

XML

Introduction

- XML stands for eXtensible Markup Language.
- XML was designed to store and transport data.
- XML was designed to be both human- and machine-readable.
- XML is used to describe the structure of document not the way that it is presented
- XML was designed to be self-descriptive
- XML is a W3C Recommendation in February 1998.
- XML Does Not DO Anything

Why Study XML?

- XML plays an important role in many different IT systems.
- XML is often used for distributing data over the Internet.
- It is important (for all types of software developers!) to have a good understanding of XML.
- XML stores data in plain text format. This provides a software- and hardware-independent way of storing, transporting, and sharing data.
- XML also makes it easier to expand or upgrade to new operating systems, new applications, or new browsers, without losing data.
- With XML, data can be available to all kinds of "reading machines" like people, computers, voice machines, news feeds, etc.

Advantages of XML

- XML is text (Unicode) based.
 - Takes up less space.
 - Can be transmitted efficiently.
- One XML document can be displayed differently in different media.
 - Html, video, CD, DVD,
 - You only have to change the XML document in order to change all the rest.
- XML documents can be modularized. Parts can be reused.

Example of an XML Document

```
<?xml version="1.0"/>
```

```
<address>
```

```
  <name>Alice Lee</name>
```

```
  <email>alee@gmail.com</email>
```

```
  <phone>212-346-1234</phone>
```

```
  <birthday>1985-03-22</birthday>
```

```
</address>
```

The Difference Between XML and HTML

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

Difference Between HTML and XML

- HTML tags have a fixed meaning and browsers know what it is.
- XML tags are different for different applications, and users know what they mean.
- HTML tags are used for display.
- XML tags are used to describe documents and data.

XML Rules

- Tags are enclosed in angle brackets.
- Tags come in pairs with start-tags and end-tags.
- Tags must be properly nested.
 - **<name><email>...</name></email> is not allowed.**
 - **<name><email>...</email><name> is.**
- Tags that do not have end-tags must be terminated by a '/'.
 - **
** is an html example.

More XML Rules

- Tags are case sensitive.
 - `<address>` is not the same as `<Address>`
- XML in any combination of cases is not allowed as part of a tag.
- Tags may not contain '`<`' or '`&`'.
- Tags follow Java naming conventions, except that a single colon and other characters are allowed. They must begin with a letter and may not contain white space.
- Documents must have a single *root* tag that begins the document.

XML Naming Rules

- XML elements must follow these naming rules:
- Element names are case-sensitive
- Element names must start with a letter or underscore
- Element names cannot start with the letters xml (or XML, or Xml, etc)
- Element names can contain letters, digits, hyphens, underscores, and periods
- Element names cannot contain spaces
- Any name can be used, no words are reserved (except xml).

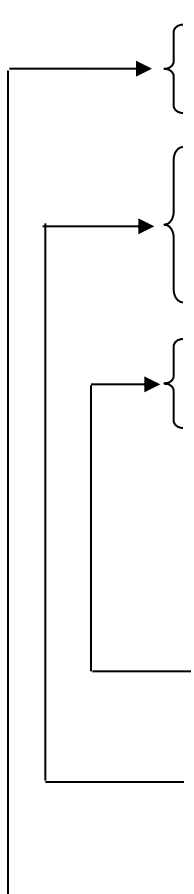
Syntax and Structure

Components of an XML Document

- Elements
 - Each element has a beginning and ending tag
 - `<TAG_NAME> . . . </TAG_NAME>`
 - Elements can be empty (`<TAG_NAME />`)
- Attributes
 - Describes an element; e.g. data type, data range, etc.
 - Can only appear on beginning tag
- Processing instructions
 - Encoding specification (Unicode by default)
 - Namespace declaration
 - Schema declaration

Syntax and Structure

Components of an XML Document



```
<?xml version="1.0" ?>  
<?xml-stylesheet type="text/xsl" href="template.xsl"?>  
<ROOT>  
  <ELEMENT1><SUBELEMENT1 /><SUBELEMENT2 /></ELEMENT1>  
  <ELEMENT2> </ELEMENT2>  
  <ELEMENT3 type='string'> </ELEMENT3>  
  <ELEMENT4 type='integer' value='9.3'> </ELEMENT4>  
</ROOT>
```

The diagram illustrates the components of an XML document. On the left, three labels are connected by lines to specific parts of the XML code block: 'Prologue (processing instructions)' points to the first two lines, 'Elements' points to the opening and closing tags of the root element, and 'Elements with Attributes' points to the opening and closing tags of the root element's children.

