The eXtensible Markup Language (XML)

Syllabus

- XML: DTD, XML schemes, Object Models, presenting and using XML, Using XML Processors: DOM and SAX, Dynamic HTML

What is XML ?

- XML stands for EXtensible Markup Language
- XML is a markup language much like HTML
- XML was designed to carry data, not to display data
- XML tags are not predefined. You must define your own tags
- XML is designed to be self-descriptive
- XML is a W3C Recommendation

More about XML...

- It is defined in the XML 1.0 specification, which is developed by the W3C (World Wide Web Consortium). XML provides a standard way, which is also simple, to encode data and text such that the content could be exchanged across driver hardware, operating systems and applications with little human intervention.

Similarities between HTML and XML?

- 1. Both are languages of web.
  2. Both are markup languages.
  3. Both are originated from SGML. [Standardized General Markup Language]
  4. Tags are basic building blocks of both HTML and XML documents.

Comparisons

HTML	XML
HTML is an abbreviation for HyperText Markup Language.	XML stands for eXtensible Markup Language.
HTML was designed to display data with focus on how data looks.	XML was designed to be a software and hardware independent tool used to transport and store data, with focus on what data is.
HTML is a markup language itself.	XML provides a framework for defining markup languages.
HTML is a presentation language.	XML is neither a programming language nor a presentation language.
HTML is case insensitive.	XML is case sensitive.
HTML is used for designing a web-page to be rendered on the client side.	XML is used basically to transport data between the application and the database.
HTML has it own predefined tags.	While what makes XML flexible is that custom tags can be defined and the tags are invented by the author of the xml document.

Comparisons

HTML	XML
HTML is not strict if the user does not use the closing tags.	XML makes it mandatory for the user the close each tag that has been used.
HTML does not preserve white space.	XML preserves white space.
HTML is about displaying data, hence static.	XML is about carrying information, hence dynamic.
HTML stylesheet for HTML are optional.	XML stylesheet for

Example of HTML :

```
<html>
<body>
<h3> An example of HTML </h3>
Hello world !
</body>
</html>
```

Here, each & every tag is predefined.

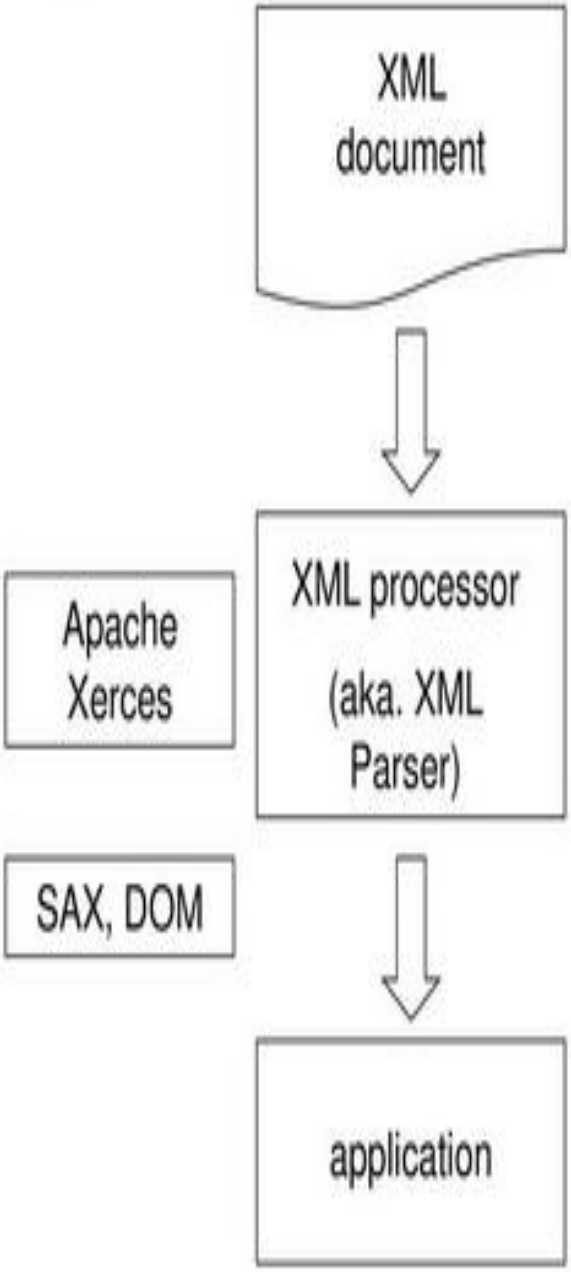
Example of XML :

```
<?XML version="1.0" ?>
<email>
<to>Minal</to>
<from>Tulshiram</from>
<subject>About holiday</subject>
<body>I have a holiday tomorrow</body>
</email>
```

Here, all tags are user defined except the very first tag <?XML version="1.0" ?>.

