it is not clear which is the book and who is the author
- XML eradicates all those problems!!

XML Document

```
<?xml version="1.0" encoding="utf-8"
    standalone="yes" ?>
<Libros>
    <Libro>
            <Titulo>Don Quijote de la
    Mancha</Titulo>
            <Autor>Miguel de
    Cervantes</Autor>
    </Libro>
    <Libro>
            <Titulo>La vida es suenno</Titulo>
            <Autor>Calderon de la
    Barca</Autor>
    </Libro>
</Libros>
```


J:\IE\Docencia\No Reglada\FSE de HTML. Febrero-2...
Archivo
Edición
Ver
Favoritos
Herramientas
Ayuda
Atrás
Búsqueda
Favoritos
Dirección
J:\IE\Docencia\No Reglada\FSE de HTML. Febrero-2004\A
Ir
Intranet local

```
<?xml version="1.0" encoding="utf-8"
standalone="yes" ?>
<Libros>
- <Libro>
    <Titulo>Don Quijote de la Mancha</Titulo>
    <Autor>Miguel de Cervantes</Autor>
  </Libro>
- <Libro>
    <Titulo>La vida es suenno</Titulo>
    <Autor>Calderon de la Barca</Autor>
  </Libro>
</Libros>
```

General Rules for XML

- One **unique** root element
- All elements need to have **opening and closing** tags
- Distinction between upper/lower case letters (**Case Sensitive**)
- Perfect **nesting** amongst elements
- The **values** of **attributes** are always placed between double quotes (“example”)
- White spaces are preserved
- CR/LF characters are transformed to LF

Well Formed and Valid Documents

- A documents is thought to be well formed when:
  - It complies with all the rules previously defined
  - Contains one or more elements
  - It has only one root element (document element)
  - If the document constitutes of more than one parts, all must be well formed
  - There are no prohibited characters in the text
- A document is valid when, except being ‘well formed’, it complies with the semantic specification defined in its definition (DTD o XML Schema)

Elements (I)

- Comments:
  - <!– This is a comment, and we can not include a double dash -->
- Processing instructions:
  - <? Instruction ?>
  - The instruction can not include characters ?>
- CDATA Sections (Character Data):
  - <![CDATA[This text will not be treated and can include "any" &character < >]]>
  - Are not treated by the *parser*
  - They can include any prohibited character (", ', &, >, <).
  - They can NOT include the character sequence]]>

Elements (II)

- Prologue:

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

  – It is an **compulsory** processing instruction
  – **Version**: indicates the XML version used (1.0 in this case). *Its* **compulsory**
  – **Encoding**: indicated the document encoding, and ***is NOT compulsory*** (default UTF-8). Valid for other character sets
  – **Standalone**: “yes” indicates that the document is not accompanied with external DTDs; “no” indicates that it has a DTD. ***Not a compulsory attribute***

Elements (III)

- DOCTYPE: **<!DOCTYPE MyDTD SYSTEM "C:\MyDTD.dtd">**
  - Indicates a reference (URI) to a DTD, in this case same to the name (MyDTD) of its root element
  - The DTD could be incorporated in the same XML document, without the need of a separate file
  - The XML document has to comply with the DTD content

Elements (IV)

- Tags:
  - Must be correctly nested: opening and closing
  - Opening tag: starts with <, the name of the tag and finishes with >. Example <Book>
  - Closing tag: </Book>
  - Empty tag: <Book />
  - You can not start a tag name with ".", ":", "-", numbers
  - After the 1st character we can put ".", numbers, "-"
  - Tag names must start with a letter or with an underscore "_"
  - Tag names can not start with "xml"

Elements (V)

- Element:
  - Is the set of an opening tag, its content, and its closing tag
  - For example: <Book>Don Quijote de la Mancha</Book>
  - There are some reserved characters (prohibited):

    □□ Greater sign: >

    □□ Smaller sign: <

    □□ Ampersand: &

    □□ Single quote: ‘

    □□ Double quotes: “
  - These prohibited characters are replaced with entities or are included in CDATA sessions

