Internet Technologies
11-XML

F. Ricci

2010/2011

Content

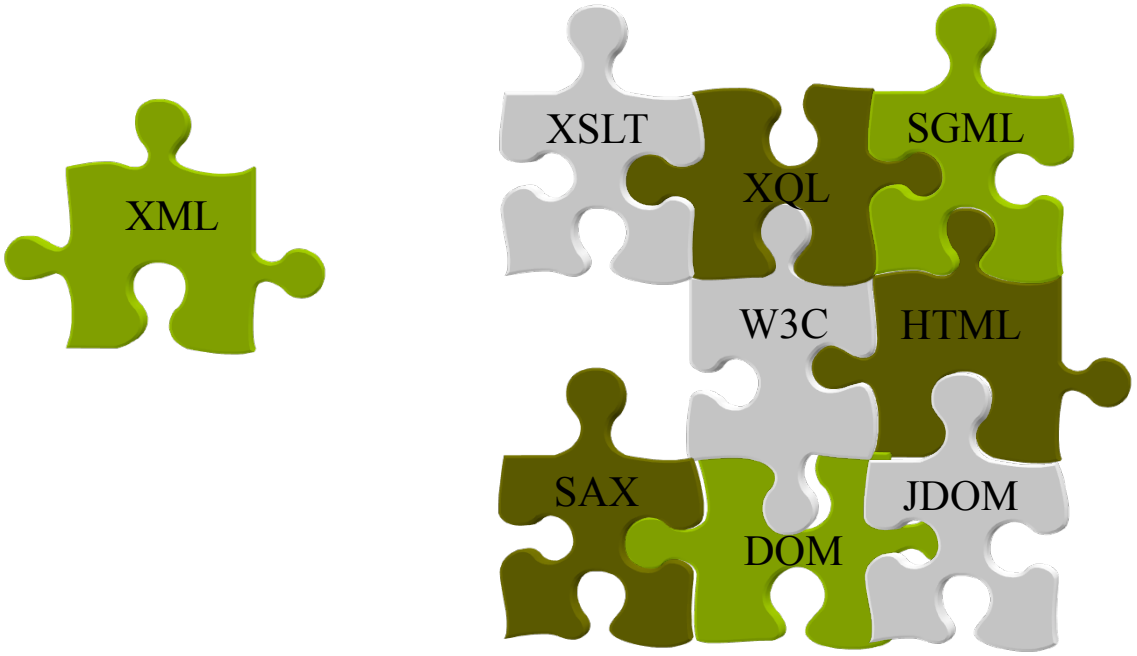
- Motivation
- Examples
- Applications
- Elements, Prologue, Document Type Definition
- Attributes
- Well-formedness
- XML and CSS
- Document Type Definition and validation
- Schemas
- Namespaces

What is XML

- XML stands for E**X**tensible **M**arkup **L**anguage
- XML is a **markup language** much like HTML
- XML was designed to **describe data**
- XML tags are not predefined - you must **define your own tags**
- XML uses a **Document Type Definition** (DTD) or an **XML Schema** to describe the data
- XML with a DTD or XML Schema is designed to be **self-descriptive**
- XML is a W3C Recommendation.

XML Overview

- When people refer to XML, they typically are referring to XML and related technologies



Markup

Information added to data to enhance its meaning

- Identification of parts, boundaries and relationships of elements within documents
- Identification of attributes

Without markup, most documents appear as meaningful to machines as this document does to humans

"Java + XML =

portable programs + portable data"

What's wrong with HTML?

```
<html>
 <head>
  <title>
   Martin's page
  </title>
 </head>
 <body bgcolour="#ffffff">
  <p align="center">
   Some text or other
  </p>
 </body>
</html>
```

- HTML is the most successful electronic publishing language ever invented, but ....
- .... HTML is for *presentation*, not content, limiting its applicability
- Actually is for **content of web pages** – as presentation is supported by CSS.

XML example

```
<?xml version="1.0" encoding="UTF-8"?>
<Curriculum>
  <Course Title="Z" Lect="Bogdanov">
      <Lecture>Props</Lecture>
      <Lecture>Predicates</Lecture>
      <Lecture>Sets</Lecture>
   </Course>
</Curriculum>
```

XML example

```
<?xml version="1.0" encoding="UTF-8"?>
<Curriculum>
   <Course Title="Z" Lect="Bogdanov">
      <Lecture>Props</Lecture>
      <Lecture>Predicates</Lecture>
      <Lecture>Sets</Lecture>
   </Course>
</Curriculum>
```

Encodes

- boundaries

XML example

```
<?xml version="1.0" encoding="UTF-8"?>
<Curriculum>
   <Course Title="Z" Lect="Bogdanov">
      <Lecture>Props</Lecture>
      <Lecture>Predicates</Lecture>
      <Lecture>Sets</Lecture>
   </Course>
</Curriculum>
```

Encodes

- boundaries
- roles

  e.g. course vs lecture

XML example

```
<?xml version="1.0"
   encoding="UTF-8"?>
<Curriculum>
   <Course Title="Z" Lect="Bogdanov">
      <Lecture>Props</Lecture>
      <Lecture>Predicates</Lecture>
      <Lecture>Sets</Lecture>
   </Course>
</Curriculum>
```

Encodes

- boundaries
- roles

  eg course vs lecture
- positions

XML example

```
<?xml version="1.0"
   encoding="UTF-8"?>
<Curriculum>
   <Course Title="Z"
   Lect="Bogdanov">
      <Lecture>Props</Lecture>
      <Lecture>Predicates</Lecture>
      <Lecture>Sets</Lecture>
   </Course>
</Curriculum>
```

Encodes

- boundaries
- roles

  eg course vs lecture
- positions
- containment

  eg lecture is part of course

XML example

```
<?xml version="1.0" encoding="UTF-8"?>
<Curriculum>
   <Course Title="Z" lect="Bogdanov">
      <Lecture>Props</Lecture>
      <Lecture>Predicates</Lecture>
      <Lecture>Sets</Lecture>
   </Course>
</Curriculum>
```

Encodes

- boundaries
- roles

  eg course vs lecture
- positions
- containment

  eg lecture is part of course
- attributes

  eg title

A brief history of markup

GML: Generalised Markup Language

- Developed in 60's and 70's by IBM
- Used for IBM technical manuals

SGML: Standardised GML

- 70's, 80's with ANSI standard in 1983
- Flexible and very general, but difficult and costly

HTML

- Early 90's: compact markup for hypertext docs
- Now seen as a step backwards

XML

- XML 1.0 became a W3C Recommendation on 1998
- XML 1.1, was initially published on 2004 (differ in the requirements of characters used for element and attribute names).

Extensible

- XML documents can be extended to carry more information

```
<note>
    <to>Tove</to>
    <from>Jani</from>
    <body>Don't forget me this weekend!</body>
</note>
```

- If the author of the XML document added some extra information to it

```
<note>
    <date>2002-08-01</date>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
</note>
```

- An existing application (using the first version) should still be able to find the <to>, <from>, and <body> elements in the XML document and produce the same output as before.

Applications of XML

- Configuration files
  - Used extensively in J2EE architectures
- Media for data interchange
  - A better alternative to proprietary data formats
- B2B transactions on the Web
  - Electronic business orders (ebXML)
  - Financial Exchange (IFX)
  - Messaging exchange (SOAP)
- Store data
  - With XML, plain text files can be used to store data (XML databases).