Elements with Attributes

Elements

Prologue (processing instructions)

Syntax and Structure

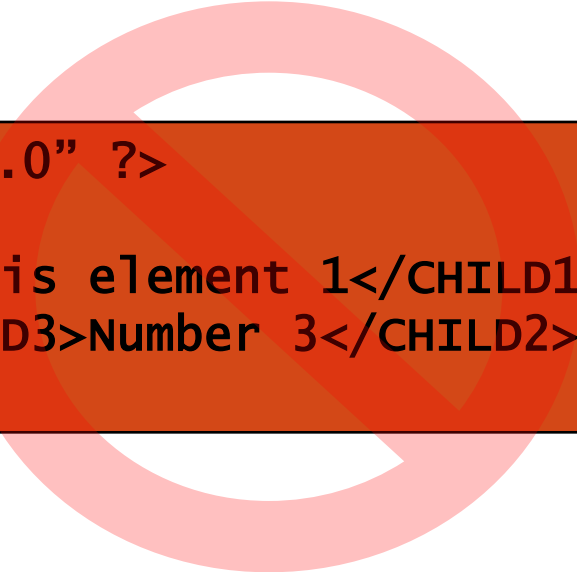
Rules For Well-Formed XML

- There must be one, and only one, root element
- Sub-elements must be properly nested
 - A tag must end within the tag in which it was started
- Attributes are optional
 - Defined by an optional schema
- Attribute values must be enclosed in “” or ‘ ’
- Processing instructions are optional
- XML is case-sensitive
 - `<tag>` and `<TAG>` are not the same type of element

Syntax and Structure

Well-Formed XML?

- No, CHILD2 and CHILD3 do not nest properly

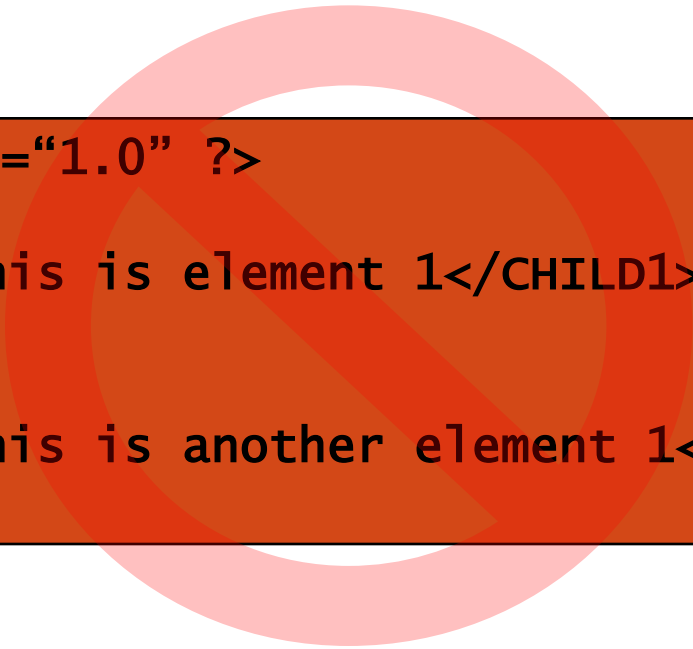


```
<xml? version="1.0" ?>  
<PARENT>  
  <CHILD1>This is element 1</CHILD1>  
  <CHILD2><CHILD3>Number 3</CHILD2></CHILD3>  
</PARENT>
```

Syntax and Structure

Well-Formed XML?

- No, there are two root elements



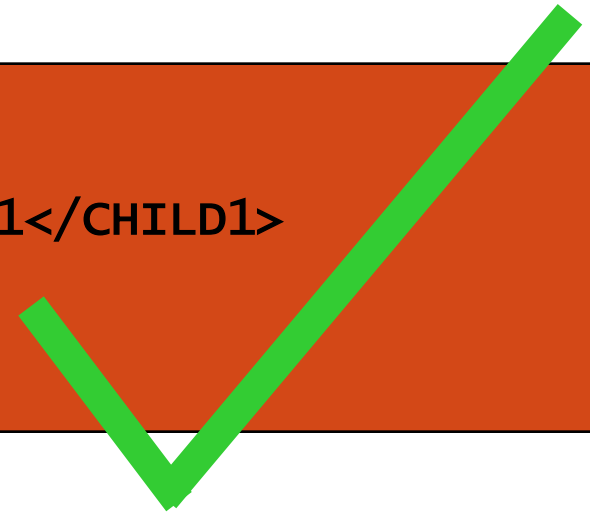
```
<xml? version="1.0" ?>  
<PARENT>  
  <CHILD1>This is element 1</CHILD1>  
</PARENT>  
<PARENT>  
  <CHILD1>This is another element 1</CHILD1>  
</PARENT>
```

Syntax and Structure

Well-Formed XML?