XML in General Application

- XML by itself does not do anything
- XML just describes the structure of the data
- Other applications parse XML and use it

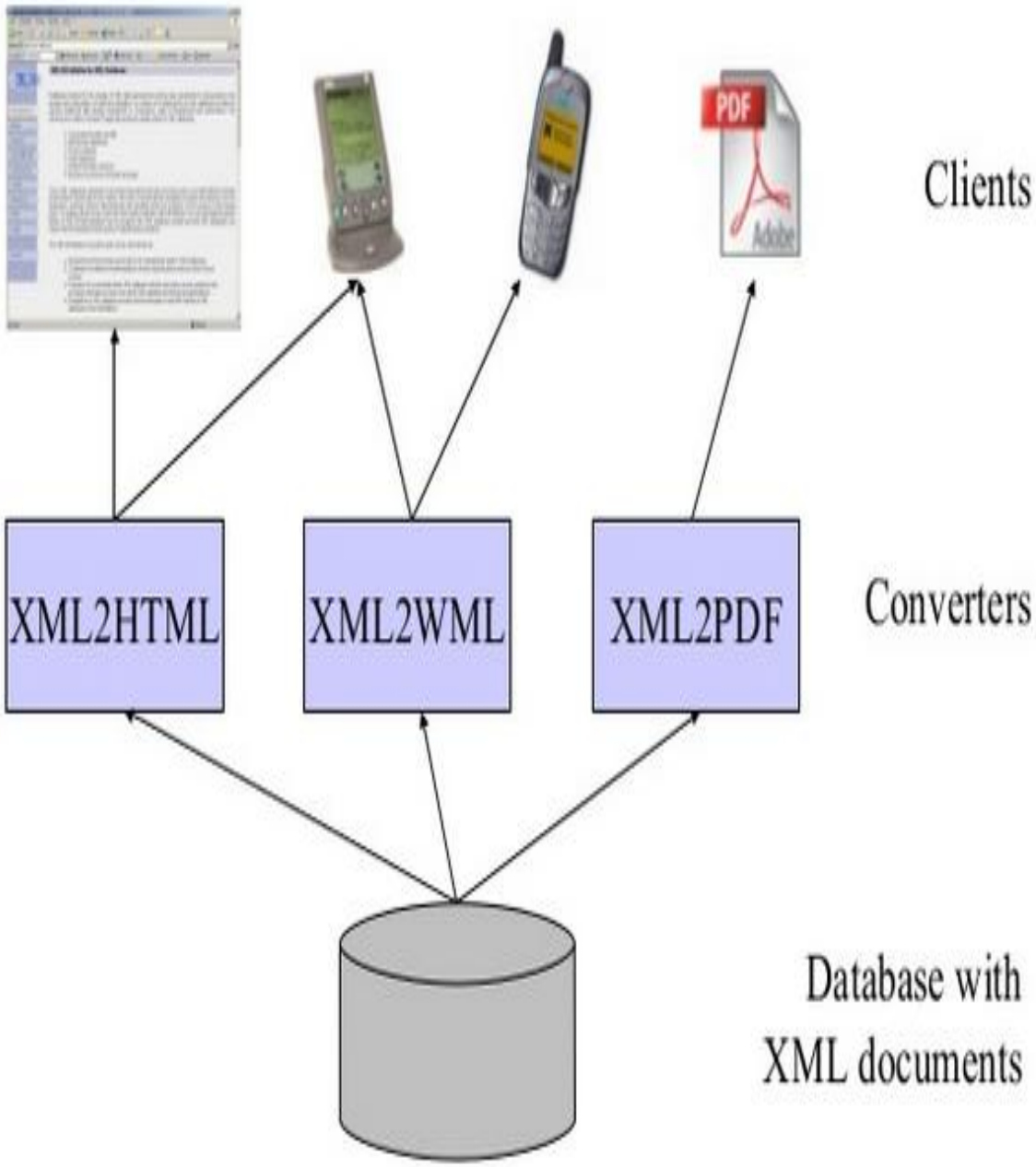


- A similar approach is used for formats (event user-defined format); so, what is the advantages of XML?!!!
  - *XML is standard*
  - *Available XML tools & technologies*

