XML/XHTML

ESA 2016/2017

Adam Belloum

a.s.z.belloum@uva.nl

Material Prepared by Eelco Schatborn

Announcements +++

- Feedback assignments
  - Exactly **one week** to finish the assignment (**deadlines Midnight Sundays and Wednesdays**)
  - Personal feedback (F2F or email) Monday lab + personal report: the required improvements/pass (no grade)
- Group assignment
  - Each group of 2/3 students will have 2 mn to present the most important points of " the further reading" (beginning of each lecture) (elect a spokesman)
  - G1:1-6, G2:6-11, G3:12-18, G4:19-24, G4:26-30, G6:33-38, G7:39-43

Content

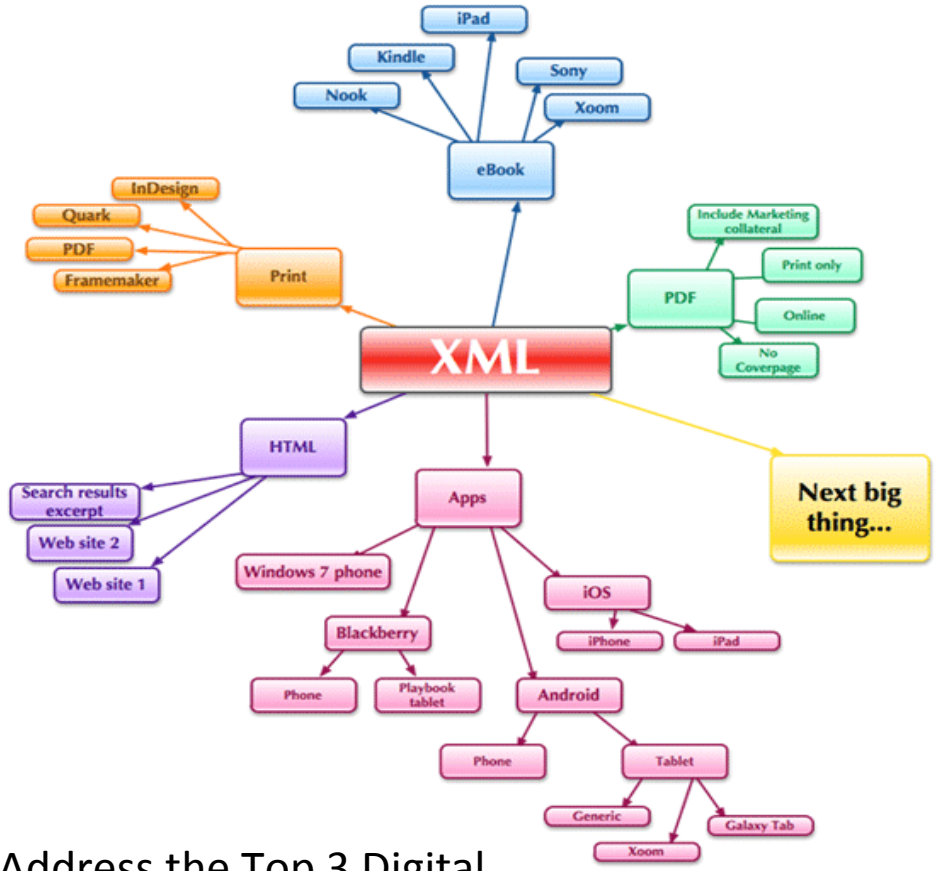
- Why XML evolved
- What it is XML
- Document Type Declaration
- XML Schema Definition

XML is used every where: Digital Publishing

3 Challenges

- The Indexing and Metadata
- The Multichannel Publishing
- The Content Re-use Challenge

"XML is a **flexible way** to author and **represent** content describing what the content is rather than how it should look."



Make XML a Solution Instead of a Challenge: Address the Top 3 Digital Publishing Challenges With XML: http://www.reallysi.com/images/pdf/Make_XML_a_solution_instead_of_a_challenge_RSuiteCloud.pdf

XML is used every where: on the web

- Web searching and automating Web tasks
- e-business applications
- …

XML provides a standard method to **access information**, making it easier for applications and devices of all kinds to use, store, transmit, and display data.