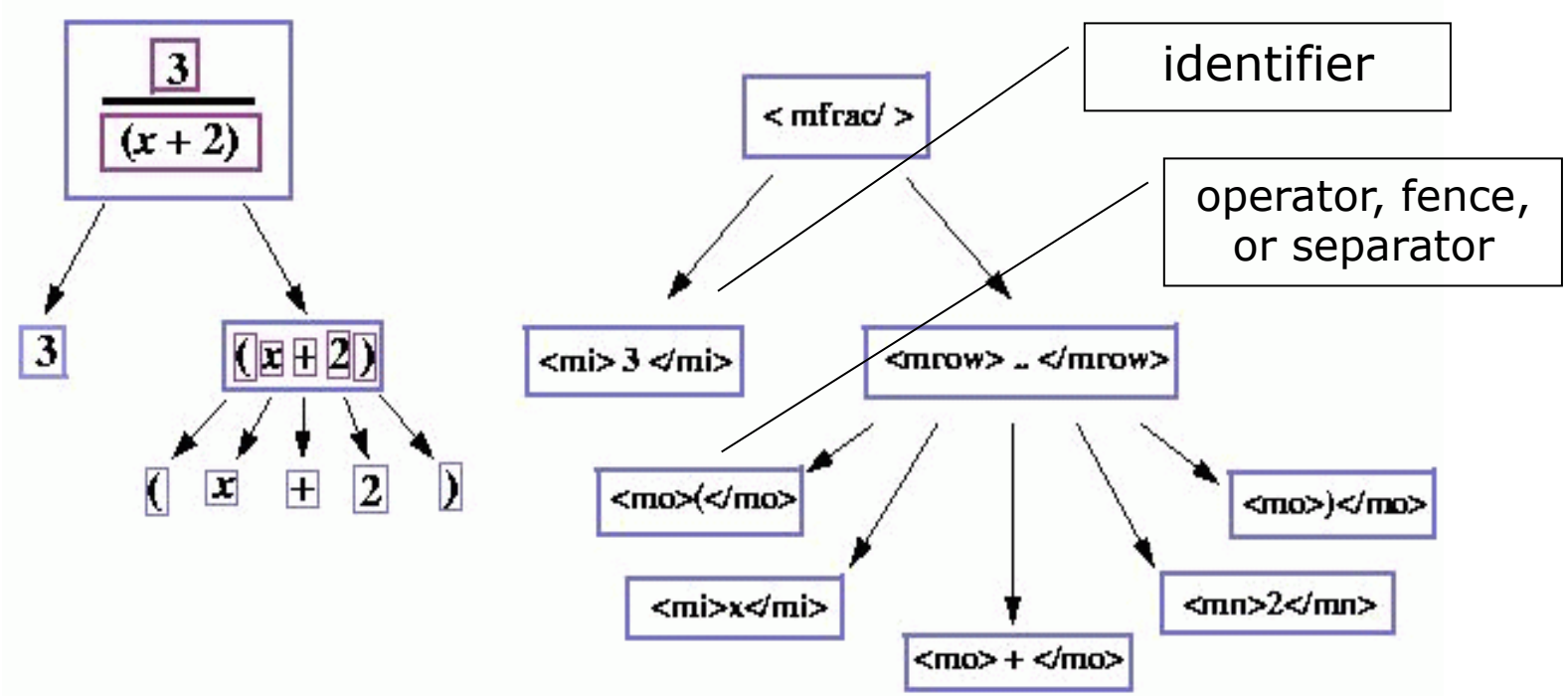
XHTML

- XHTML 1.0 is a reformulation of the three HTML 4 document types (*strict*, *transitional*, and *frameset*) as applications of XML 1.0
- XHTML documents are XML conforming
- As the XHTML family evolves, XHTML 1.0 documents be more likely to interoperate within and among various XHTML environments.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to <a href="http://vlib.org/">vlib.org</a>.</p>
  </body>
</html>
```

Source: http://www.w3.org/TR/xhtml1/

MathML example

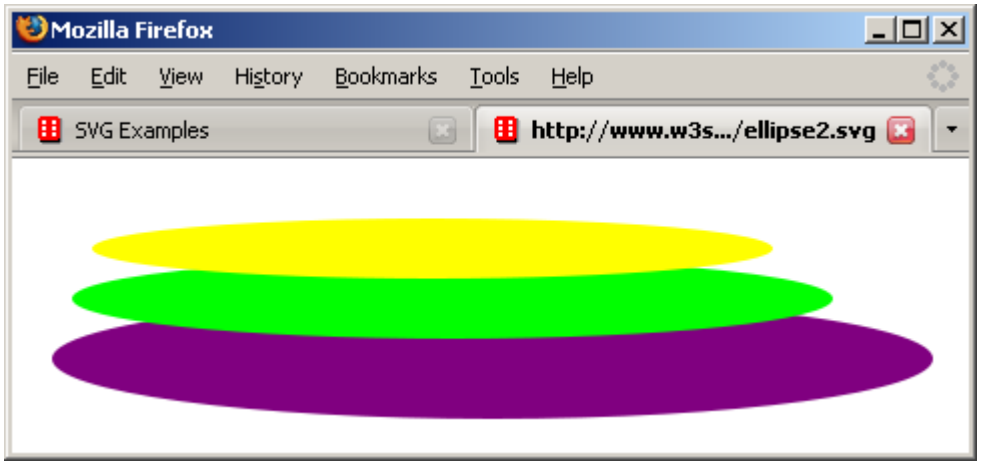


- MathML is about encoding the structure of mathematical expressions so that they can be displayed, manipulated and shared over the World Wide Web.

http://www.w3.org/TR/MathML3/

SVG (scalable vector graphics)

- SVG is a language for describing two-dimensional graphics in XML
- SVG benefits:
  - **Zooming**
  - **Text stays text:** text in SVG images remains editable and searchable
  - **Small file size**
  - **Display independence**
  - **Interactivity and intelligence.**



Source of: http://www.w3schools.com/svg/ellipse2.svg - Mo...

File Edit View Help

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN"
"http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">

<svg width="100%" height="100%" version="1.1"
xmlns="http://www.w3.org/2000/svg">

<ellipse cx="240" cy="100" rx="220" ry="30"
style="fill:purple"/>

<ellipse cx="220" cy="70" rx="190" ry="20"
style="fill:lime"/>

<ellipse cx="210" cy="45" rx="170" ry="15"
style="fill:yellow"/>

</svg>
```

Line 15, Col 22

http://www.w3schools.com/svg/

VoiceXML

- **VoiceXML** (**VXML**) is the W3C's standard XML format for specifying interactive voice dialogues between a human and a computer
- VoiceXML has tags that instruct the **voice browser** to provide speech synthesis, automatic speech recognition, dialog management, and audio playback.

```
<?xml version="1.0"?>
<vxml version="2.0" xmlns="http://www.w3.org/2001/vxml">
        <form>
                <block>
                        <prompt> Hello world! </prompt>
                </block>
        </form>
</vxml>
```

Tomcat Configuration

- A Java 2EE web application is described by a file called web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app
    PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
    "http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
  <servlet>
    <servlet-name>bob</servlet-name>
    <servlet-class>HitServlet</servlet-class>
  </servlet>