Elements (VI)

- Attributes:
  - Every element can contain 0 or more attributes
  - Its value has to be always between double quotes. (“value”)
  - They can only be placed into opening or empty tags
  - The same attribute can not be repeated in the same tag
  - If the document has a DTD, every attribute must be defined as an attribute for that element
  - Can not contain any reference to an external reference
  - Are always treated as sequences of text

Elements (VII)

```
<Book>
<Title>Don Quijote de la Mancha</Title>
<Author>Miguel de Cervantes</Author>   (Without attributes)
<Price> 21,95 euros </Price>
<Publisher> Santillana </ Publisher >
</Book>

<Book Price = “21,95 euros" Publisher = "Santillana">
<Title>Don Quijote de la Mancha</Title>
<Author>Miguel de Cervantes</Author>
</Book>                    (One element has two attributes)
```

Elements vs Attributes

http://www.w3schools.com/dtd/dtd_el_vs_attr.asp

http://www.ibm.com/developerworks/xml/library/x-eleatt.html

http://w3future.com/html/stories/elemvsattrs.xml

Exercise

- Make an XML document based on the text given in the class

DTDs (I)

- Document Type Definition
- Defines the grammar to be followed in the XML document in order to be considered as valid.
- It can be included in an external file:

  <!DOCTYPE root-element SYSTEM “DTD_File.dtd">

  and/or in the same XML file:

  <!DOCTYPE root-element [element-declarations]>

DTDs (II) (Types Declaration)

```
<!DOCTYPE Books SYSTEM
    "Books1.dtd">
<Books>
    <Book>
        <Title>Don Quijote de
    la Mancha</Title>
        <Author>Miguel de
    Cervantes</Author>
    </Book>
    <Book>
        <Title>La vida es
    sueno</Title>
        <Author>Calderon de
    la Barca</Author>
    </Book>
</Books>
```

```
<!DOCTYPE Books [
<!ELEMENT Books (Book)+>
<!ELEMENT Book (Title, Author)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
]>
<Books>
    <Book>
        <Title>Don Quijote de la Mancha</Title>
        <Author>Miguel de Cervantes</Author>
    </Book>
    <Book>
        <Title>La vida es suenno</Title>
        <Author>Calderon de la Barca</Author>
    </Book>
</Books>
```

DTDs (III)

- All the DTD must have **one and only one root element** (also known as **document element**)
- This **root document** must **coincide** with the name that appears after the DOCTYPE
- A DTD document can contain:
  - Element declarations
  - Attribute declarations for an element
  - Entities declarations (&nbsp; or &lt;)
  - Notations declarations
  - Processing instructions
  - Comments
  - References to parameter entities

DTDs (IV) (Root Element)

- After the root element, we can optionally list (in hierarchical form) other elements

```
<!ELEMENT Books (Book)+>
<!ELEMENT Book (Title, Author)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
```

DTDs (V) (Element Contents)

- Contents of an element:
  - **EMPTY**: the element is empty (it can contain attributes).

    <!ELEMENT IMAGEN EMPTY>
  - **ANY**: an element can contain any other element including textual content.

    <!ELEMENT IMAGE ANY>
  - **Other elements**: an element can contain one or more child elements in a certain sequence (E.g. Book)
  - **#PCDATA**: parsed character data.

    <!ELEMENT BOOK (#PCDATA)>
  - **#CDATA**: character data. *(Not parsed by parser)*

    <!ELEMENT BOOK (#CDATA)>
  - **Mixed**: the element can include character sequences optionally mixed with child elements.