A common data format was needed …

- 1960-1980 **Infrastructure** for the Internet
- 1986 SGML (**Standard Generalized Markup Language**) for defining and representing structured documents
- 1991 **WWW** and **HTML** introduced for the Internet
- 1991 **Business adopts** the WWW technology; expansion in the use of the Internet
- 1995 **New businesses** evolve, based on the connectivity of people all over the world and connectivity of applications built by various software providers (B2C, B2B)

Urgent **need** for a **common data format** for the Internet

The common data format had to be …

- ***Simple***, common rules easy to understand by people with different backgrounds (like HTML)
- Capability to ***describe Internet resources*** and their relationships (like HTML)
- Capability to ***define information structures*** for different kinds of business sectors (unlike HTML, like SGML)

The common data format had to be …

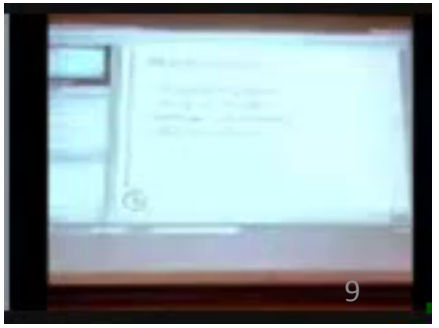
- Format ***formal enough*** for computers
  & ***clear enough*** to be human-legible (like SGML)
- Rules ***simple enough*** to allow easy building
  of software (unlike SGML)
- ***Strong*** support for diverse ***natural languages***
  (unlike SGML)

What is XML good for:

- XML provide the capability to **represent** and **enable access** to data/information

- XML has the quality of
  - Word processors (representation)
  - Databases (access)

What is XML good for, Bob Boiko a Senior Lecturer for the iSchool. University of Washington)
http://www.youtube.com/watch?v=CDawM8F-fqY&feature=related



XML = eXtensible **M**arkup **L**anguage

A markup language is a modern system for **annotating a text**, the idea and terminology evolved from the "marking up" of manuscripts

→ in a way that is **syntactically distinguishable** from that text.

- DocBook
- HTML
- MathML
- RTF
- TEX
- Troff
- Wikipedia/WikiMedia markup language
- DokuWiki
- . . .

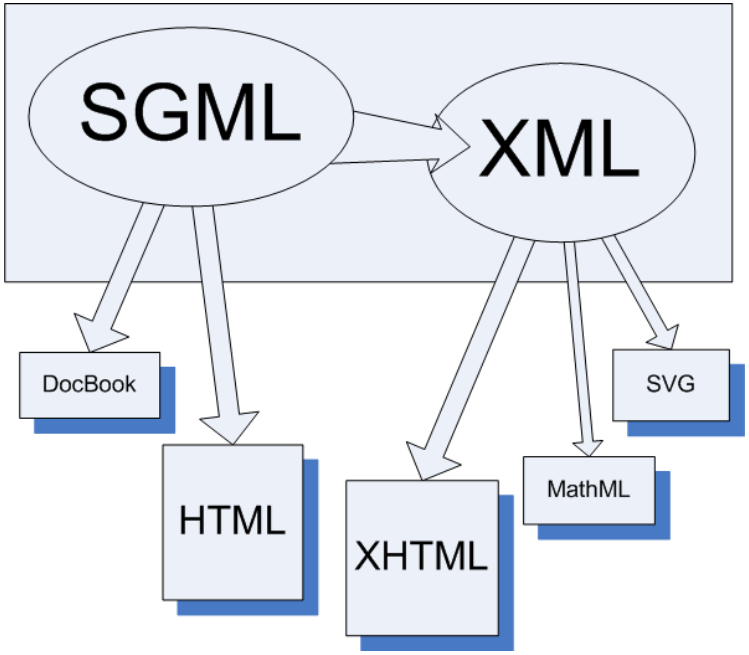
http://en.wikipedia.org/wiki/List_of_markup_languages

XML = e**X**tensible **M**arkup **L**anguage

A **set of rules** for **defining** and **representing information** as **structured documents** for applications on the Internet; a restricted form of SGML

T. Bray, J. Paoli, and C. M. Sperberg-McQueen (Eds.), Extensible Markup Language (XML) 1.0, **W3C Recommendation 10- February-1998**, http://www.w3.org/TR/1998/REC-xml-19980210/.