- Yes

```
<xml? version="1.0" ?>  
<PARENT>  
  <CHILD1>This is element 1</CHILD1>  
  <CHILD2/>  
  <CHILD3></CHILD3>  
</PARENT>
```



Syntax and Structure

An XML Document

```
<?xml version='1.0'?>
<bookstore>
  <book genre='autobiography' publicationdate='1981'
        ISBN='1-861003-11-0'>
    <title>The Autobiography of Benjamin Franklin</title>
    <author>
      <first-name>Benjamin</first-name>
      <last-name>Franklin</last-name>
    </author>
    <price>8.99</price>
  </book>
  <book genre='novel' publicationdate='1967' ISBN='0-201-63361-2'>
    <title>The Confidence Man</title>
    <author>
      <first-name>Herman</first-name>
      <last-name>Melville</last-name>
    </author>
    <price>11.99</price>
  </book>
</bookstore>
```

Well-Formed Documents

- An XML document is said to be well-formed if it follows all the rules.
- An XML parser is used to check that all the rules have been obeyed.
- Recent browsers such as Internet Explorer 5 and Netscape 7 come with XML parsers.
- Parsers are also available for free download over the Internet. One is Xerces, from the Apache open-source project.
- Java 1.4 also supports an open-source parser.

XML Example Revisited

```
<?xml version="1.0"/>
```

```
<address>
```

```
  <name>Alice Lee</name>
```

```
  <email>alee@gmail.com</email>
```

```
  <phone>212-346-1234</phone>
```

```
  <birthday>1985-03-22</birthday>
```

```
</address>
```

- Markup for the data aids understanding of its purpose.
- A flat text file is not nearly so clear.

Alice Lee

alee@gmail.com

212-346-1234

1985-03-22

- The last line looks like a date, but what is it for?

Expanded Example

<?xml version = “1.0” ?>

<address>

 <name>

 <first>Alice</first>

 <last>Lee</last>

 </name>

 <email>alee@gmail.com</email>

 <phone>123-45-6789</phone>

 <birthday>

 <year>1983</year>

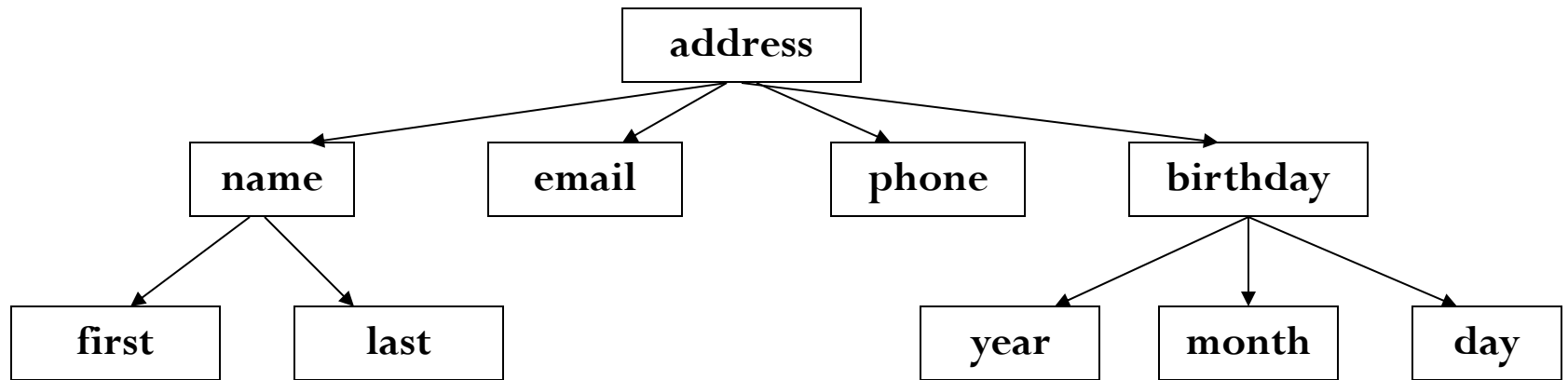
 <month>07</month>

 <day>15</day>

 </birthday>

</address>

XML Files are Trees



XML Trees

- An XML document has a single root node.
- The tree is a general ordered tree.
 - A parent node may have any number of children.
 - Child nodes are ordered, and may have siblings.
- Preorder traversals are usually used for getting information out of the tree.

Validity

- A well-formed document has a tree structure and obeys all the XML rules.
- A particular application may add more rules in either a DTD (document type definition) or in a schema.
- Many specialized DTDs and schemas have been created to describe particular areas.
- These range from disseminating news bulletins (RSS) to chemical formulas.
- DTDs were developed first, so they are not as comprehensive as schema.

Document Type Definitions

- A DTD describes the tree structure of a document and something about its data.
- An XML document with correct syntax is called "Well Formed".
- An XML document validated against a DTD is both "Well Formed" and "Valid".
- There are two data types, PCDATA and CDATA.
 - PCDATA is parsed character data.
 - CDATA is character data, not usually parsed.
- A DTD determines how many times a node may appear, and how child nodes are ordered.