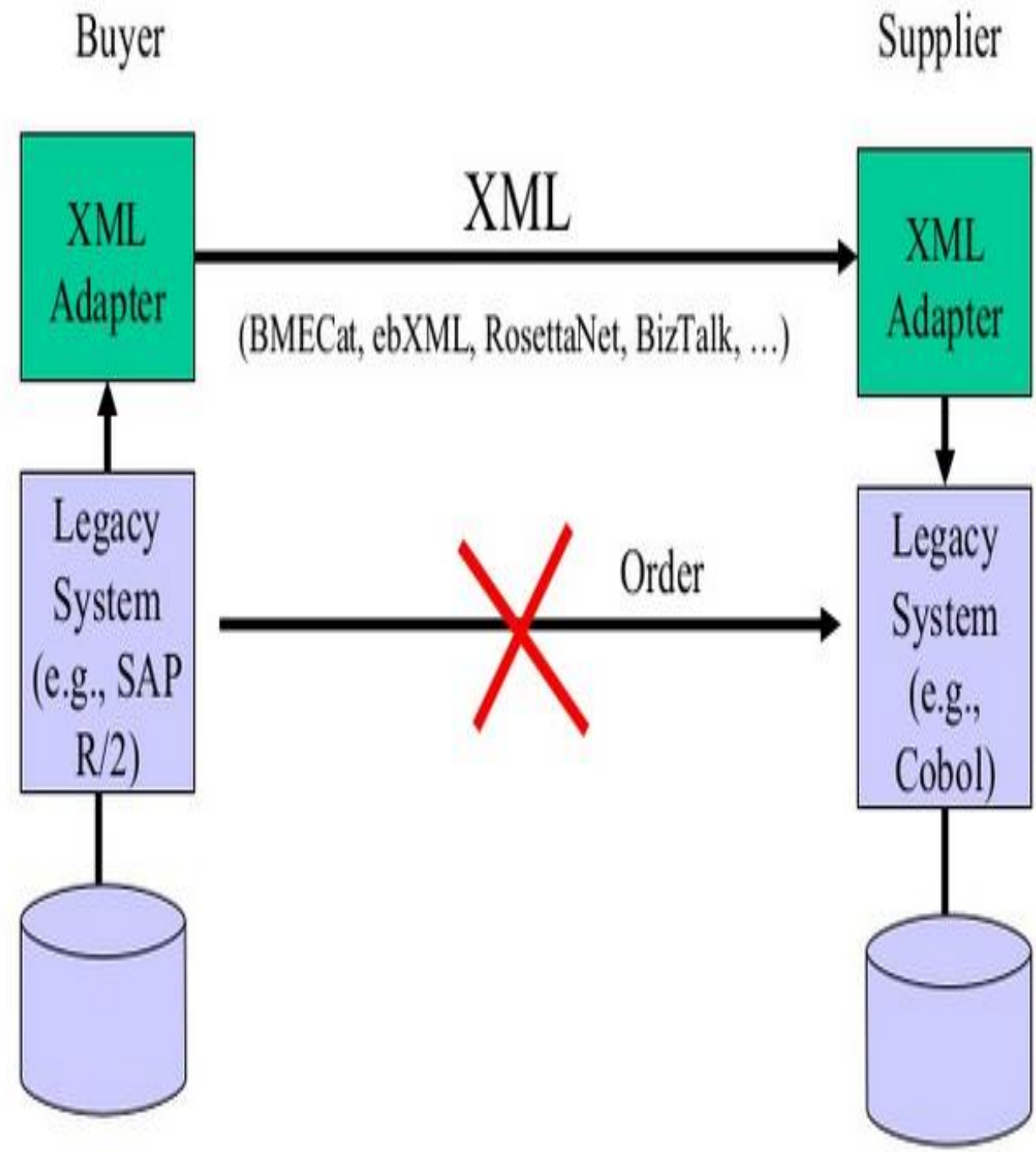
XML Technology Components

- Data structure (tree) representation
  - XML document (a text file)
- Validation & Conformance
  - Document Type Definition (DTD) or XML Schema
- Element access & addressing
  - XPath, DOM
- Display and transformation
  - XSLT or CSS
- Programming, Database, Query, ...

App. Scenario 1: Content Mgt.



App. Scenario 2: Data Exchange



XML Document

What's in an XML document?

- Elements
- Attributes
- plus some other details

A Simple XML Document

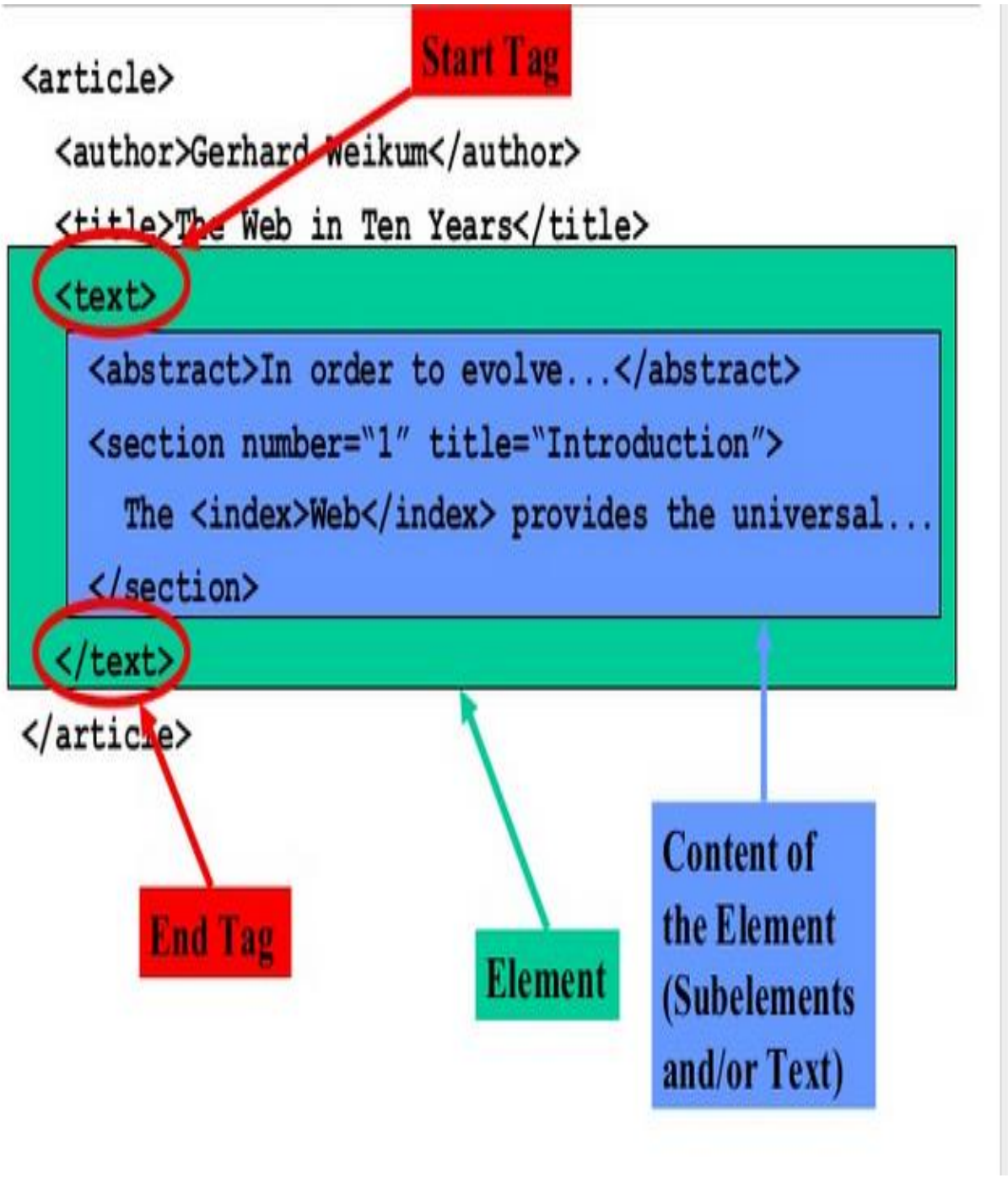
```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the universal...
    </section>
  </text>
</article>
```