T. Bray, J. Paoli, C. M. Sperberg-McQueen, and E. Maler (Eds.), Extensible Markup Language (XML) 1.0 (Second Edition), **W3C Recommendation 6 October 2000**, http://www.w3.org/TR/2000/REC-xml-20001006/.



SGML
(Standard Generalized Markup Language)

- SGML was originally designed to enable the **sharing** of machine-readable large-project documents in government, law, and industry
- Borne from **the publishing world**
- Existed before HTML
- Using SGML you can define markup languages for documents

- A conforming SGML document must be:
  - either a **type-valid** SGML document,
  - a **tag-valid** SGML document,
  - or both.

- ISO standard; ISO 8879:**1986** SGML

Wikipedia: A fragment of the OED (1985), showing SGML markup

International Organization for Standardization (ISO)

- setting international Standards
- representatives from various
  national standards organizations
- founded 23 February 1947, headquartered in Geneva, Switzerland
- abbreviation ISO comes from greek isos meaning `equal'
- named ISO because the acronym would be different in each Country
- naming ISO the acronym was the first standardisation

eXtensible Markup Language (XML)

- **General-purpose specification** for creating custom markup Languages
  - you can define your **own elements** for structuring data
  - By adding **semantic** constraints, application languages can be implemented in XML.
  - XML is used as the specification language for a number of application languages like: XHTML, RSS, MathML, GraphML, MusicXML, . . .

XML based languages

- **XHTML** Markup language for WWW pages
- **RSS** Markup language for Really Simple Syndication (v2)
- **MathML** Markup language for formula's
- **GraphML** Markup language for graphs
- **SVG** Scalable Vector Graphics
- **MusicXML** Markup language for sheet music
- **SAML** Security Assertion Markup Language
- **VML** Voice Markup language
- **SYNCML** Device Synchronisation
- . . .

http://en.wikipedia.org/wiki/List_of_XML_markup_languages

XML Basic Rules

- Rule 1:
  - Information is represented in **units** called ***XML documents***.
- Rule 2:
  - An **XML document** contains one or more ***elements***.
- Rule 3:
  - An **element** denoted in the document by **explicit markup**
    - has a **name**,
    - can **contain other** elements,
    - can be associated with ***attributes***.

and lots of other rules ...

Example of an XML document

```
1. <?xml version="1.0"?>
2. <catalog>
3.   <product category = "mobile phone">
4.     <mfg>Nokia</mfg> <model>8890</model>
5.     <description>
6.         Intended for EGSM 900 and GSM 190 networks …
7.     </description>
8.    <clock setting= "nist" alarm = "yes"/>
9.   </product>

10.  <product category = "mobile phone">
11.   <mfg>Ericsson</mfg><model>A3618</model>
12.              <description>...</description>
13.  </product>
14.       ...
15.</catalog>
```

Classes of text markup

descriptive (the previous example)

presentational

Mobile phones:
Nokia 8890
Ericsson A3618

```
<document>
  <newPage style="box"/>
  <bold>Mobile phones:</bold>
  <list>
     <newItem/>
       <italic>Nokia8890</italic>
     <newItem/>
       <italic>EricssonA3618</italic>
  </list>
</document>
```

procedural

XML is primarily for descriptive markup.