Cont...

- The purpose of a DTD is to define the structure of an XML document. It defines the structure with a list of legal elements:

Syntax

- Basic syntax of a DTD is as follows –
- `<!DOCTYPE element DTD identifier [declaration1 declaration2]>`
- In the above syntax,
- The **DTD** starts with `<!DOCTYPE` delimiter.
- An **element** tells the parser to parse the document from the specified root element.
- **DTD identifier** is an identifier for the document type definition, which may be the path to a file on the system or URL to a file on the internet. If the DTD is pointing to external path, it is called **External Subset**.
- The **square brackets []** enclose an optional list of entity declarations called *Internal Subset*.

Internal DTD

- A DTD is referred to as an internal DTD if elements are declared within the XML files.
- **Syntax**
- Following is the syntax of internal DTD –
- `<!DOCTYPE root-element [element-declarations]>`
- where *root-element* is the name of root element and *element-declarations* is where you declare the elements.

Example for Internal DTD

- `<?xml version = "1.0" encoding = "UTF-8" standalone = "yes" ?>`
- `<!DOCTYPE address`
- `[<!ELEMENT address (name,company,phone)>`
- `<!ELEMENT name (#PCDATA)>`
- `<!ELEMENT company (#PCDATA)>`
- `<!ELEMENT phone (#PCDATA)>]>`
- `<address>`
- `<name>Tanmay Patil</name>`
- `<company>TutorialsPoint</company>`
- `<phone>(011) 123-4567</phone>`
- `</address>`

External DTD

- In external DTD elements are declared outside the XML file. They are accessed by specifying the system attributes which may be either the legal *.dtd* file or a valid URL.
- **Syntax**
- Following is the syntax for external DTD –
- `<!DOCTYPE root-element SYSTEM "file-name">` where *file-name* is the file with *.dtd* extension.

- `<?xml version = "1.0" encoding = "UTF-8" standalone = "no" ?>`
- `<!DOCTYPE address SYSTEM "address.dtd">`
- `<address>`
- `<name>Tanmay Patil</name>`
`<company>TutorialsPoint</company>`
- `<phone>(011) 123-4567</phone>`
- `</address>`

External DTD (address.dtd)

- `<!ELEMENT address (name,company,phone)>`
`<!ELEMENT name (#PCDATA)>`
- `<!ELEMENT company (#PCDATA)>`
- `<!ELEMENT phone (#PCDATA)>`

External DTD for address Example

<!ELEMENT address (name, email, phone, birthday)>

<!ELEMENT name (first, last)>

<!ELEMENT first (#PCDATA)>

<!ELEMENT last (#PCDATA)>

<!ELEMENT email (#PCDATA)>

<!ELEMENT phone (#PCDATA)>

<!ELEMENT birthday (year, month, day)>

<!ELEMENT year (#PCDATA)>

<!ELEMENT month (#PCDATA)>

<!ELEMENT day (#PCDATA)>

Cont...

- !ELEMENT address defines that the address element must contain the elements: " name, email, phone, birthday “
- !ELEMENT name defines that the name element must contain the elements: " first,last”
- !ELEMENT first defines the from element to be of type "#PCDATA“
- !ELEMENT last defines the from element to be of type "#PCDATA“
- !ELEMENT email defines the heading element to be of type "#PCDATA"
- !ELEMENT phone defines the body element to be of type "#PCDATA"
- !ELEMENT birthday defines that the birthday element must contain the elements: " year,month,day“

Cont...

- !ELEMENT year defines the body element to be of type "#PCDATA"
- !ELEMENT month defines the body element to be of type "#PCDATA"
- !ELEMENT day defines the body element to be of type "#PCDATA"

Namespaces

- XML Namespaces provide a method to avoid element name conflicts.
- XML data has to be exchanged between organizations
- Same tag name may have different meaning in different organizations, causing confusion on exchanged documents
- Specifying a unique string as an element name avoids confusion
- Better solution: use **unique-name:element-name**
- Avoid using long unique names all over document by using XML Namespaces

Namespaces

- In XML, element names are defined by the developer. This often results in a conflict when trying to mix XML documents from different XML applications.
- This XML carries HTML table information:
- `<table>`
 - `<tr>`
 - `<td>Apples</td>`
 - `<td>Bananas</td>`
 - `</tr>`
 - `</table>`

Namespaces

- This XML carries information about a table (a piece of furniture):
- `<table>`
 - `<name>African Coffee Table</name>`
 - `<width>80</width>`
 - `<length>120</length>`
 - `</table>`
- If these XML fragments were added together, there would be a name conflict. Both contain a `<table>` element, but the elements have different content and meaning.
- A user or an XML application will not know how to handle these differences.