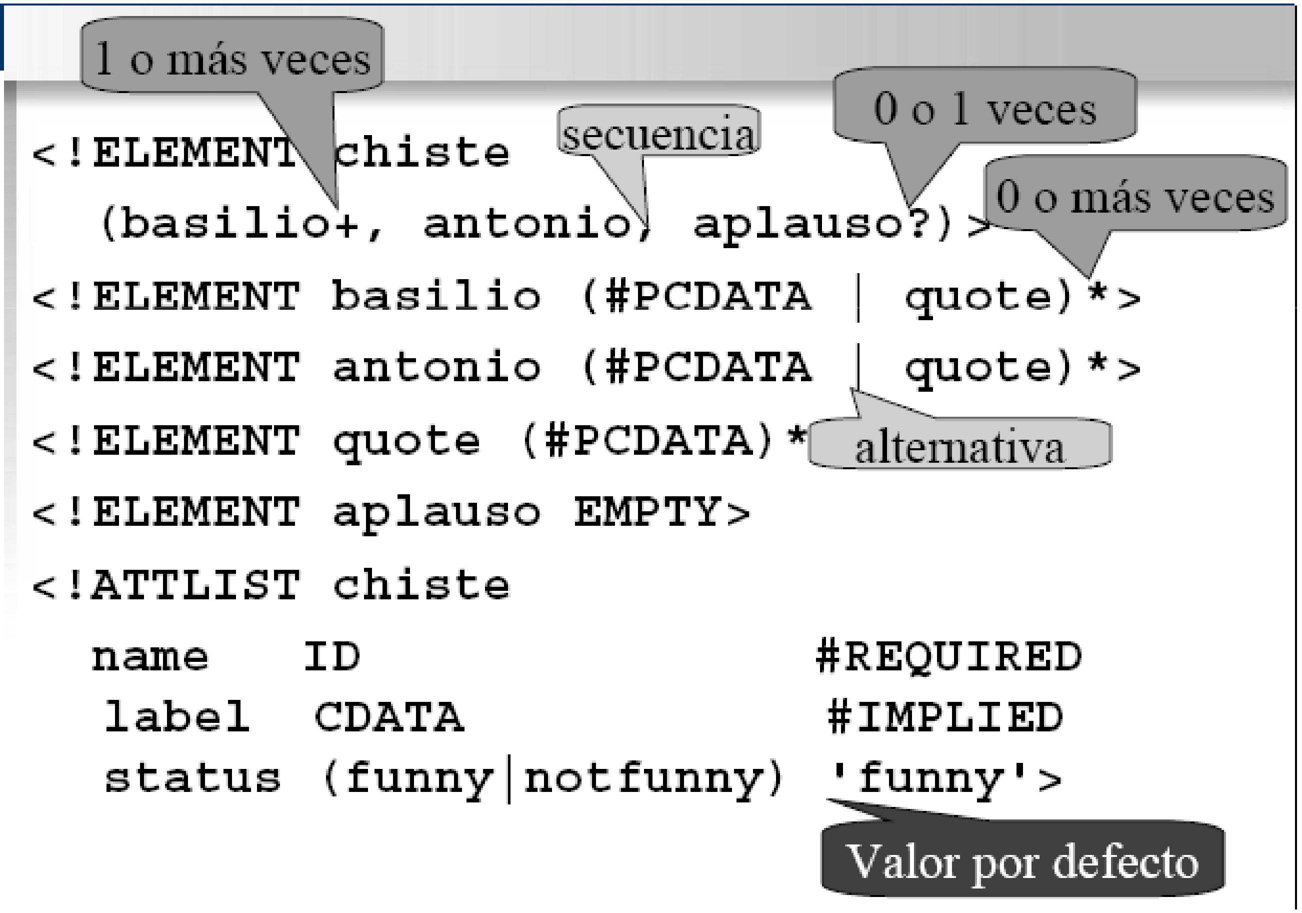
    <!ELEMENT BOOK (#PCDATA | AUTOR)*>

DTDs (VI)

- Sequences of child elements:
  - Sequence:
    - Ordered sequence: comma separated children
    - Options: Pipe (|) separated children functioning as OR
    - Groups of elements can be grouped inside parenthesis
  - Cardinality: one element, or a group of elements may be repeated 0, 1 or more times:

element	**Element repeated 1 single time**
?	**Element repeated 0 or 1 times**
*****	**Element repeated 0 or more times**
+	**Element repeated 1 or more times**