  <servlet-mapping>
    <servlet-name>bob</servlet-name>
    <url-pattern>/hits</url-pattern>
  </servlet-mapping>
</web-app>
```

Wireless Markup Language (1.x)

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.3//EN"
   "http://www.wapforum.org/DTD/wml13.dtd">

<wml>
  <card id="card1" title="WML Tutorial">
    <p>Hello World</p>
  </card>

  <card id="card2" title="WML Tutorial">
    <p>Welcome to the world of WML</p>
  </card>
</wml>
```

http://www.developershome.com/examples/wap/wml/helloWorldEg1.wml

Wireless Markup Language 2.0

- XHTML MP - the markup language defined in WAP 2.0. is a subset of XHTML

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"
    "http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
    <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="it">
    <head>
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <title>Südtiroler Bürgernetz - Rete Civica dell'Alto Adige - MOBILE version</tit.
        <link href="css/mobile.css" rel="stylesheet" type="text/css" />
    </head>
    <body id="indexpage">
        <div id="header">
            <div id="logo">
                <img src="images/logo.gif" alt="Südtiroler Bürgernetz - Rete Civica dell'
            </div>
            <h1>Südtiroler Bürgernetz<br />
            Rete Civica dell'Alto Adige</h1>
        </div>
        <div id="content">
            <ul>
                <li><a href="home_d.htm" xml:lang="de">Deutsch</a></li>
                <li><a href="home.htm">Italiano</a></li>
            </ul>
        </div>
    </body>
</html>
```

http://www.provincia.bz.it/mobile

Simple example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
   <Course Title="IT" Lect="F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title="ATIS" Lect="F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
   </Course>
</Curriculum>
```

Document prolog

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
   <Course Title="IT" Lect="F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title="ATIS" Lect="F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
   </Course>
</Curriculum>
```

- Indicates that this document is marked up in XML
- Format:
  - <?xml prop=value … ?>
  - Param values *must* be quoted with single or double quotes

Comments

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
   <Course Title="IT" Lect="F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title="ATIS" Lect="F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
   </Course>
<!- This is a comment -->
</Curriculum>
```

- Comments tag start with '<!--' and end with '-->' (as in HTML)
- `<!DOCTYPE` is not a comment
- No comment before the prologue `<?xml version="1.0" encoding="UTF-8" ?>`

Document type definition

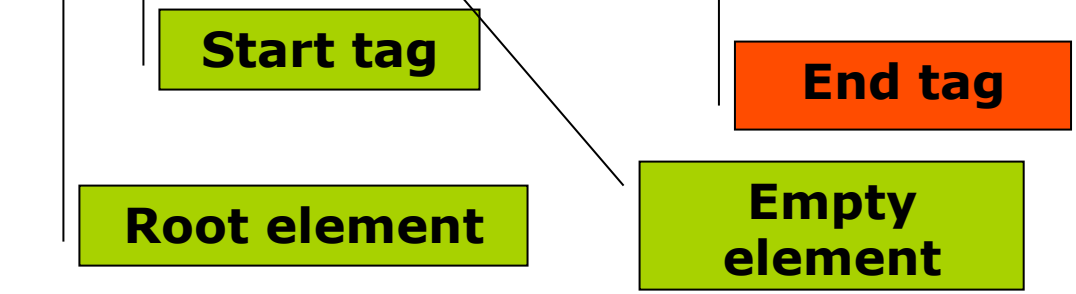
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
   <Course Title="IT" Lect="F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title="ATIS" Lect="F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
   </Course>
</Curriculum>
```

URI of the DTD

- Specifies **validation** and **root** element
- Format:
  - <!DOCTYPE root-element SYSTEM dtd>
  - Document type declaration is optional.

Elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
   <Course Title="IT" Lect="F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title="ATIS" Lect=" F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
      <Lecture/>
   </Course>
</Curriculum>
```



- One root element (must be)
- **Start Tag Format**
  - <name att=value …> content </name>
  - Or:
  - Empty Tag format: <name att=val />
- Non-empty elements **must have** a closing tag (unlike HTML)

Elements

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
   <Course Title="IT" Lect= "F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title= "ATIS" Lect="F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
      <Lecture/>
   </Course>
</Curriculum>
```

- Elements may **contain** other elements
- An element's start and end tags must reside within the same parent (*boxes cannot overlap*)
- Tag names are **case-sensitive**
- White spaces are preserved (it is up to the application to decided what to do).

Attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
    <Course Title="IT" Lect="F.Ricci">
        <Lecture>Introduction</Lecture>
        <Lecture>HTML</Lecture>
        <Lecture>XML</Lecture>
    </Course>
    <Course Title="ATIS" Lect="F. Ricci">
        <Lecture>ACF</Lecture>
        <Lecture>Personalization</Lecture>
    </Course>
</Curriculum>
```

- Used to identify specific elements, or to elaborate elements
- Values must be **quoted** (single or double)
- Attribute names (and values) are **case sensitive**
- *Not always clear whether to use attributes or elements*.

Attributes or elements?