Namespaces

- **Solving the Name Conflict Using a Prefix**
- Name conflicts in XML can easily be avoided using a name prefix.
- This XML carries information about an HTML table, and a piece of furniture:

Namespaces

- `<h:table>`

- `<h:tr>`

- `<h:td>Apples</h:td>`

- `<h:td>Bananas</h:td>`

- `</h:tr>`

- `</h:table>`

- `<f:table>`

- `<f:name>African Coffee Table</f:name>`

- `<f:width>80</f:width>`

- `<f:length>120</f:length>`

- `</f:table>`

- **XML Namespaces - The xmlns Attribute**
- When using prefixes in XML, a **namespace** for the prefix must be defined.
- The namespace can be defined by an **xmlns** attribute in the start tag of an element.
- The namespace declaration has the following syntax.
`xmlns:prefix="URI"`.

Namespace Example

- <root>

```
<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="https://www.w3schools.com/furniture">
```

```
<f:name>African Coffee Table</f:name>
```

```
<f:width>80</f:width>
```

```
<f:length>120</f:length>
```

```
</f:table>
```

```
</root>
```

Namespaces

- Part of XML's extensibility
- Allow authors to differentiate between tags of the same name (using a prefix)
 - Frees author to focus on the data and decide how to best describe it
 - Allows multiple XML documents from multiple authors to be merged
- Identified by a URI (Uniform Resource Identifier)
 - When a URL is used, it does NOT have to represent a live server

Syntax and Structure

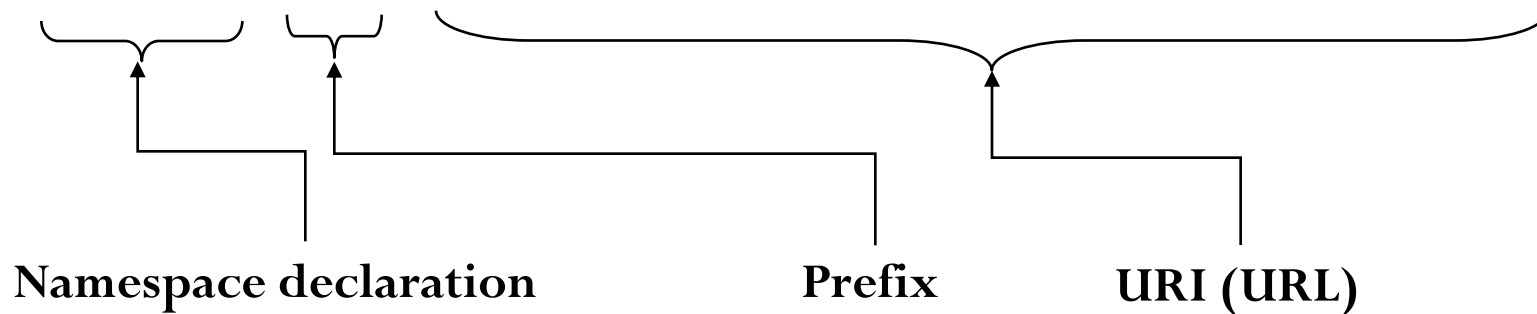
Namespaces: Declaration

Namespace declaration examples:

```
xmlns: bk = "http://www.example.com/bookinfo/"
```

```
xmlns: bk = "urn:mybookstuff.org:bookinfo"
```

```
xmlns: bk = "http://www.example.com/bookinfo/"
```



Syntax and Structure

Namespaces: Examples

```
<BOOK xmlns:bk="http://www.bookstuff.org/bookinfo">  
  <bk:TITLE>All About XML</bk:TITLE>  
  <bk:AUTHOR>Joe Developer</bk:AUTHOR>  
  <bk:PRICE currency='US Dollar'>19.99</bk:PRICE>
```

```
<bk:BOOK xmlns:bk="http://www.bookstuff.org/bookinfo"  
  xmlns:money="urn:finance:money">  
  <bk:TITLE>All About XML</bk:TITLE>  
  <bk:AUTHOR>Joe Developer</bk:AUTHOR>  
  <bk:PRICE money:currency='US Dollar'>  
    19.99</bk:PRICE>
```

Syntax and Structure

Namespaces: Default Namespace

- An XML namespace declared without a prefix becomes the default namespace for all sub-elements
- All elements without a prefix will belong to the default namespace:

```
<BOOK xmlns="http://www.bookstuff.org/bookinfo">  
  <TITLE>All About XML</TITLE>  
  <AUTHOR>Joe Developer</AUTHOR>
```

XML Schema

What is an XML Schema?