A Simple XML Document

```
<article>
  <author>Gernard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the universal...
    </section>
  </text>
</article>
```

Freely definable tags

A Simple XML Document



A Simple XML Document

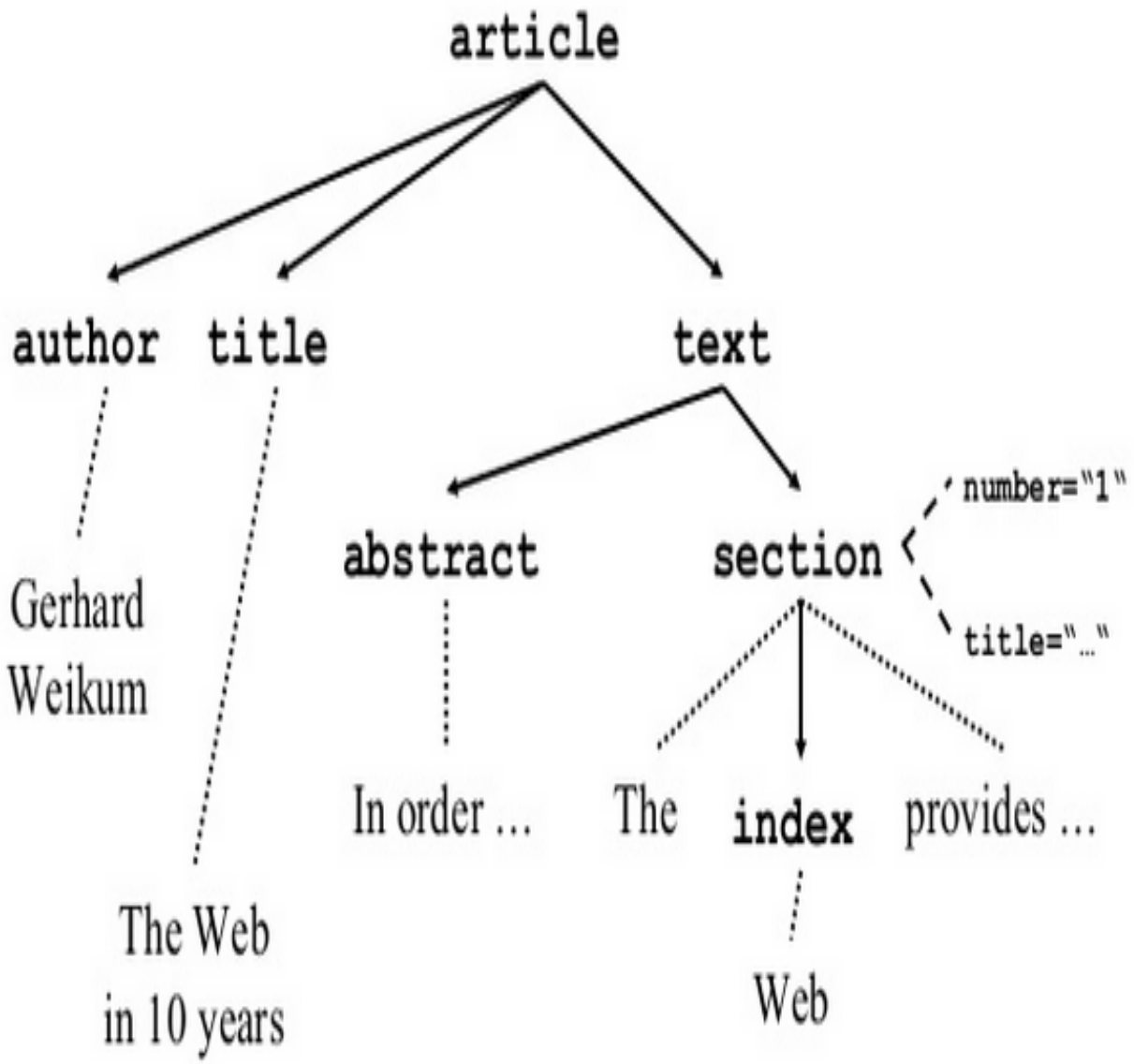
```
<article>
  <author>Gerhard Weikum</author>
  <title>The Web in Ten Years</title>
  <text>
    <abstract>In order to evolve...</abstract>
    <section number="1" title="Introduction">
      The <index>Web</index> provides the universal...
    </section>
  </text>
</article>
```

Attributes with name and value

Elements of XML Document

- (Freely definable) **tags**: `article`, `title`, `author`
  - with start tag: `<article>` etc.
  - and end tag: `</article>` etc.
- **Elements**: `<article> ... </article>`
- Elements have a **name** (`article`) and a **content** (...)
- Elements may be nested.
- Elements may be empty: `<this_is_empty/>`
- Element content is typically parsed character data (PCDATA), i.e., strings with special characters, and/or nested elements (*mixed content* if both).
- Each XML document has exactly one root element and forms a tree.
- Elements with a common parent are ordered.