DTDs (VII)



```
<!ELEMENT chiste
  (basilio+, antonio, aplauso?)>
<!ELEMENT basilio (#PCDATA | quote)*>
<!ELEMENT antonio (#PCDATA | quote)*>
<!ELEMENT quote (#PCDATA)*>
<!ELEMENT aplauso EMPTY>
<!ATTLIST chiste
   name   ID                     #REQUIRED
   label  CDATA                  #IMPLIED
   status (funny|notfunny) 'funny'>
```

DTDs (VIII) (Example)

```
<!ELEMENT BOOK (Author, Publisher)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT FILM (Actor|Actress|Director)+>
<!ELEMENT FILM ((Actor | Actress)*, Director,
   Makeup?)>
<!ELEMENT FILM (#PCDATA | Actor)*>
<!ELEMENT FILM (Title, Category, (Actor |
   Actress | Narrator)*)>
```

DTDs (IX)

Exercise: Make a DTD.

```
<?xml version="1.0" encoding="utf-8" ?>
<Agenda>
<Person>
     <Name> Anabel </Name>
     <Surname> FRaga </Surname>
     <Email> afraga@inf.uc3m.es </Email>
     <Office>  2.1 B18 </Office>
     <Telephone> 5555555 </Telephone >
     <Mobile> 5557777 </Mobile>
</Person>
</Agenda>
```

DTDs (X) (Attributes)

- An element can optionally declare one or more attributes

  <!ATTLIST element-name attribute-name attribute-type Modifier>

- The attribute of an element can be included in one or more declarations <!ATTLIST ...>. If it is done in the same declaration, it can be separated with a space (space, tab, carriage return)

DTDs (XI) (Attribute Types)

- Type of an attribute:
  - Sequence type: CDATA (Character Data)

    <!ATTLIST Author Nationality CDATA>
  - Enumerated type:

    <!ATTLIST Film Category (Fiction | Terror | Humor)>
  - Symbolic type:
    - **ID**: will be a unique identifier for the rest of the document, only one ID attribute for each element
    - **IDREF**, **IDREFS**: its value has to coincide with another value of type ID in the rest of the XML document. IDREFS separates the references with space “ID1 ID2 ID3”
    - **ENTITY**, **ENTITIES**: its value has to coincide with one or more entities (alias to large bit of text)
    - **NMTOKEN**, **NMTOKENS**: its value has to be a sequence of type *token*

DTDs (XII) (Attribute Types)

Type	Description
CDATA	The value is character data
(en1|en2|..)	The value must be one from an enumerated list
ID	The value is a unique id
IDREF	The value is the id of another element
IDREFS	The value is a list of other ids
NMTOKEN	The value is a valid XML name
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities
NOTATION	The value is a name of a notation
xml:	The value is a predefined xml value

DTDs (XIII) (Attribute Modifiers)

- Modifiers:
  - **#REQUIRED**: this attribute has to be introduced compulsorily.

    <!ATTLIST Film Title CDATA #REQUIRED>
  - **#IMPLIED**: indicates that this attribute is ***optional***
  - **PredefinedValue**: if the attribute is omitted, the processors use this value as default

    <!ATTLIST Film Category (Fiction | Terror | Humor) "Humor">

    <!ATTLIST Author Nationality CDATA "Spaniard">
  - **#FIXED**: if the attribute is included, the processors will always use this value

    <!ATTLIST Author Nationality CDATA #FIXED "Spaniard">

DTDs (XIV) (Attribute Modifiers)

Value	Explanation
value	The default value of the attribute
#REQUIRED	The attribute is required
#IMPLIED	The attribute is not required
#FIXED value	The attribute value is fixed

DTDs (XV) (Entities)

Entities are variables used to define shortcuts to standard text or special characters.