- An XML Schema describes the structure of an XML document.
- The XML Schema language is also referred to as XML Schema Definition (XSD).
- XML Schema documents are used to define and validate the content and structure of XML data.
- XML Schema was originally proposed by Microsoft, but became an official W3C recommendation in May 2001

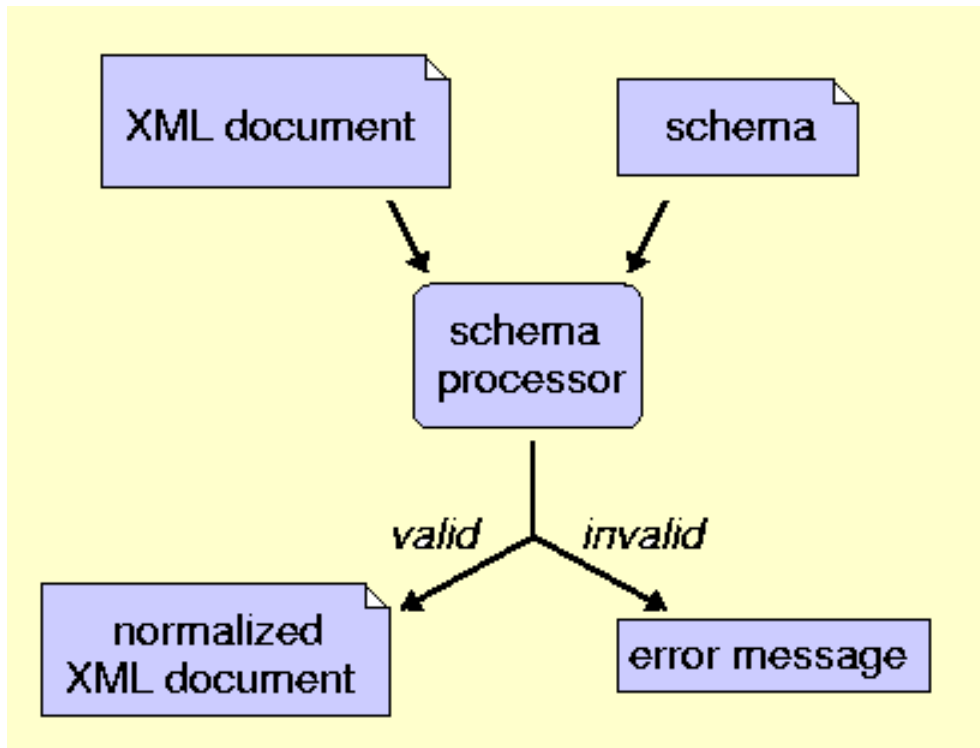
Why XML Schema?

- The purpose of an XML Schema is to define the legal building blocks of an XML document:
- the elements and attributes that can appear in a document
- the number of (and order of) child elements
- data types for elements and attributes
- default and fixed values for elements and attributes

Why Learn XML Schema?

- In the XML world, hundreds of standardized XML formats are in daily use.
- Many of these XML standards are defined by XML Schemas.
- XML Schema is an XML-based (and more powerful) alternative to DTD.

Why Schema?



Schema Workflow

DTD vs Schema

No.	DTD	XSD
1)	DTD stands for Document Type Definition .	XSD stands for XML Schema Definition.
2)	DTDs are derived from SGML syntax.	XSDs are written in XML.
3)	DTD doesn't support datatypes .	XSD supports datatypes for elements and attributes.
4)	DTD doesn't support namespace .	XSD supports namespace .
5)	DTD doesn't define order for child elements.	XSD defines order for child elements.
6)	DTD is not extensible .	XSD is extensible .
7)	DTD is not simple to learn .	XSD is simple to learn because you don't need to learn new language.
8)	DTD provides less control on XML structure.	XSD provides more control on XML structure.

DTD versus Schema

Limitations of DTD

- No constraints on character data
- Not using XML syntax
- No support for namespace
- Very limited for reusability and extensibility

Advantages of Schema

- Syntax in XML Style
- Supporting Namespace and import/include
- More data types
- Able to create complex data type by inheritance
- Inheritance by extension or restriction
- More ...

XML Schema Definition (XSD) elements

- An element is the building block of an XML document and is defined within the XSD.
- The three types of XSD schema elements can be defined as:
 - Simple
 - Complex
 - Global
- The choice depends on whether the element is a parent element or a leaf element. An element can be defined in the XSD as
 - *<xs:element name = "x" type = "y" />.*

XSD Simple Elements

- A simple element is an XML element that contains only text. It cannot contain any other elements or attributes.
- `<xs:element name="xxx" type="yyy" />`
- where xxx is the name of the element and yyy is the data type of the element.
- XML Schema has a lot of built-in data types.
- The most common types are:
 - xs:string
 - xs:decimal
 - xs:integer
 - xs:boolean
 - xs:date
 - xs:time