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
    <Course Title="IT" Lect="F.Ricci">
       <Lecture>Introduction</Lecture>
       <Lecture>HTML</Lecture>
       <Lecture>XML</Lecture>
    </Course>
    <Course Title="ATIS" Lect="F. Ricci">
       <Lecture>ACF</Lecture>
       <Lecture>Personalization</Lecture>
    </Course>
</Curriculum>
```

- Attributes shouldn't really hold content
- Attribute order is ignored, whilst element order is significant
- Attributes values can be restricted (see DTDs later)
- Use attributes as unique references if needed.

Attributes Issues

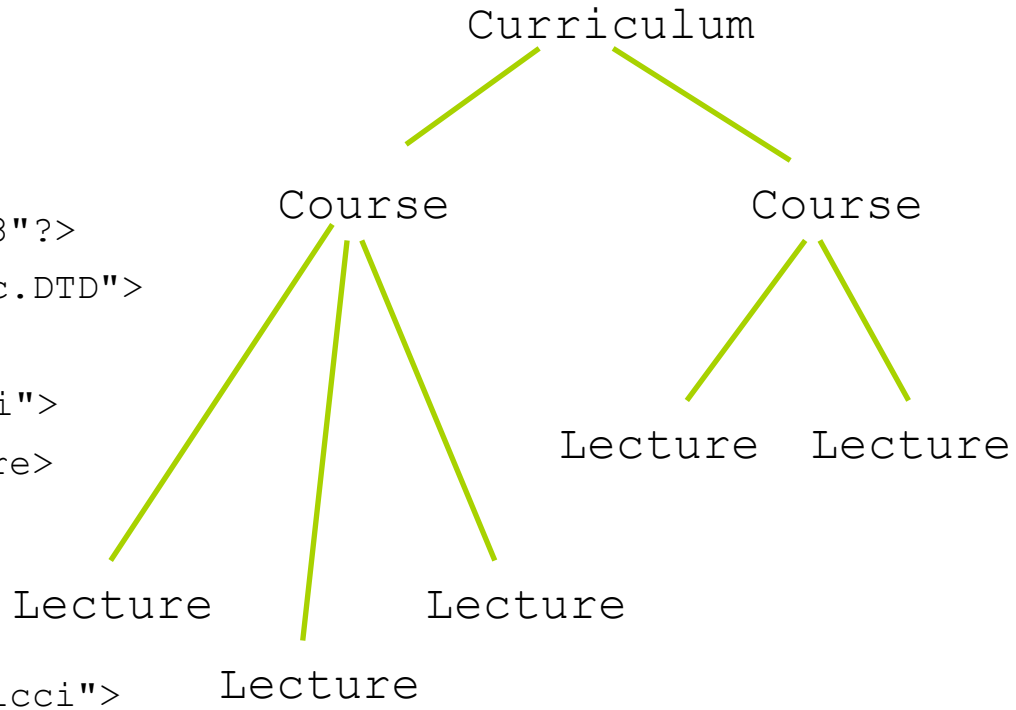
- Attributes **cannot contain multiple values** (child elements can)
- Attributes **are not easily expandable** (for future changes)
- Attributes **cannot describe structures** (child elements can)
- Attributes are **more difficult to manipulate by program code**
- Attribute values are **not easy to test** against a Document Type Definition (DTD) - which is used to define the legal elements of an XML document
- If you use attributes as containers for data, you end up with documents that are **difficult to read and maintain**
- Try to use **elements** to **describe data**
- Use attributes only to **provide information that is not relevant to the data**.

Entities

- An **entity reference** refers to the content of a named entity
  - `EntityRef ::= '&' EntityName ';'`
- Example
  - <lecture title='Introduction to: '<' and '>'' >
  - ' refers to ‘
  - < refers to <
  - > refers to >
  - " referes to “
  - & refers to &
- Entities are declared in the DTD (see later).

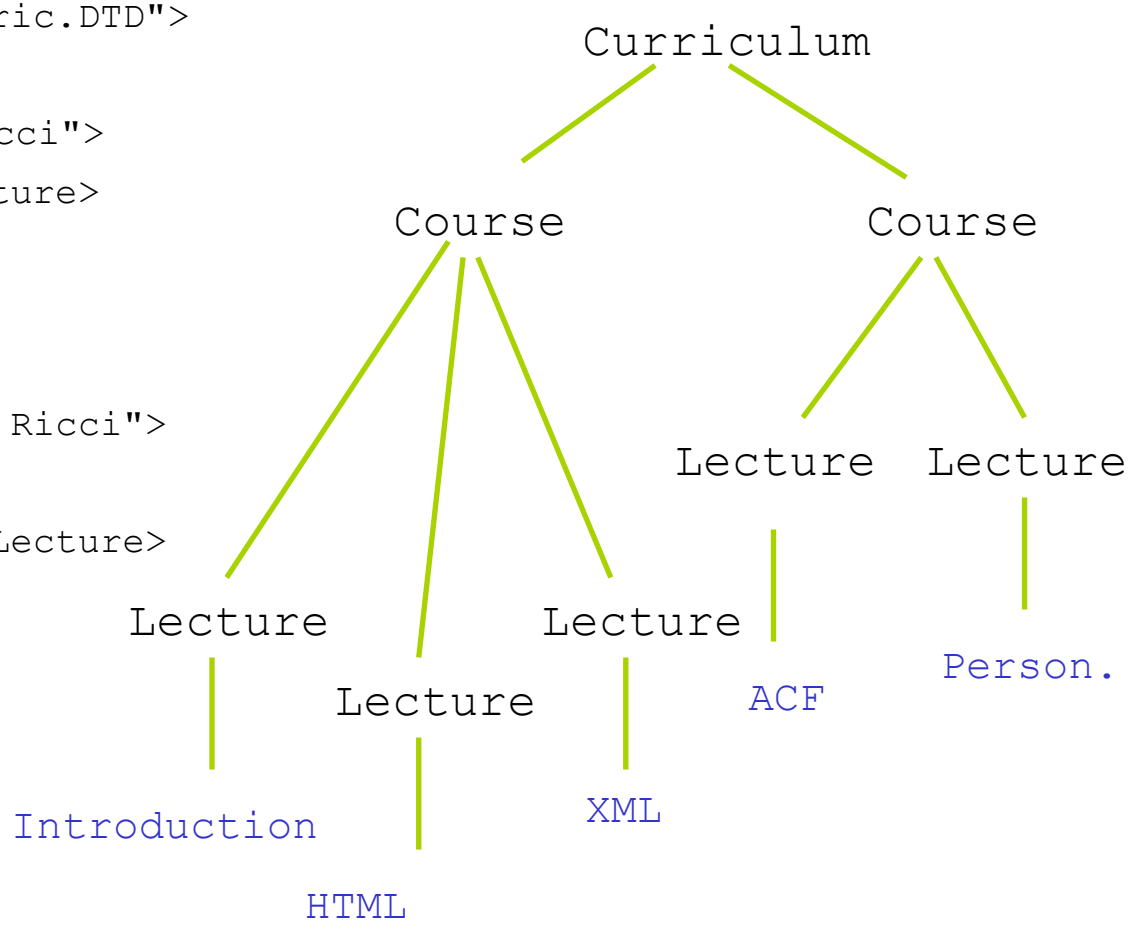
An XML document is a tree

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
   <Course Title="IT" Lect="F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title="ATIS" Lect="F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
   </Course>
</Curriculum>
```



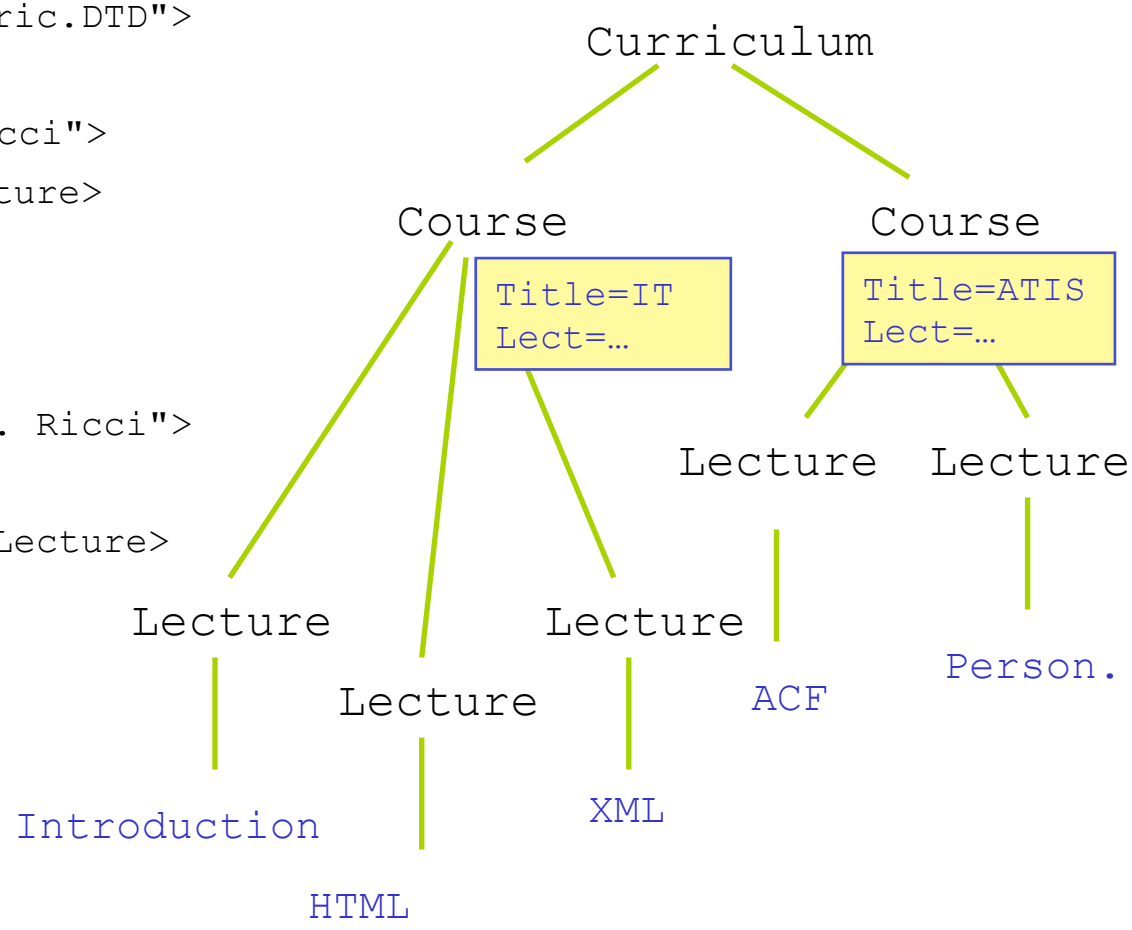
… with content at its leaves

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
   <Course Title="IT" Lect="F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title="ATIS" Lect="F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
   </Course>
</Curriculum>
```



… and attributes

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
   <Course Title="IT" Lect="F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title="ATIS" Lect=" F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
   </Course>
</Curriculum>
```