- Entity references are references to entities
- Entities can be declared internal or external

Syntax

<!ENTITY entity-name "entity-value">	Internal Entity Declaration
<!ENTITY entity-name SYSTEM "URI/URL">	External Entity Declaration

DTD Example:

```
<!ENTITY writer "Donald Duck.">
<!ENTITY copyright "Copyright W3Schools.">
<!ENTITY writer SYSTEM "http://www.w3schools.com/entities.dtd">
<!ENTITY copyright SYSTEM "http://www.w3schools.com/entities.dtd">
```

XML example:

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

DTDs (XVI) (Attributes Exercise)

- Make a DTD using attributes: (INCLUDE THIS IN YOUR ASSIGNMENTS REPORT)

```
<?xml version="1.0" encoding="utf-8" ?>
<Agenda>
<Person>
    <Name> Anabel </Name>
    <Surname> Fraga </Surname>
    <DNI> 44444444-O </DNI>
    <Nationality> Spanish </Nationality>
    <Email> afraga@inf.uc3m.es </Email>
    <Office>  2.1.B18 </Office>
    <Telephone> 5555555 </Telephone >
    <Mobile> 5557777 </Mobile>
</Person>
</Agenda>
```

DTDs (XVII) (Problems)

- DTD does not follow the format of a **standard** XML document**.** This represents a **problem** for the ***parsers***
- **Distinct types of data** is not supported in the style of programming languages (CDATA, #PCDATA)
- You can not create personalized data types
- Namespaces are not supported
- The number of elements occurrences can not be 100% controlled (E.g. min 2 occurrences)
- **For these and other reasons, XML *Schemas* have emerged**

Namespaces (I)

- XML permits the creation of tags with ‘almost’ no limitation in their names
- This implicates that, mixing two documents, with different tags, could result to a duplicity of tags
- Through namespace definition, these collisions can be avoided
- Technologies like XSL and many others make use of *Namespaces*

Namespaces (II) (Definition)

- A namespace is identified by its prefix.

For example:

<pref:elementName
  xmlns:pref="http://www.w3.org/XSL/Transform/1.0">

where:

– **pref** is the *namespace* prefix

– **elemenName** is the complete name of the *element*

– **http://www**... the address used to identify the namespace is not used by the parser to look up information. The only purpose is to give the namespace a unique name.

– Other attributes like ***version*** may be included...

Namespaces (III)

This XML document carries information in a table:

```
<table>
      <tr>
                <td>Apples</td><td>Bananas</td>
      </tr>
</table>
```

This XML document carries information about a table (a piece of furniture):

```
<table>
      <name>African Coffee Table</name>
      <width>80</width>
      <length>120</length>
</table>
```

If these two XML documents were added together, there would be an element name conflict because **both documents contain a <table>** element with different content and definition.

Namespaces (IV)

This XML document carries information in a table:

```
<h:table xmlns:h="http://www.w3.org/TR/html4/" >
      <h:tr>
                <h:td>Apples</h:td><td>Bananas</h:td>
      </h:tr>
</h:table>
```

This XML document carries information about a table (a piece of furniture):

```
<f:table xmlns:f="http://www.w3schools.com/furniture" >
      <f:name>African Coffee Table</f:name>
      <f:width>80</f:width>
      <f:length>120</f:length>
</f:table>
```

By using a prefix, we have created two different types of <table> elements. We have added an xmlns attribute to the <table> tag to give the prefix a qualified name associated with a namespace.

Namespaces

One more illustrative example at :