XML Documents as ordered Trees



Applications of XML

- Database applications
- Document Mark-up(with HTML)
- Mathematical Mark-up language(MATHML)
- Messaging b/w different business platforms
- Channel definition Format (CDF)
- Metacontent definition
- Platform for Internet Context Selection (PICS)
- Platform for Privacy References Syntax Specification (P3P)
- Resource Description Format (RDF)
- Scaleable Vector Graphics (SVG)
- Synchronized Multemedia Integration Language (SMIL)

XML DTD

- An XML document may have an *optional* DTD, which defines the document's grammar.

  Since the DTD defines the XML document's grammar, we can use an XML parser to check that if an XML document conforms to the grammar defined by the DTD.

- The purpose of a DTD is to define the legal building blocks of an XML document.

- Terminology for XML:

  -- well-formed: if tags are correctly closed.

  -- valid: if it has a DTD and conforms to it.

Validation is useful in data exchange.

- A DTD can be declared inside the XML document, or as an external reference.
- 1) ***Internal DTD***

  This is a example of a simple XML document with an internal DTD:

```
<?xml version="1.0" standalone="yes"?>
<!DOCTYPE tutorials [
<!ELEMENT tutorials (tutorial)+>
<!ELEMENT tutorial (name,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST tutorials type CDATA #REQUIRED>
]>
<tutorials type="web">
  <tutorial>
    <name>XML Tutorial</name>
    <url>http://www.xyz/xml/tutorial</url>
  </tutorial>
  <tutorial>
    <name>HTML Tutorial</name>
    <url>http://www.xx.com/html/tutorial</url>
  </tutorial>
</tutorials>
```

The DTD a is interpreted like this:

!DOCTYPE tutorials defines that the root element of this document is tutorials

!ELEMENT tutorials defines that the tutorials element contains minimum one occurrence of child element.

!ELEMENT tutorial defines that the tutorials element must contain two element name,url.

!ELEMENT name defines the to element to be of type "#PCDATA"

!ELEMENT url defines the from element to be of type "#PCDATA"

EXTERNAL DTD

- If the DTD is declared in an external file, the <!DOCTYPE> definition must contain a reference to the DTD file:
- **The keyword SYSTEM indicates that it's a private DTD**

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE tutorials SYSTEM "tutorials.dtd">
<tutorials type="web">
  <tutorial>
    <name>XML Tutorial</name>
    <url>http://www.q.com/xml/tutorial</url>
  </tutorial>
  <tutorial>
    <name>HTML Tutorial</name>
    <url>http://www.ii.com/html/tutorial</url>
  </tutorial>
</tutorials>
```

```
<!ELEMENT tutorials (tutorial)+>
<!ELEMENT tutorial (name,url)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT url (#PCDATA)>
<!ATTLIST tutorials type CDATA #REQUIRED>
```

DTD - XML Building Blocks

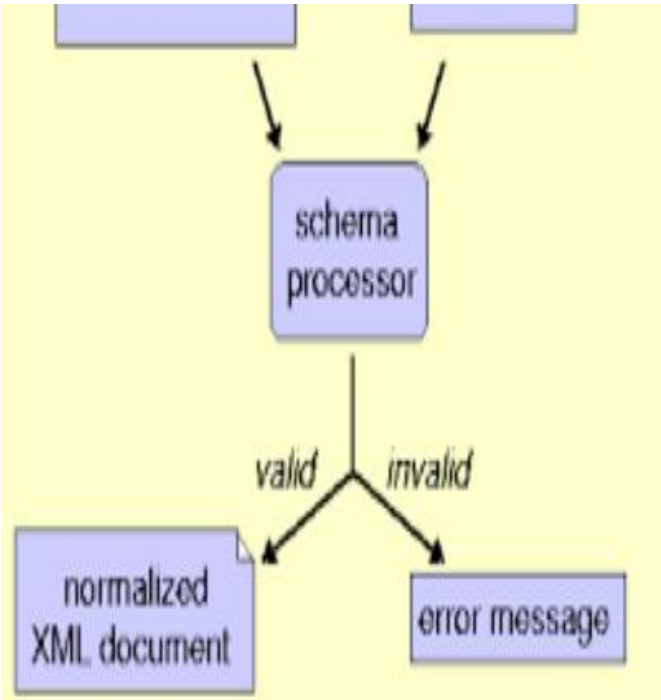
- The main building blocks of both XML and HTML documents are elements.
- Seen from a DTD point of view, all XML documents are made up by the following building blocks:
- Elements
- Attributes
- Entities
- PCDATA
- CDATA
- Entities

Entity References	Character
<	<
>	>
&	&
"	"
'	'

- Some characters have a special meaning in XML, like the less than sign (<) that defines the start of an XML tag.
- Most of you know the HTML entity: " ". This "no-breaking-space" entity is used in HTML to insert an extra space in a document. Entities are expanded when a document is parsed by an XML parser.
- The following entities are predefined in XML:

XMLSCHEMA

- It is also called as XSD(Xml Schema Definition).
- It is used to describe and validate the structure and content of xml data
- Xml schema defines the elements, attributes, and data types.
- <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">



More about XML schema ...

- XML schema is a language which is used for expressing constraint about XML documents. There are so many schema languages which are used now a days for example Relax- NG and XSD (XML schema definition).
- An XML schema is used to define the structure of an XML document. It is like DTD but provides more control on XML structure.

XML Schema Example

Let's create a schema file.

employee.xsd

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://www.javatpoint.com"
xmlns="http://www.javatpoint.com"
elementFormDefault="qualified">

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

</xs:schema>
```