Well-formedness

- **Element containing text or elements, must have start and end tags**

Good

```
<curriculum>
  <course>Z</course>
  <course>Java</course>
</curriculum>
```

Bad

```
<curriculum>
  <course>Z
  <course>Java
</curriculum>
```

Well-formedness

- Element containing text or elements, must have start and end tags
- **Empty element's tag must close with />**

Bad

```
<graphic filename="icon.png">
```

Good

```
<graphic filename="icon.png"/>
```

Well-formedness

- Element containing text or elements must have start and end tags
- Empty element's tag must close with />
- **Attributes must be in quotes**

Good

```
<course Title="Java">
<course Title='Z'>
```

Bad

```
<course Title=UML>
```

Well-formedness

- Element containing text or elements must have start and end tags
- Empty element's tag must close with />
- Attributes must be in quotes
- **Elements must not overlap**

Bad

```
<curriculum>
  <course>Z
  </curriculum>
</course>
```

Good

```
<curriculum>
  <course>Z</course>
</curriculum>
```

Well-formedness

- Element containing text or elements must have start and end tags
- Empty element's tag must close with />
- Attributes must be in quotes
- Elements may not overlap
- **Markup chars must not appear in parsed content**

Good

```
<equation>5 &lt; 2</equation>
```

Bad

```
<equation>5 < 2</equation>
```

Well-formedness

- Element containing text or elements must have start and end tags
- Empty element's tag must close with />
- Attributes must be in quotes
- Elements may not overlap
- Markup chars must not appear in parsed content
- **Element (and attribute) names start with letters, ':' or '_', and contain letters, numbers, '-', '.', ':', and '_'**

Good

```
<curriculum>
<_course>
<time-slot>
<book.chapter>
```

Bad

```
<9example>
<the first>
<book*chapter>
```

Pros and cons of adding a grammar

PROS

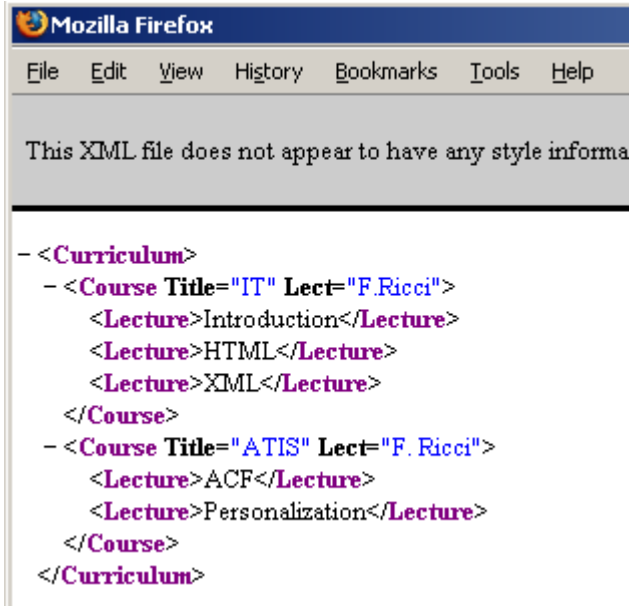
- Grammars enables documents to be validated
  - Enforce restrictions such as required fields, limited choices
- Serves as a clear description of the syntax for users and developers
- Can act as a standard e.g. XHTML
- Good for debugging

CONS

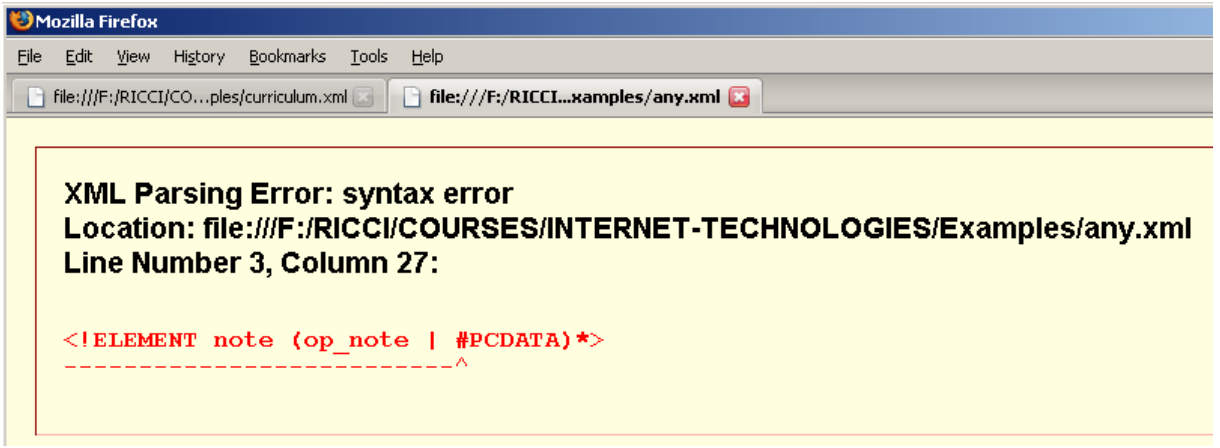
- More effort during development
- Grammars need to be maintained
- Can slow down processing
- Need to learn a new syntax (although it is trivial for CS!)