XSD Complex Elements

- A complex element contains other elements and/or attributes.
- A complex element is an XML element that contains other elements and/or attributes.
- There are four kinds of complex elements:
 - empty elements
 - elements that contain only other elements
 - elements that contain only text
 - elements that contain both other elements and text
- `<employee>`
 `<firstname>John</firstname>`
 `<lastname>Smith</lastname>`
 `</employee>`
- A complex XML element, "employee", which contains only other elements:

XML Schemas use XML Syntax

- Another great strength about XML Schemas is that they are written in XML.
- You don't have to learn a new language
- You can use your XML editor to edit your Schema files
- You can use your XML parser to parse your Schema files
- You can manipulate your Schema with the XML DOM
- You can transform your Schema with XSLT
- XML Schemas are extensible, because they are written in XML.
- With an extensible Schema definition you can:
 - Reuse your Schema in other Schemas
 - Create your own data types derived from the standard types
 - Reference multiple schemas in the same document

XSD How To?

- XML documents can have a reference to a DTD or to an XML Schema.

- `<?xml version="1.0"?>`

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

A Reference to an XML Schema

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

# XSD - The <schema> Element

- The <schema> element is the root element of every XML Schema.
- <?xml version="1.0"?>

<xs:schema>

...

...

</xs:schema>

- The <schema> element may contain some attributes. A schema declaration often looks something like this:

- Here are some XML elements:
- `<lastname>Refsnes</lastname>`  
`<age>36</age>`  
`<dateborn>1970-03-27</dateborn>`
- And here are the corresponding simple element definitions:
- `<xs:element name="lastname" type="xs:string" />`  
`<xs:element name="age" type="xs:integer" />`  
`<xs:element name="dateborn" type="xs:date" />`

# Default and Fixed Values for Simple Elements

- Simple elements may have a default value OR a fixed value specified.
- A default value is automatically assigned to the element when no other value is specified.
- In the following example the default value is "red":
- `<xs:element name="color" type="xs:string" default="red"/>`
- A fixed value is also automatically assigned to the element, and you cannot specify another value.
- In the following example the fixed value is "red":
- `<xs:element name="color" type="xs:string" fixed="red"/>`

# XSD Restrictions

- Restrictions are used to define acceptable values for XML elements or attributes. Restrictions on XML elements are called facets.
- The following example defines an element called "age" with a restriction. The value of age cannot be lower than 0 or greater than 120:
- ```
<xs:element name="age">  
  <xs:simpleType>  
    <xs:restriction base="xs:integer">  
      <xs:minInclusive value="0"/>  
      <xs:maxInclusive value="120"/>  
    </xs:restriction>  
  </xs:simpleType>
```



studentSchema.xsd

- `<xs:schema`
 `xmlns:xs="http://www.w3.org/2001/XMLSchema">`
- `<xs:element name="student">`
- `<xs:complexType>`
- `<xs:sequence>`
- `<xs:element name="sname" type="xs:string"/>`
- `<xs:element name="spinno" type="xs:integer"/>`
- `<xs:element name="sbranch" type="xs:string"/>`
- `</xs:sequence>`
- `</xs:complexType>`
- `</xs:element>`
- `</xs:schema>`

Student.xml

- `<student`
 `xmlns:xsi="http://www.w3.org/2001/XMLSchema-`
 `instance"`
- `xsi:noNamespaceSchemaLocation="studentSchema.xsd">`
- `<name>Rajkumar</sname>`
- `<spinno>789654</spinno>`
- `<sbranch>CSE</sbranch>`
- `</student>`

Displaying XML documents with CSS

- There are two ways to applying styles to XML Document
- First one is to create CSS file for XML Document
- Second one is XSLT Style Sheet Technology
- The form of CSS style sheet for XML document is simple
 - List of element name, each followed by a brace-delimited set of the element's CSS attributes.
 - Ex:
to { color:red; font-size:12pt;}
from { color:blue; font-size: 12pt;}
body { color:green; font-size:20pt;}
•

Displaying XML documents with CSS

- The connection between XML document and CSS using **xml-stylesheet** processing instructions
- `<?xml-stylesheet type = "text/css" href = "first.css" ?>`