ELEMENTS

- Elements are building blocks of xml document. An element can be defined with in an XSD.
- <xs:element name="x" type="y"/>
- Three types of elements: 1.Simple Type 2.Complex Type 3.Global Type

1. **Simple Type**

   These are used only in the context of text. They are

- ❑ **xs:integer**
- ❑ **xs:boolean**
- ❑ **xs:string**
- ❑ **xs:date**

➢ <xs:element name="phone_number" type="xs:int"/>

- *name* is the name of the element

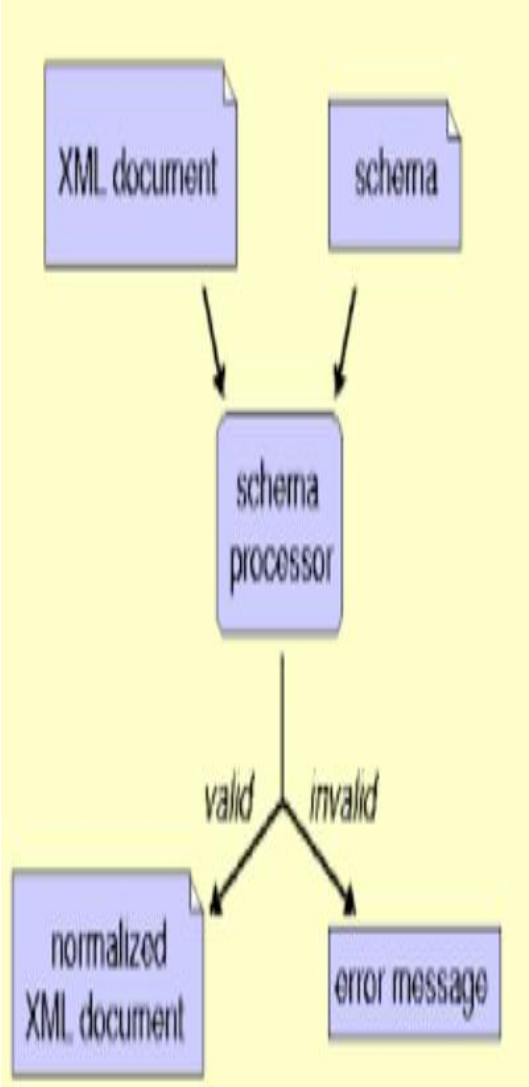
- Other attributes a simple element may have:
  - default="*default value*" *if no other value is specified*
  - fixed="*value*" *no other value may be specified*

XML Schema Data types

- There are two types of data types in XML schema.
- simpleType
- complexType
- **simpleType**
- The simpleType allows you to have text-based elements. It contains less attributes, child elements, and cannot be left empty.
- **complexType**
- The complexType allows you to hold multiple attributes and elements. It can contain additional sub elements and can be left empty.