Viewing XML files

- You can view an XML file with the browser

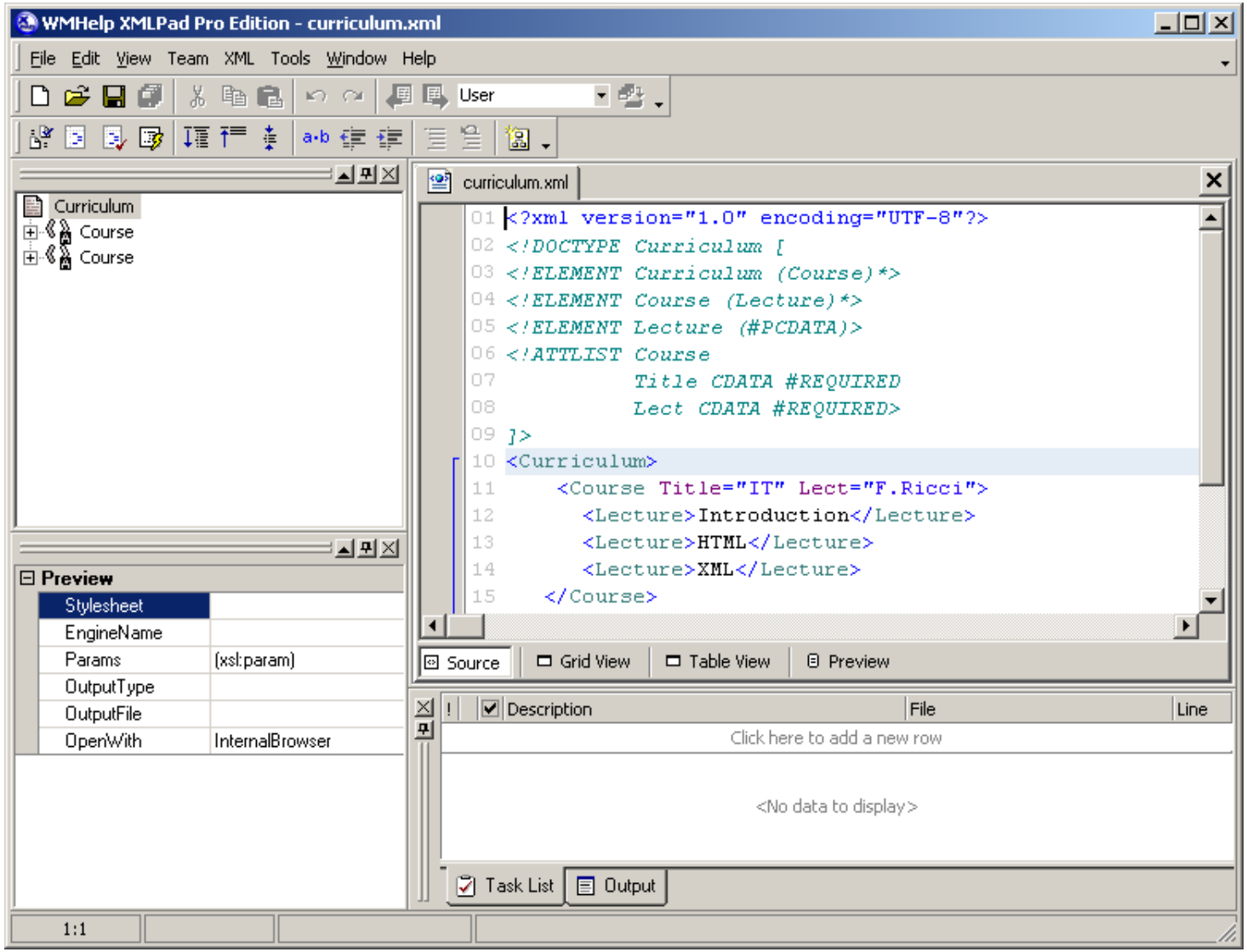


- The browser will parse the file and detect errors



XML Editors

- Are better suited to create, edit, validate, and use XML files



XML and CSS

- It is possible to use CSS to format an XML document
- CD catalogue

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/css" href="cd_catalog.css"?>
<CATALOG>
   <CD>
        <TITLE>Empire Burlesque</TITLE>
        <ARTIST>Bob Dylan</ARTIST>
        <COUNTRY>USA</COUNTRY>
        <COMPANY>Columbia</COMPANY>
        <PRICE>10.90</PRICE>
        <YEAR>1985</YEAR>
   </CD>
```

- CSS file contains rules

```
CATALOG {
background-color: #ffffff;
width: 100%;}
```

- Output

Document Type Definition (DTD)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum SYSTEM "Curric.DTD">
<Curriculum>
    <Course Title="IT" Lect="F.Ricci">
        <Lecture>Introduction</Lecture>
        <Lecture>HTML</Lecture>
        <Lecture>XML</Lecture>
    </Course>
    <Course Title="ATIS" Lect=" F. Ricci">
        <Lecture>ACF</Lecture>
        <Lecture>Personalization</Lecture>
    </Course>
</Curriculum>
```

DOCTYPE declaration

- The DOCTYPE declaration points to **markup declarations that provide a grammar** for a class of documents
- This grammar is known as a **document type definition**, or **DTD**.

DTD Example - XHTML

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
       "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type" />
  <title></title>
  <meta content="Francesco Ricci" name="author" />
</head>
<body>
Hello World!
</body>
</html>
```

DOCTYPE declaration

- The DOCTYPE is “html” (**name** of the doctype), and it is “PUBLIC”
- "-//W3C//DTD XHTML 1.0 Strict//EN“ is the **public identifier** of the DTD
- "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd“ is the **physical address** URI of the DTD.

Internal DTD

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE Curriculum [
<!ELEMENT Curriculum (Course*)>
<!ELEMENT Course (Lecture*)>
<!ELEMENT Lecture (#PCDATA)>
<!ATTLIST Course Title CDATA #REQUIRED Lect CDATA #REQUIRED>
]>
<Curriculum>
   <Course Title="IT" Lect="F.Ricci">
      <Lecture>Introduction</Lecture>
      <Lecture>HTML</Lecture>
      <Lecture>XML</Lecture>
   </Course>
   <Course Title="ATIS" Lect="F. Ricci">
      <Lecture>ACF</Lecture>
      <Lecture>Personalization</Lecture>
   </Course>
</Curriculum>
```

Root element of the XML document

- Here the DTD is written in the same XML doc (bad idea!- why?)

Document Type Definition (DTD)

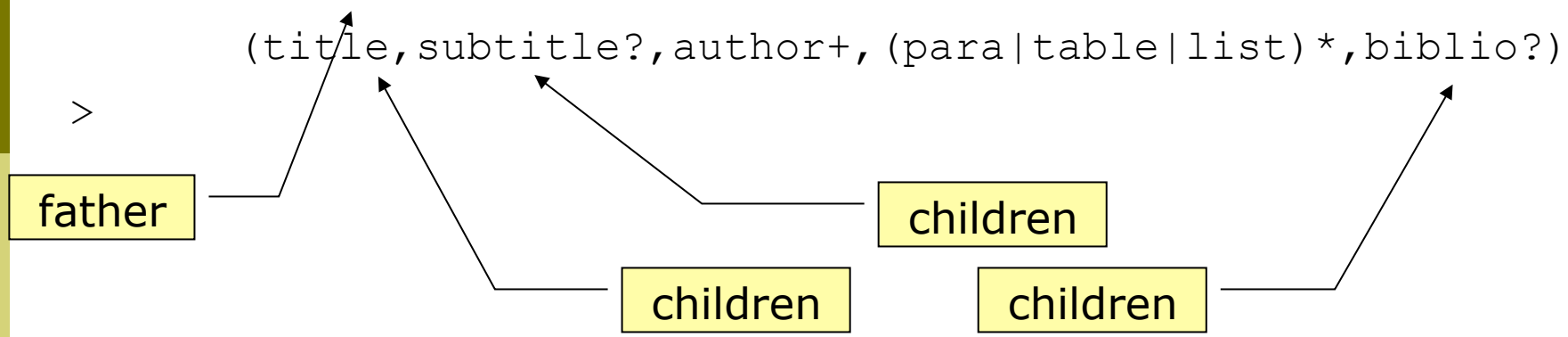
```
<!ELEMENT Curriculum (Course*)>

<!ELEMENT Course (Lecture*)>

<!ATTLIST Course
   Title CDATA #REQUIRED
   Lect  CDATA #REQUIRED
>

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

- A DTD is a sequence of declarations
- Easy to understand for CS
- #PCDATA keyword stands for “parsed character data” and means that the textual content will be parsed to look for XML entities (see later)

DTD specification is described in the XML specification document

Element definitions

```
<!ELEMENT article
        (title,subtitle?,author+,(para|table|list)*,biblio?)
>
```



`a,b,c`	sequence
`a?`	option
`a*`	zero or more
`a+`	one or more
`(a|b|c)`	alternatives
`(a,b,c)`	grouping

Element definitions

- **Mixed content**
  - DTD extract:
```
<!ELEMENT note (#PCDATA | op_note)*>
<!ELEMENT op_note (#PCDATA)>
```
  - XML extract:
```
<note> Bad state
    <op_note>That was really bad</op_note>
    <op_note>Who is the culprit?</op_note>
    But at the end we survived
</note>
```
- **Any kind of content (even elements not defined in the DTD!)**
  - DTD extract:
```
<!ELEMENT note ANY>
```
```
<note> Bad state
    <op_note>That was really bad</op_note>
    <op>John</op>
</note>
```

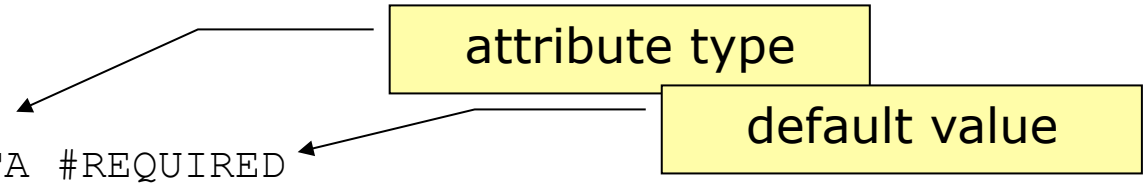
Element definitions

- Empty elements