XML units

- **XML document** comprises one or more *entities*
  - A "**document entity**" serves as the **root**
  - Other **entities** may include
    - **external** portion of DTD
    - **parsed** character data, which replaces any references to the entity
    - **unparsed** data
  - Entities located by URIs

XML 1.0 fundamentals

- XML declaration: `<?xml version="1.0"?>`
- Logical data components:
  - Markup vocabulary: **elements**, **attributes**

```
<product  category = "mobile phone">
   <mfg>Nokia</mfg><model>8890</model>
   <clock  setting = "nitz" alarm = "yes"/> …
</product>
```

  - White space — `xml:space`
  - Parsed and unparsed Character Data — `PCDATA and CDATA`
  - Entity references — `&diagram;`
  - Comments — `<!-- how interesting… -->`
  - Processing instructions

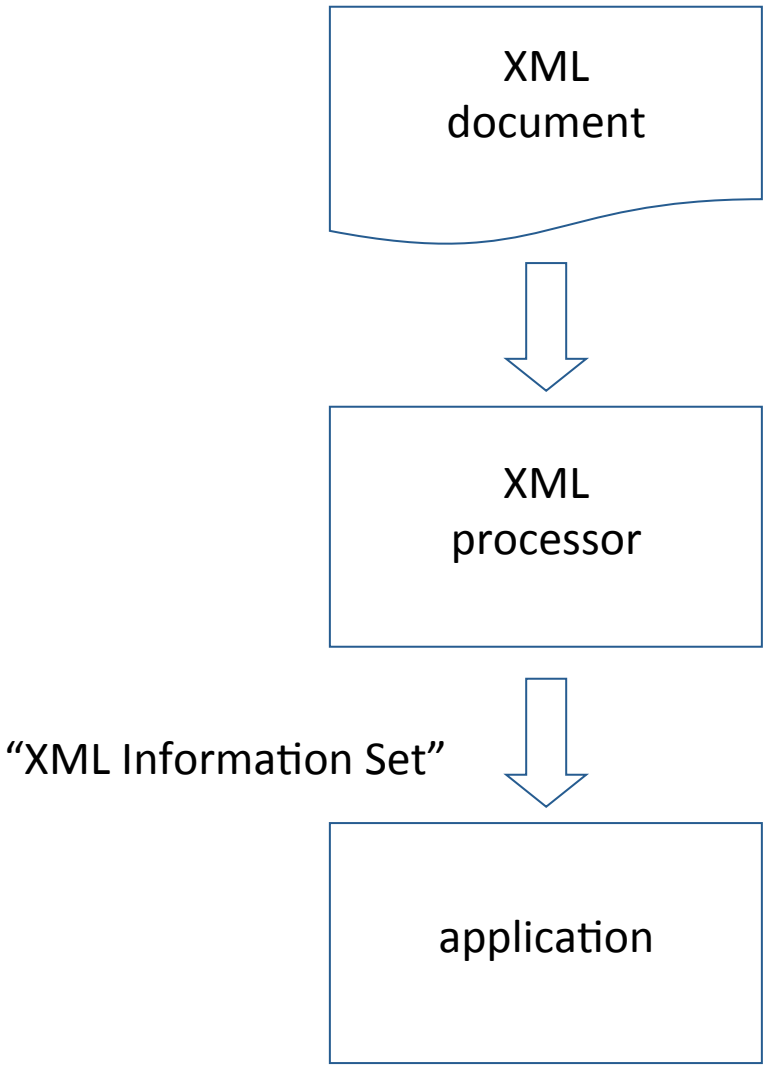
```
<?xml-stylesheet  href="catalog-style.css"
  type="text/css"?>
```

XML 1.0 Fundamentals

The ***processor*** **analyzes** the markup and passes structured information to an *application*.

The specification places requirements on what an XML processor must do and not do, but the application is outside its scope.

The processor is often referred to colloquially as an *XML parser*.



Example SVG ML (Scalable Vector Graphics)

Source Code: Submit Code »

```
<svg id="body" width="500" height="10.5cm" viewBox="0 0 210 135" >

<desc>
      Rectangle with red border and light blue interior, with (intended) gray
shadow rectangle.
</desc>

<rect x="10" y="20" width="150" height="70" fill="#eeeeff"   stroke="red"
stroke-width="1" />

<rect x="10" y="20" width="150" height="70" transform="translate(3, 3)"
fill="#999999" "stroke="#999999" stroke-width="1" />




</svg>
```

Result: W3Schools.com - Try it yourself

- Why XML evolved
- What it is XML
- Document Type Declaration
- XML Schema Definition