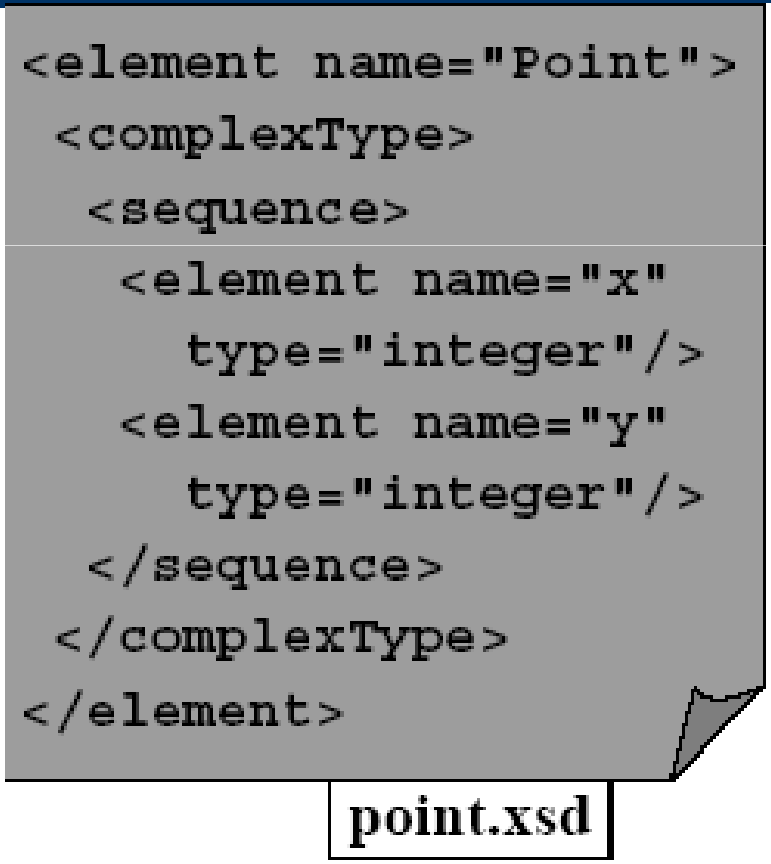
http://www.xml.com/pub/a/1999/01/namespaces.html

XML Schemas (I)

- Currently exists a new W3C recommendation of May 2001 for XML definitions:

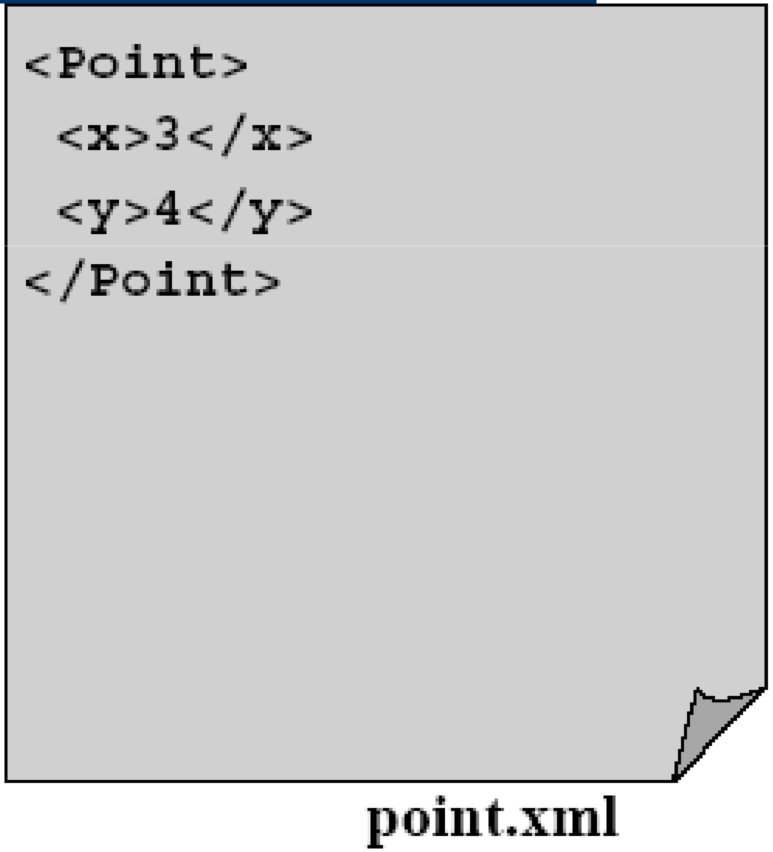
  XML Schemas
- XML Schema is an XML-based alternative to DTDs.
- An XML Schema describes the structure of an XML document.
- The XML Schema language is also referred to as XML Schema Definition (XSD).
- Limited usage: Currently there is a great quantity of documents defined with DTDs.