XML - Processors

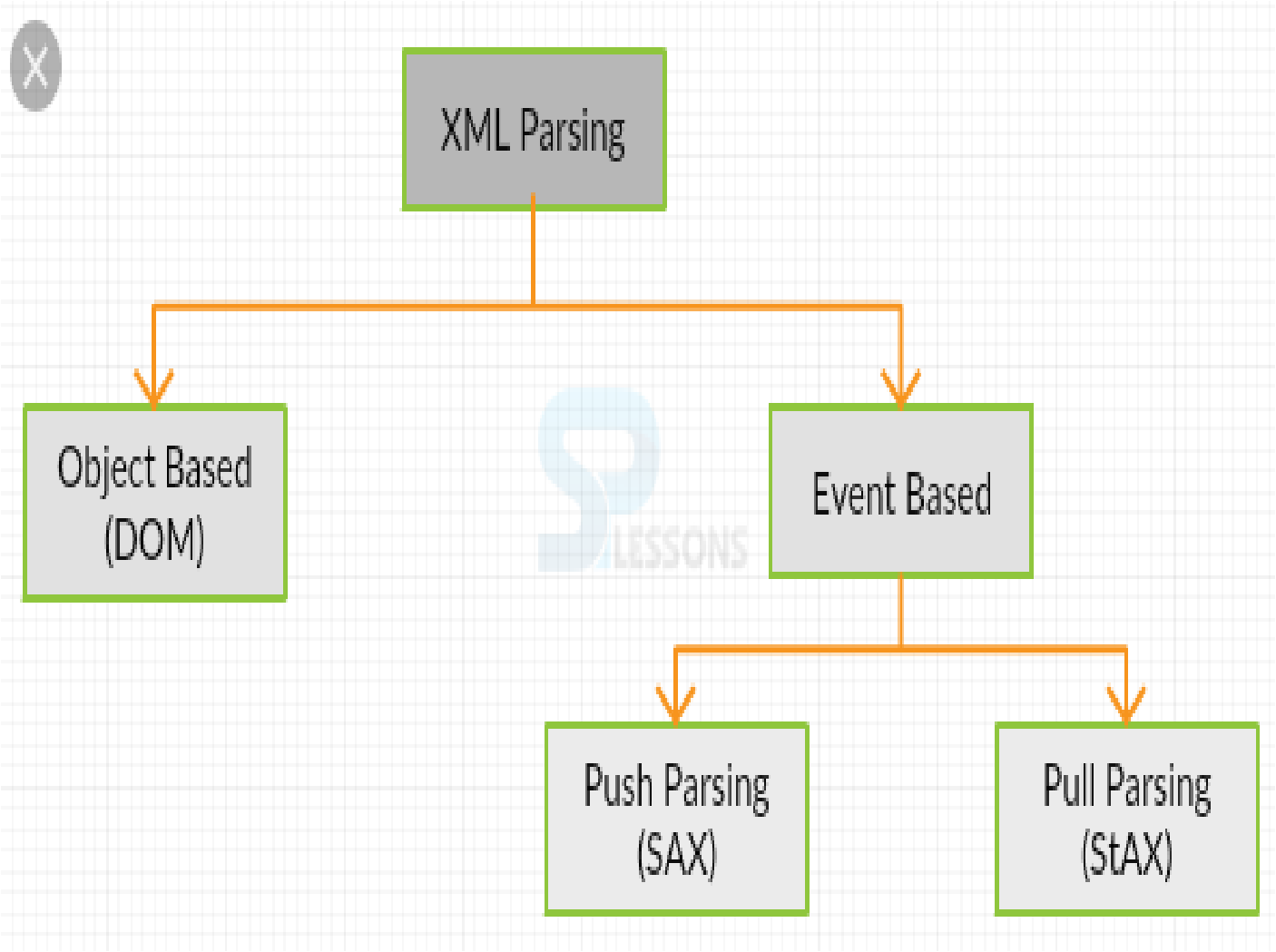
- When a software program reads an XML document and takes actions accordingly, this is called *processing* the XML. Any program that can read and process XML documents is known as an *XML processor*. An XML processor reads the XML file and turns it into in-memory structures that the rest of the program can access.
- The most fundamental XML processor reads an XML document and converts it into an internal representation for other programs or subroutines to use. This is called a *parser*, and it is an important component of every XML processing program.