XML 1.0 Fundamentals

- Conformance:
  - *Well-formed:*
    - syntactically correct tags
    - matching tags
    - nested elements
    - all entities declared before they are used
  - *Valid*:
    - **well-formed**
    - Type Definition (Doc Type Definition or Schema Definition)
    - unique IDs
    - no dangling IDREFs

XML 1.0 Fundamentals

- Markup declarations: Document Type Definition (DTD):
  - Internal *vs*. external declarations
  - **Element** types: EMPTY, children, mixed, ANY

```
<!ELEMENT product (mfg, model, description , clock?)>
<!ELEMENT description  (#PCDATA | feature)*>
<!ELEMENT clock           EMPTY>
```

  - **Attribute** types: CDATA, ID, IDREF(-S), ENTITY(-TIES), NMTOKEN(-S)

```
<!ATTLIST product category  #PCDATA >

<!ATTLIST clock  setting CDATA  #IMPLIED
                 alarm  (yes | no )  "yes"    >
```

  - Notations
  - Entities

Document Type Definition (DTD)

- set of **declarations** that **conform** to a particular **markup syntax** that describe a **class**, or **type** of document, **in terms of constraints** on the structure of that document
- often used to **describe a document** authored in the DTD language
- specifies the **syntax** of an "application" of SGML or XML, (like HTML or XHTML)
- one of several SGML and XML schema languages, others include XML Schema and RELAX NG

DTD declarations

- A DTD **is associated with** an XML document **via a**
  - **Document Type Declaration**,
  - **a tag** near the start of the XML document.

- declarations can be **internal** (in the DTD declaration itself) or **external** (located in a separate file)

- **external declarations** may be referenced via a
  - public identifier and/or a system identifier
  - Example

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

XML with DTD

- **XML without DTD**

```
<?xml version="1.0" standalone="yes"?>
 <hello> Hello XML et hello cher lecteur ! </hello>
```

- **XML with Internal DTD**

```
<?xml version="1.0" standalone="yes"?>
  <!DOCTYPE hello [
     <!ELEMENT hello (#PCDATA)>
     ]>
  <hello> Hello XML et hello dear readers ! </hello>
```

- **XML with external DTD**

```
<?xml version="1.0" ?>
 <!DOCTYPE hello SYSTEM "hello.dtd">
 <hello> This is a very simple XML document </hello>
```

Using the DTD Example

An example of a very simple XML DTD to describe a **list of persons** (name, birthday (optional), gender (optional), and social security number (optional). Elements should not contain any <,>,& etc.)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE people_list SYSTEM "example.dtd">
<people_list>
  <person>
   <name>Eelco Schatborn</name>
   <birthdate>6/6/1974</birthdate>
   <gender>Male</gender>
</person>
</people_list>
```

XML DTD Example

- An example of a very simple XML DTD to describe a **list of persons** (name, birthday (optional), gender (optional), and social security number (optional). Elements should not contain any <,>,& etc.)

```
<!ELEMENT people_list (person*)>
<!ELEMENT person (name, birthdate ,
   gender , socialsecuritynumber )>
<!ELEMENT name (#PCDATA)>
<!ELEMENT birthdate (#PCDATA)>
<!ELEMENT gender (#PCDATA)>
<!ELEMENT socialsecuritynumber(#PCDATA)>
```

- Why XML evolved
- What it is XML
- Document Type Declaration
- XML Schema Definition

Both DTD and XSD

- Declare the list of elements and attributes.
- Describe how those elements are grouped, nested or used within the XML.
  - In other words, they define the rules by which an XML file can be created
- Provide methods for restricting, or forcing, the type or the format of an element.
  - For example, within the DTD or Schema you can force a date field to be written as 01/05/06 or 1/5/2006.

DTD vs. XSD

1. XML Schema is **namespace aware**, while DTD is not.
2. XML Schemas are written in XML, while DTDs are not.
3. XML Schema is **strongly typed**, while DTD is not.
4. XML Schema has **a wealth of derived and built-in data types** that are not available in DTD.
5. XML Schema **does not allow inline definitions**, while DTD does.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<shiporder orderid="889923" >

  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  <item>
    <title>Hide your heart</title>
    <quantity>1</quantity>
    <price>9.90</price>
  </item>
</shiporder>
```