XML Schemas (II) (Example)



```
<element name="Point">
 <complexType>
  <sequence>
   <element name="x"
     type="integer"/>
   <element name="y"
     type="integer"/>
  </sequence>
 </complexType>
</element>
```

point.xsd



```
<Point>
 <x>3</x>
 <y>4</y>
</Point>
```

point.xml

XML Schemas vs. DTDs (III)

DTDs Disadvantages

- You don’t write in XML syntax
- Small usage of namespaces
- Few data types (and what’s worst, can not define new types)
- Even though you can group elements between entities (%;) they are a little developed

DTDs Advantages

- Supported by many tools
- Many documents already exist: DTDs y XMLs based on those
- Easy to learn

XML Schemas vs. DTDs (IV)

- Advantages:
  - They permit multiple data types (e.g. xs:date, xs:int, xs:language, ...)
  - Ample use of namespaces
  - Permits the grouping of elements for its reutilization, permits inheritance (e.g.: Personal Data in distinct Domains)

- READ this on XML Schemas:

  http://www.w3schools.com/schema/schema_intro.asp

The XML Family (I)

- **XPointer/XLink**: permit referencing to other resources, within or outside the XML document
- **XPath**: Query language for parsing and searching XML files
- **XQL** (XML Query Language): useful for locating and extracting elements from an XML document
- **XIRQL**: An XQL extension for Information Retrieval
- **XSLT:** Language for transforming XML documents

The XML Triumph

- Structure and content separated
- Data has to be interchanged through the net:
  - Tree structured documents are in a portable format useful for everything
  - XML is used as a data interchange mechanism

XML Utilization Domains

- Data interchange for medicines
- Handling of mathematical information (XMath)
- Interchange of information between executable programs (SOAP)
- Interchange of information between tools CASE (XMI)
- Interchange of information over Human Resources (XML-HR)
- Interchange of information over the stock exchange and finance (IFX)
- Ample utilization in the EDI sector (*Electronic Data Exchange)*
- Electronic Commerce (ECML, eCo, ebXML, xml-edifact)
- ‘Web’ Standards like WML y XHTML

XML according to W3C

XML is a method for putting structured data in a text file
XML looks a bit like HTML but isn't HTML
XML is text, but isn't meant to be read
XML is a family of technologies
XML is verbose, but that is not a problem
XML is new, but not that new
XML is license-free, platform-independent and well-supported

-- Bert Bos, W3C

References

TUTORIALS

- http://www.xml.org
- http://www.florida-uni.es/~fesabid98/Comunicaciones/f_santamaria/f_santamaria.htm
- http://www.stud.ifi.uio.no/~lmariusg/download/xml/xml_eng.html
- http://www.it.uc3m.es/entry/index.html
- http://www.xml.com/pub/a/98/08/xmlqna0.html
- http://www.dat.etsit.upm.es/~abarbero/curso/xml/xmltutorial.html
- http://www.hypermedic.com/style/xml/xmltut.txt
- http://aries17.uwaterloo.ca/tutorial/xml/

RESOURCES

- http://www.programacion.net/xml.htm
- http://www.hypermedic.com/style/xml/xmlindex.htm
- http://slug.ctv.es/~olea/sgml-esp/recursos.html
- http://www.xmlspy.com

VARIOUS

- http://slug.ctv.es/~olea/sgml-esp/
- http://aries17.uwaterloo.ca/tutorial/xml/
- http://www.epsilon-eridani.com/PHPdoc/EEdoc.php3
- http://slug.ctv.es/~olea/
- http://www.centurycomputing.com/ng-html/xml/xml-syntax.html
- http://www.hypermedic.com/style/xml/xmlindex.htm
- http://www.ramon.org/index2.htm
- http://www.haifa.il.ibm.com/sigir00-xml/final-papers/KaiGross/sigir00.html
- www.w3schools.com/xpath/
- www.w3schools.com/xpath/tryit.asp?filename=try_xpath_select_cdnodes
- www.zvon.org/xxl/XPathTutorial/General/examples.html
- ftp://www6.software.ibm.com/software/developer/library/mcolan/
- ibm.com/developerworks/speakers/colan