XSLT Style Sheets

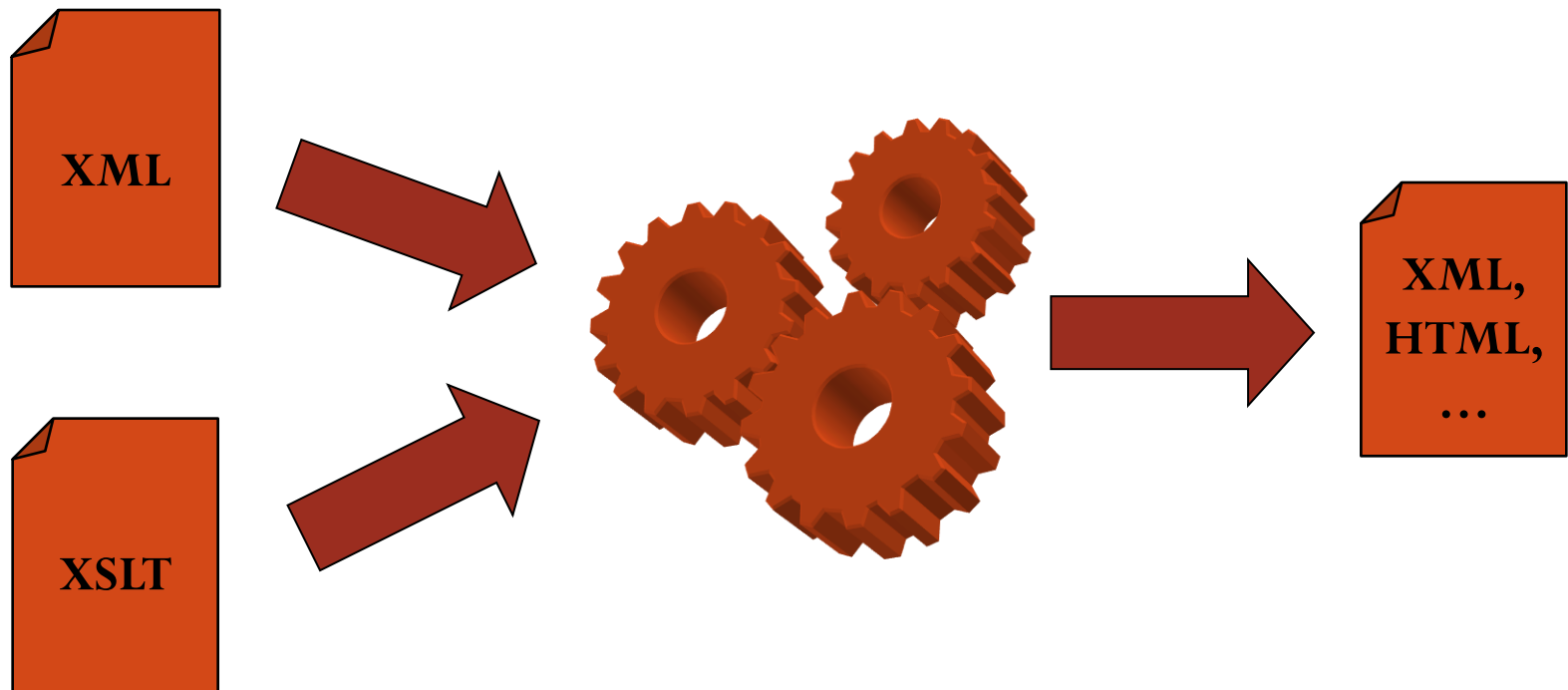
- XSL stands for eXtensible Stylesheet Language
- It is a family of recommendations for XML document transformations and presentation
 - XSLT(eXtensible Stylesheet Language)
 - XPath(XML Path Language)
 - XSL Formatting Objects(XSL-FO)
- XSLT style sheets are used to transform XML document into different forms of formats

XSLT Style Sheets

- XSLT is transform XML document into XHTML documents
- The XSLT standard is given at
- <http://www.w3.org/TR/xslt>
- **2017-06-08: XSLT 3.0 is a W3C Recommendation**
- Expressed as an XML document (.xsl)
- Template rules
 - Patterns match nodes in source document
 - Templates instantiated to form part of result document
- **XSLT model of processing XML data is called the template driven model**

XSLT Style Sheets

- XSLT – a language used to transform XML data into a different form (commonly XML or HTML)



XSLT Style Sheets

- XSLT style sheet must include a processing instruction
- Syntax:
- `<?xml-stylesheet type="text/xsl" href="XSL_Stylesheet_name"?>`
- Ex:
- `<?xml-stylesheet type="text/xsl" href="books.xsl"?>`
- A style sheet document must include at least one template element

XSLT Style Sheets

- A template is include to match the root node of the XML document
- `<xsl:template match = "/">`
- or
- `<xsl:template match = "book-info">`
- The content of an elemet of the XML document is to be copied to the output document using **value-of** element and **select** attribute
- `<xsl:value-of select="Title-of-the-book" />`

XML DOM

- The DOM defines a standard for accessing and manipulating documents:
- *"The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure, and style of a document."*
- All XML elements can be accessed through the XML DOM.
- This code retrieves the text value of the first <title> element in an XML document:
- ```
txt =
xmlDoc.getElementsByTagName("title")[0].childNodes[0].nodeValue;
```



# XML DOM