The XML document consists of

- "**shiporder**" the root element, that contains
  - a required attribute called "**orderid**".
  - three different child elements: "**orderperson**", "**shipto**" and "**item**".
- "**item**" element appears twice, and it contains a "**title**", an optional "**note**" element, a "**quantity**", and a "**price**" element.

```
<shiporder orderid="889923" xmlns:xsi="h
      xsi:noNamespaceSchemaLocation="shi
  <orderperson>John Smith</orderperson>
  <shipto>
    <name>Ola Nordmann</name>
    <address>Langgt 23</address>
    <city>4000 Stavanger</city>
    <country>Norway</country>
  </shipto>
  <item>
    <title>Empire Burlesque</title>
    <note>Special Edition</note>
    <quantity>1</quantity>
    <price>10.90</price>
  </item>
  </shiporder>
```

Create a file **shiporder.xsd**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="shiporder">
<xs:complexType>
  <xs:sequence>
   <xs:element name="orderperson" type="xs:string"/>
   <xs:element name="shipto">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="name" type="xs:string"/>
      <xs:element name="address" type="xs:string"/>
      <xs:element name="city" type="xs:string"/>
      <xs:element name="country" type="xs:string"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>
   <xs:element name="item" maxOccurs="unbounded">
    <xs:complexType>
     <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="note" type="xs:string" minOccurs="0"/>
      <xs:element name="quantity" type="xs:positiveInteger"/>
      <xs:element name="price" type="xs:decimal"/>
     </xs:sequence>
    </xs:complexType>
   </xs:element>

   <xs:attribute name="orderid" type="xs:string " use="required"/>

  </xs:sequence>
 </xs:complexType>
 </xs:element>
</xs:schema>
```

Create a file **shiporder.xsd**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<!-- definition of simple elements -->
<xs:element name="orderperson" type="xs:string"/>
<xs:element name="name" type="xs:string"/>
<xs:element name="address" type="xs:string"/>
<xs:element name="city" type="xs:string"/>
<xs:element name="country" type="xs:string"/>
<xs:element name="title" type="xs:string"/>
<xs:element name="note" type="xs:string"/>
<xs:element name="quantity" type="xs:positiveInteger"/>
<xs:element name="price" type="xs:decimal"/>

<!-- definition of attributes -->
<xs:attribute name="orderid" type="xs:string"/>

<!-- definition of complex elements -->
<xs:element name="shipto">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="name"/>
   <xs:element ref="address"/>
   <xs:element ref="city"/>
   <xs:element ref="country"/>
  </xs:sequence>

<xs:element name="item”>
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="title"/>
   <xs:element ref="note" minOccurs="0"/>
   <xs:element ref="quantity"/>
   <xs:element ref="price"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

 </xs:schema>
```

Create a file **shiporder.xsd**

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<xs:simpleType name="stringtype">
  <xs:restriction base="xs:string"/>
</xs:simpleType>

<xs:simpleType name="inttype">
  <xs:restriction base="xs:positiveInteger"/>
</xs:simpleType>

<xs:simpleType name="dectype">
  <xs:restriction base="xs:decimal"/>
</xs:simpleType>

<xs:simpleType name="orderidtype">
  <xs:restriction base="xs:string">
    <xs:pattern value="[0-9]{6}"/>
  </xs:restriction>
</xs:simpleType>

<xs:complexType name="shiptotype">
  <xs:sequence>
    <xs:element name="name" type="stringtype"/>
    <xs:element name="address" type="stringtype"/>
    <xs:element name="city" type="stringtype"/>
    <xs:element name="country" type="stringtype"/>
  </xs:sequence>
</xs:complexType>

<xs:complexType name="itemtype">
  <xs:sequence>
    <xs:element name="title" type="stringtype"/>
    <xs:element name="note" type="stringtype"
minOccurs="0"/>
    <xs:element name="quantity" type="inttype"/>
    <xs:element name="price" type="dectype"/>
  </xs:sequence>
</xs:complexType>