```
<?xml version = "1.0" encoding = "utf-8"?>
<!DOCTYPE report [
   <!ELEMENT report (note*)>
   <!ELEMENT note (op_note, done?)>
   <!ELEMENT op_note (#PCDATA)>
   <!ELEMENT done EMPTY>
]>
<report>
   <note>
     <op_note>Change oil</op_note>
   <done/>
   </note>
   <note>
        <op_note>Pump was broken</op_note>
   </note>
</report>
```

Attribute definitions

```
<!ATTLIST Course
          Title CDATA #REQUIRED
          Lect  CDATA #REQUIRED
          Lect2 CDATA #IMPLIED
          Semester (first|second) "first"
>
```



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 (same rules as tag names)
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 (same rules as tag names)
NMTOKENS	The value is a list of valid XML names
ENTITY	The value is an entity
ENTITIES	The value is a list of entities

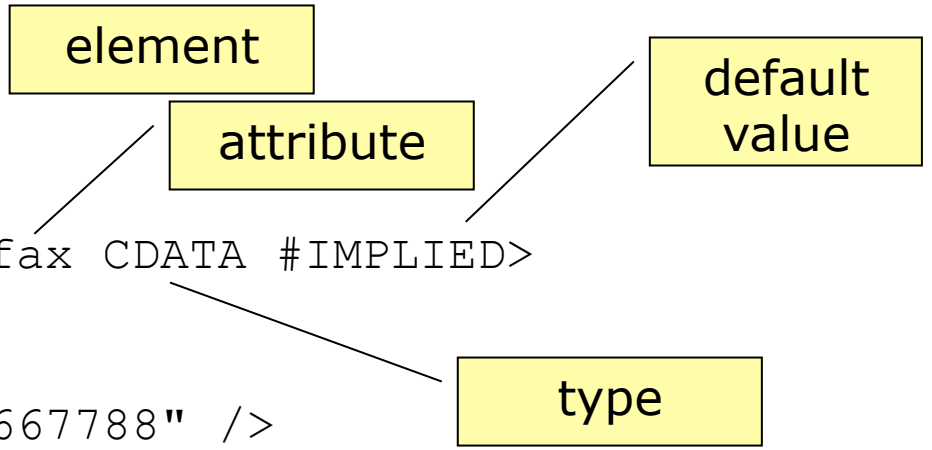
Default Values

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

IMPLIED example

DTD:

```
<!ATTLIST contact fax CDATA #IMPLIED>
```



Valid XML:

```
<contact fax="555-667788" />
```

Valid XML:

```
<contact />
```

Entities

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE books [
<!ELEMENT books (book)*>
<!ELEMENT book (author+, title,
   publisher)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT publisher (#PCDATA)>
<!ENTITY spr "Springer Verlag">
<!ENTITY mch "McGraw Hill">
]>
<books>
<book><author>JFK</author>
      <title>aaa</title>
      <publisher>&spr;</publisher>
</book>
<book><author>JOP</author>
      <title>bbb</title>
      <publisher>&mch;</publisher>
</book>
</books>
```

- DTDs can also contain entity definitions
- Simplest use (in an application) is to substitute in any parsed text (PCDATA) the entity reference (&isbn;) with the value

Parameter Entity

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % pc "(#PCDATA)">
<!ELEMENT author (#PCDATA)>
<!ELEMENT books (book)*>
<!ELEMENT book (author+, title, publisher)>
<!ELEMENT title %pc;>
<!ELEMENT publisher (#PCDATA)>
<!ENTITY spr "Springer Verlag">
<!ENTITY mch "McGraw Hill">
```

← definition

← usage
equivalent to:
<!ELEMENT title (#PCDATA) >

- Parameter entities are **defined** and **used** inside the DTD
- The DTD must be external
- Typical usage: to reuse definitions.

External Entities

```
<?xml version="1.0" encoding="UTF-8"?>
<!ENTITY % pc "(#PCDATA)">
<!ELEMENT author (#PCDATA)>
<!ELEMENT books (book)*>
<!ELEMENT book (author+, title, publisher)>
<!ELEMENT title %pc;>
<!ELEMENT publisher (#PCDATA)>
<!ENTITY spr SYSTEM "spr.txt">
<!ENTITY mch "McGraw Hill">
```

- In the file "spr.txt" is contained the value of the entity, i.e., the string "Springer Verlag".

Example

```
<!DOCTYPE NEWSPAPER [

<!ELEMENT NEWSPAPER (ARTICLE+)>
<!ELEMENT ARTICLE (HEADLINE,BYLINE,LEAD,BODY,NOTES)>
<!ELEMENT HEADLINE (#PCDATA)>
<!ELEMENT BYLINE (#PCDATA)>
<!ELEMENT LEAD (#PCDATA)>
<!ELEMENT BODY (#PCDATA)>
<!ELEMENT NOTES (#PCDATA)>

<!ATTLIST ARTICLE AUTHOR CDATA #REQUIRED>
<!ATTLIST ARTICLE EDITOR CDATA #IMPLIED>
<!ATTLIST ARTICLE DATE CDATA #IMPLIED>
<!ATTLIST ARTICLE EDITION CDATA #IMPLIED>

<!ENTITY NEWSPAPER "Vervet Logic Times">
<!ENTITY PUBLISHER "Vervet Logic Press">
<!ENTITY COPYRIGHT "Copyright 2011 Vervet Logic Press">
]>
```

Example

```
<!DOCTYPE TVSCHEDULE [

<!ELEMENT TVSCHEDULE (CHANNEL+)>
<!ELEMENT CHANNEL (BANNER,DAY+)>
<!ELEMENT BANNER (#PCDATA)>
<!ELEMENT DAY (DATE,(HOLIDAY|PROGRAMSLOT+))>
<!ELEMENT HOLIDAY (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT PROGRAMSLOT (TIME,TITLE,DESCRIPTION?)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>

<!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
<!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
<!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
<!ATTLIST TITLE RATING CDATA #IMPLIED>
<!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
]>
```

Validity

- A **Valid** XML document is:
  - a **Well Formed** XML document,
  - **which also conforms to the rules of a Document Type Definition (DTD)**
- All the tools for XML editing can test validity
- *Exercise*
  - *Build an XML document that respect the DTDs shown in the previous slides*
  - *Test validity of these XML document with XMLPad (or another tool that does it, e.g., NetBeans).*

Namespaces

- Since element names in XML are not predefined, a name conflict will occur when in the same document one uses an **element name** with **different meanings**
- XML Namespaces provide a method to avoid element name conflicts
- Example: 2 different *tables*

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

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

Using Namespaces

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

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

- Instead of using only prefixes, we have added an `xmlns` attribute to the `<table>` tag to give the prefix a qualified name associated with a namespace

The XML Namespace (xmlns) Attribute

- The XML namespace attribute is placed in the start tag of an element and has the following syntax:
  - xmlns:*namespace-prefix*="*namespaceURI*"
- When a namespace is defined in the start tag of an element, all child elements with the same prefix are associated with the same namespace
- The address used to identify the namespace is **not used by the parser to look up information** - give the namespace a **unique name**
- Often companies use the namespace as a pointer to a real Web page containing information about the namespace.

Default namespace

- Defining a default namespace for an element saves us from using prefixes in all the child elements. It has the following syntax:
  - xmlns="*namespaceURI*"

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type" />
  <title></title>
  <meta content="Francesco Ricci" name="author" />
</head>
<body>
Hello World!
</body>
</html>
```

Alternative to DTDs: XML Schema

- 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)
- XML Schemas are extensible to future additions
- XML Schemas are richer and more powerful than DTDs
- XML Schemas are written in XML
- XML Schemas support data types
- XML Schemas support namespaces.