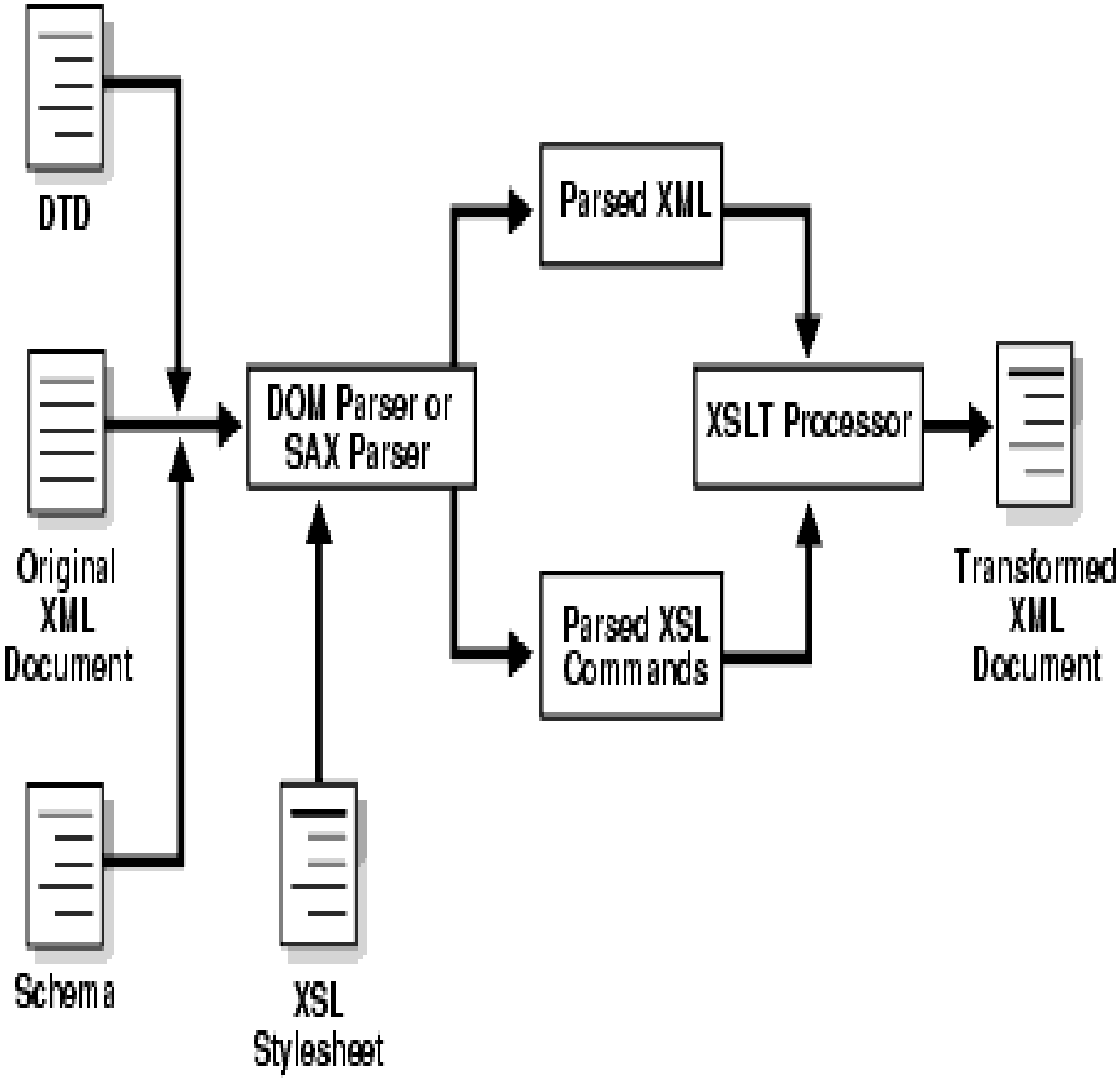
XML Processor : Parsers

- There are two basic types of XML parsers:
- Tree (DOM)-based parser:
  - Whole document is analyzed to create a DOM tree
  - Advantages: Multiple & Random access to elements, easier to validate the structure of XML
- Event-based parser (SAX):
  - XML document is interpreted as a series of events
  - When a specific event occurs, a function is called to handle it (we will see it later in PHP)
  - Advantages: less memory usage and no wait to complete faster

Types of XML Processor : Parsers



Working of a Parser

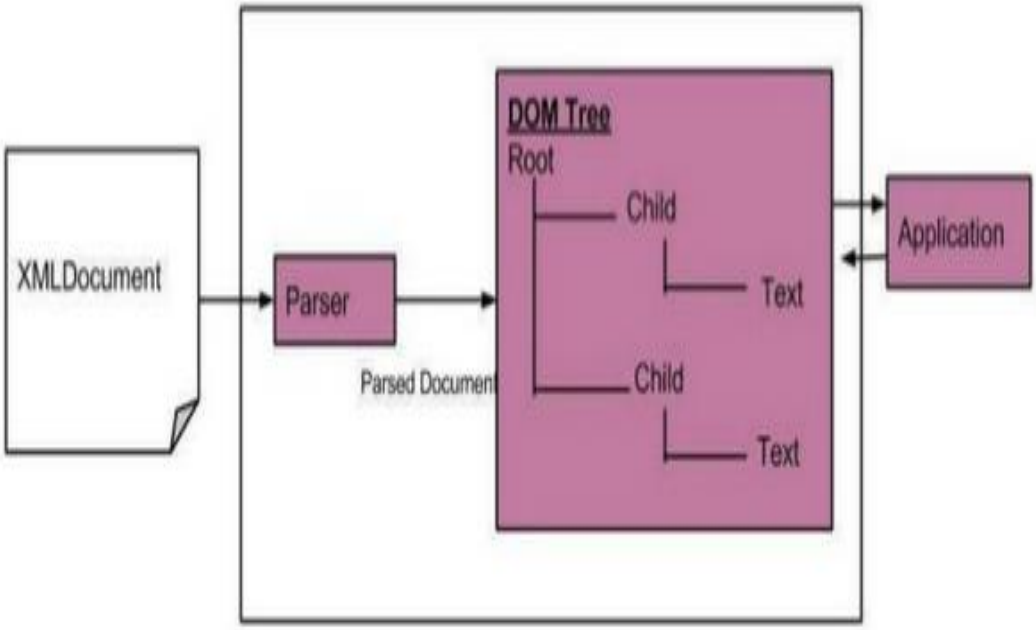


DOM

- The Document Object Model (DOM) is a W3C standard. It defines a standard for accessing documents like HTML and XML.
- Definition of DOM as put by the W3C is:
- The Document Object Model (DOM) is an application programming interface (API) for HTML and XML documents. It defines the logical structure of documents and the way a document is accessed and manipulated. DOM defines the objects and properties and methods (interface) to access all XML elements. The DOM is separated into 3 different parts / levels:
- **Core DOM** - standard model for any structured document
- **XML DOM** - standard model for XML documents
- **HTML DOM** - standard model for HTML documents

XML DOM

- XML DOM is a standard object model for XML.
- XML documents have a hierarchy of informational units called *nodes*; DOM is a standard programming interface of describing those nodes and the relationships between them.
- As XML DOM also provides an API that allows a developer to add, edit, move or remove nodes at any point on the tree in order to create an application.
- Below is the diagram for the DOM structure which depicts that parser evaluates an XML document as a DOM structure by traversing through each nodes.



Dom Document

- A DOM document is a collection of *nodes* or pieces of information, organized in a hierarchy.
- Some types of *nodes* may have *child* nodes of various types and others are leaf nodes that cannot have anything below them in the document structure.
- Below is a list of the node types, and which node types they may have as children:
- ***Document*** -- Element (maximum of one), ProcessingInstruction, Comment, DocumentType (maximum of one)
- ***DocumentFragment*** -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- ***EntityReference*** -- Element, ProcessingInstruction, Comment, Text, CDATASection, EntityReference
- ***Element*** -- Element, Text, Comment, ProcessingInstruction, CDATASection, EntityReference

ADVANTAGES and DISADVANTAGES

- XML DOM is language and platform independent.
- XML DOM is **travesible** - Information in XML DOM is organized in a hierarchy which allows developer to navigate around the hierarchy looking for specific information.
- XML DOM is **modifiable** - It is dynamic in nature providing developer a scope to add, edit, move or remove nodes at any point on the tree.
- It consumes more memory (if the XML structure is large) as program written once remains in memory all the time until and unless removed explicitly.
- Due to the larger usage of memory its operational speed, compared to SAX is slower.