</xs:schema>
```

XSD Example

- An example of a very **simple XSD to describe a country name and population)** (taken from Wikipedia)
- Write the XSD
- Write a snippet of XML which can be validated by the schema

```
<xs:schema xmlns:xs=http://www.w3.org/2001/XMLSchema>

  <xs:element name="country" type="Country"/>
     <xs:complexType name="Country">
     <xs:sequence>
        <xs:element name="name" type="xs:string"/>
        <xs:element name="population" type="xs:decimal"/>
      </xs:sequence>
   </xs:complexType>

</xs:schema>
```

Using the XSD Example

- An example of a very **simple XSD to describe a country name and population)** (taken from Wikipedia)
- Write the XSD
- Write a snippet of XML which can be validated by the previous schema

```
<country xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="country.xsd">
   <name>France</name>
   <population>64473140</population>
</country>
```

Recursion in DTD and XSD

Recursive structures are possible in schemas.

A common example would be recursive <section> elements, say with a title and one or more paras and zero or more sections.

Here's a DTD sample:

```
<!ELEMENT section  (title , para+ , section*) >
```

Here's a Schema sample:

```
<xsd:element name="section">
   <xsd:complexType>
      <xsd:sequence>
         <xsd:element ref="title"/>
         <xsd:element ref="para" maxOccurs="unbounded"/>
         <xsd:element ref="section" minOccurs="0" maxOccurs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>
```

http://www.stylusstudio.com/xmldev/200501/post80430.html

Content (part2)

- HTML
- XHTML
- Difference between HTML and XHTML
- XHTML 1.0
- Some details about HTML documents

HTML 2.0 published November 1995 as ***IETF RFC 1866***, and declared obsolete/ historic by RFC 2854 in June 2000	**XHTML 1.0** became a W3C Recommendation on January 26, 2000.
HTML 3.2 published January 14, 1997 as W3C Recommendation	**XHTML 1.1** became a W3C Recommendation on May 31, 2001
HTML 4.0 published December 18, 1997 as W3C Recommendation	**XHTML 2.0** August 2002 -2006- W3C **released drafts**- clean break from the past **NO backward compatibility**.
HTML 4.01 (minor fixes) published December 24, 1999 as a W3C Recommendation	**XHTML 5.0** , recommendation on October 2014, is simply an XML-serialized **HTML5** data

- According to the W3C, "XHTML 1.0 is a reformulation of HTML 4.01 in XML, and combines the strength of HTML 4 with the power of XML."
- This means that, by using XHTML, you are adding the potential of the power of XML to your Web pages.
- **XHTML is "strict" HTML (in the sense you cannot get away with sloppy coding)**

Differences between XHTML and HTML

- All tags are **lowercase** (<p>)
- All documents have a **doctype** and an XHTML **namespace**
- All documents must be **well-formed**
- All tags (elements) must **be closed**, even empty ones (<empty />)
- Attribute values must be bounded by double quotes (")
- All tags must be nested correctly

Common mistakes in XHTML (1)

- Not closing an empty element
  - wrong:
  - right:
- Not closing non empty elements
  - wrong: paragraph. another paragraph.
  - right: paragraph. another paragraph.
- Bad nesting
  - wrong: This is some text.
  - right: This is some text.

Common mistakes in XHTML (2)

- Missing alternate text
  - wrong:
  - right:

- Using text directly in the body
  - wrong: <body>Welcome to my page.</body>
  - right: <body> Welcome to my page. </body>
- Nesting block levels and inline elements
  - wrong: <h2>Introduction</h2>
  - right: <h2>Introduction</h2>

Common mistakes in XHTML (3)

- Missing quotes
  - wrong: <td rowspan=3>
  - right: <td rowspan="3">
- Literal use of the ampersand (&)
  - wrong: <title>Cars & Trucks</title>
  - right: <title>Cars & Trucks</title>
- Uppercase vs. lowercase
  - wrong: <BODY><P>The Best Page Ever</P></BODY>
  - right: <body> The Best Page Ever </body>

Benefit of XTHML over HTML

- XHTML is easier to maintain
- XHTML is XSL ready
  - Extensible Style sheet Language Transformations (XSLT)

XHTML 1.0

There are three formal **Document Type Descriptions** (DTDs) for XHTML 1.0, corresponding to the three different versions of HTML 4.01

XHTML 1.0 Strict

the equivalent to **HTML 4.01 strict** , includes **elements and attributes not marked deprecated** in the HTML 4.01 specification.

XHTML 1.0 Transitional

the equivalent of **HTML 4.01Transitional**, includes **the presentational element**s (such as center, font and strike) excluded from the strict version.

XHTML 1.0 Frameset

the equivalent of **HTML 4.01 Frameset**, allows for the definition of **frameset documents** a common Web feature in the late 1990s.

Document type declarations

- **XHTML 1.0** Strict

  <!DOCTYPE html PUBLIC "//W3C//DTD XHTML 1.0 **Strict**//EN"
  "http://www.w3.org/TR/xhtml1/DTD/**xhtml1strict**.dtd">

- **XHTML 1.0** Transitional

  <!DOCTYPE html PUBLIC "//W3C//DTD XHTML 1.0
  Transitional//EN" "http://www.w3.org/TR/xhtml1.0/DTD/
  xhtml1transitional.dtd">

- **XHTML 1.0** Frameset

  <!DOCTYPE html PUBLIC "//W3C//DTD XHTML 1.0
  Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/
  xhtml1frameset.dtd">

Document structure