Example - DTD

```
<?xml version="1.0"?>
<note>
   <to>Tove</to>
   <from>Jani</from>
   <heading>Reminder</heading>
   <body>Don't forget me this weekend!</body>
</note>

<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

Example - XML Schema

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.w3schools.com" xmlns="http://
    www.w3schools.com" elementFormDefault="qualified">
<xs:element name="note">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="to" type="xs:string"/>
            <xs:element name="from" type="xs:string"/>
            <xs:element name="heading" type="xs:string"/>
            <xs:element name="body" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

xs:schema element

```
<xs:schema
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
     targetNamespace="http://www.w3schools.com"
      xmlns="http://www.w3schools.com"
          elementFormDefault="qualified">
...
</xs:schema>
```

the elements prefixed with xs comes from "http://www.w3.org/2001/XMLSchema" namespace

the elements defined by this schema come from the "http://www.w3schools.com" namespace

default namespace is "http://www.w3schools.com"

all elements must be namespace qualified

The example XML

```
<?xml version="1.0"?>
<note xmlns="http://www.w3schools.com"
   xmlns:xsi="http://www.w3.org/2001/XMLSchema-
   instance" xsi:schemaLocation="http://
   www.w3schools.com/ note.xsd">
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
</note>
```

Simple Elements - example

```
<xs:element name="age">
<xs:simpleType>
   <xs:restriction base="xs:integer">
        <xs:minInclusive value="0"/>
        <xs:maxInclusive value="120"/>
   </xs:restriction>
   </xs:simpleType>
</xs:element>
```

- "simple" means that it can contain only **text**
- "restriction" are constraints on the possible values

Restriction on data types

Constraint	Description
enumeration	Defines a list of acceptable values
fractionDigits	Specifies the maximum number of decimal places allowed. Must be equal to or greater than zero
length	Specifies the exact number of characters or list items allowed. Must be equal to or greater than zero
maxExclusive	Specifies the upper bounds for numeric values (the value must be less than this value)
maxInclusive	Specifies the upper bounds for numeric values (the value must be less than or equal to this value)
maxLength	Specifies the maximum number of characters or list items allowed. Must be equal to or greater than zero
minExclusive	Specifies the lower bounds for numeric values (the value must be greater than this value)
minInclusive	Specifies the lower bounds for numeric values (the value must be greater than or equal to this value)
minLength	Specifies the minimum number of characters or list items allowed. Must be equal to or greater than zero
pattern	Defines the exact sequence of characters that are acceptable
totalDigits	Specifies the exact number of digits allowed. Must be greater than zero
whiteSpace	Specifies how white space (line feeds, tabs, spaces, and carriage returns) is handled

Complex Elements

XML

```
<employee>
   <firstname>John</firstname>
   <lastname>Smith</lastname>
</employee>
```

Schema

```
<xs:element name="employee">
<xs:complexType>
   <xs:sequence>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
</xs:element>
```

Reusing the data type

```
<xs:element name="employee" type="personinfo"/>
<xs:element name="student" type="personinfo"/>
<xs:element name="member" type="personinfo"/>
```

```
<xs:complexType name="personinfo">
   <xs:sequence>
        <xs:element name="firstname" type="xs:string"/>
        <xs:element name="lastname" type="xs:string"/>
   </xs:sequence>
</xs:complexType>
```

Instead of using:
<xs:element name="employee">

XHTML

- XHTML stands for E**X**tensible **H**yper**T**ext **M**arkup **L**anguage
- XHTML is aimed to **replace** HTML
- XHTML is almost **identical** to HTML 4.01
- XHTML is a reformulation of HTML 4.01 in XML
- XHTML is a **stricter and cleaner** version of HTML
- XHTML is HTML defined as an **XML application**
- XHTML is a W3C Recommendation
- All new browsers have support for XHTML

Why XHTML

- Many pages on the WWW contain "bad" HTML

```
<html>
<head>
<title>This is bad HTML</title>
<body>
<h1>Bad HTML
</body>
```

- Browsers can show that page even if it does not follow the HTML rules - but not simple applications (as in a Mobile phone)
- In XML everything has to be marked up correctly, which results in "well-formed" documents
- XML was designed to describe **data** and HTML was designed to **display data**
- XHTML pages can be read by all XML enabled devices: you can write "well-formed" documents now, that work in all browsers and that are backward browser compatible.

HTML vs. XHTML

- XHTML elements must be **properly nested**
- XHTML elements must always be **closed**
- XHTML elements must be in **lowercase**
- XHTML documents must have **one root element**
- Many HTML documents are "already" XHTML documents

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://
www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type" />
  <title>My first XHTML</title>
</head>
<body>
Hello World!<br />
</body>
</html>
```

Is XML

A new DOCTYPE

Tags live in a namespace

Three XHTML DTDs

- **XHTML 1.0 Strict**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

- Clean markup, free of presentational clutter - used together with Cascading Style Sheets
- **XHTML 1.0 Transitional**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
   Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/
   xhtml1-transitional.dtd">
```

- To take advantage of HTML's presentational features - support browsers that don't understand Cascading Style Sheets
- **XHTML 1.0 Frameset**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN"
   "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
```

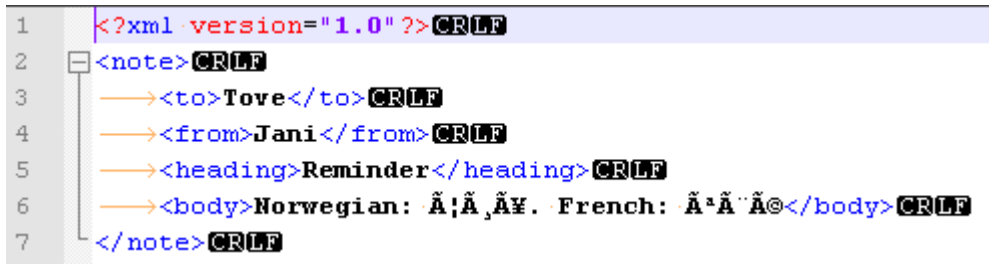
- To use HTML Frames to partition the browser window into two or more frames.

From HTML to XHTML

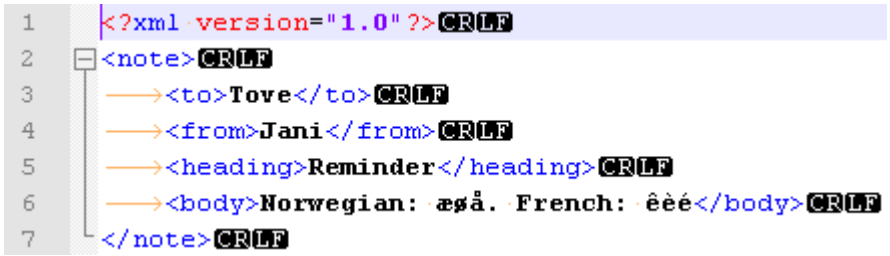
- **Modify the DOCTYPE** declaration as the first line of every page
- **Search and replace all upper case tags** (and attributes) **with lowercase tags** (and attributes)
- Check every page to see that **attributes values were properly quoted**
- **Empty tags** are not allowed in XHTML: hence (e.g.) <hr> and
 tags should be replaced with <hr /> and
- **Validate all pages** against the official W3C DTD.

Encoding

- A file encoded as UTF but not explicitly stated



decoded as ANSI



decoded as UTF-8

- Better to state what is the encoding

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
        <to>Tove</to>
        <from>Jani</from>
        <heading>Reminder</heading>
        <body>Norwegian: æøå. French: êèé</body>
</note>
```