```
<?xml version="1.0" encoding="ISO-8859-15"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
"http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en">

<head>
</head>

<title>Home Page</title>

<body>
<p>Hallo!</p>
</body>
</html>
```

Verification

- Verification of the code can be done locally or on-line
  - You can use the W3C Validator on the web
  - You can use tidy locally
  - You can use a browser

- Note: browsers use different engines and can have different results!

Verification covers most errors, though finding what the mistake was can be tricky. Verifiers are getting better and better though.

Layout engines

- Trident (MSHTML) → Internet Explorer 4 to 8
- Tasman → Internet Explorer 5 for Mac
- Gecko → All Mozilla software, including Firefox; Galeon; Flock; also Epiphany
- WebKit → Safari; Shiira; iCab 4; Epiphany; Adobe Air; Google Chrome; Midori
- KHTML→ Konqueror
- Presto → Opera; Nintendo DS Browser; Internet Channel; future Adobe Systems products
- iCab → iCab 1-3
- Prince XML → Prince XML
- Amaya → Amaya

Suggested Reading/Watching about XML

- IBM technical Series
  - An Introduction to XML: The Basics Part 1 http://www.youtube.com/watch?v=Q0k5ySZGPBc
  - XML and Web 2.0 Part http://www.youtube.com/watch?v=16q5bbeO3xI&feature=related
  - An Introduction to XML Managing XML Data Part 3 http://www.youtube.com/watch?v=R63g3h5f5oA&feature=related

Suggested Reading/Watching about XML

- What is XML good for:
  - http://www.youtube.com/watch?v=CDawM8F-fqY&feature=related
- Just Enough XML to Survive
  - http://www.youtube.com/watch?v=clJcs2-UB40&feature=related
- Learn XML in 4 Minutes
  - http://www.youtube.com/watch?v=saQK7SlFiho&feature=related

Suggested Reading/Watching about HTML/XTML

- Tutorial 1:
  - HTML Document http://www.youtube.com/watch?v=UfBvTsF3fqU
- Tutorial 2:
  - HTML Header http://www.youtube.com/watch?v=jW1JmeLBWwk&feature=channel
- Tutorial 3:
  - HTML Horizontal Rule http://www.youtube.com/watch?v=Xi6XBQPym2I&feature=channel
- Tutorial 4:
  - HTML Elements http://www.youtube.com/watch?v=UwD-cVGbTAM&feature=channel
- Tutorial 5:
  - HTML Colors http://www.youtube.com/watch?v=H8b5zZ0LscU&feature=channel
- Tutorial 6:
  - HTML Tables http://www.youtube.com/watch?v=4Us2q5eSWNc&feature=channel