eXtensible Markup Language

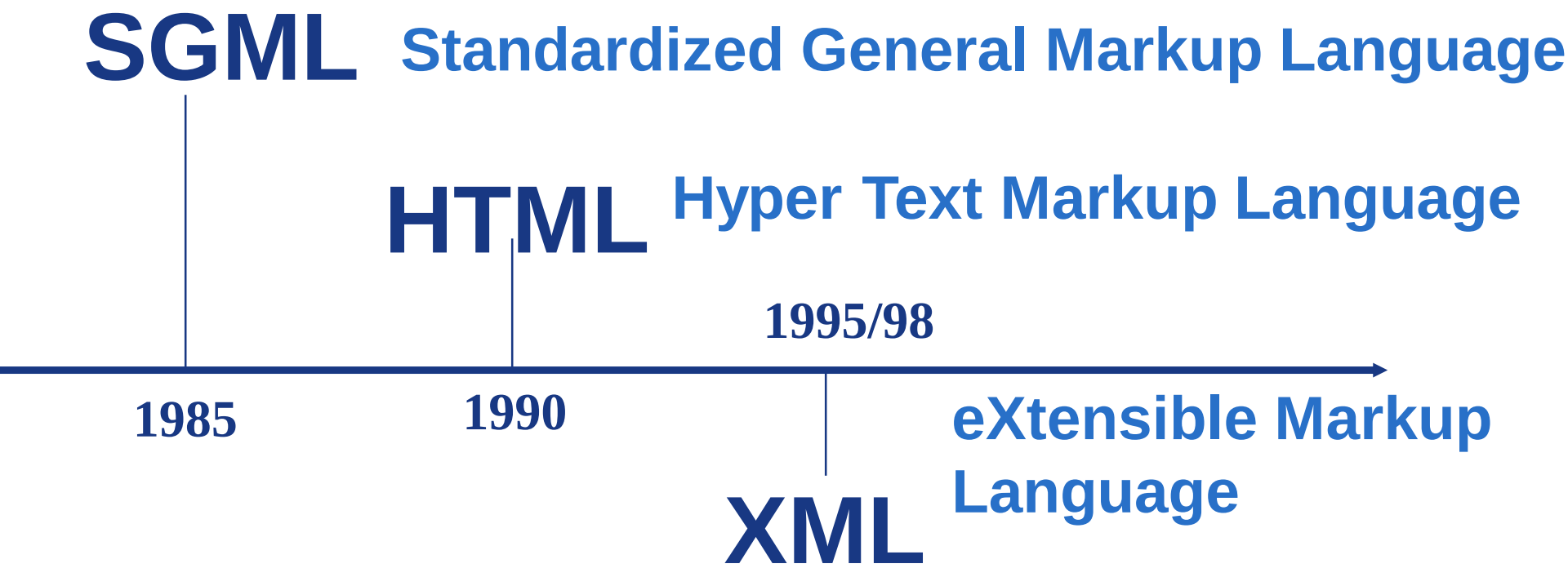
Presented by

Sagar
Java Consultant

eXtensible Markup Language

Markup : A markup language is a computer language that uses tags to define elements within a document.

It is human-readable, meaning markup files contain standard words, rather than typical programming syntax.



Why XML?

- Electronic Data Interchange has been a predominate mode of exchanging data among businesses.
- Electronic data interchange is critical in today's networked world and needs to be standardized
    - Examples:
      Banking: funds transfer
      Education: e-learning contents
- Each application has its own set of standards for representing infromation

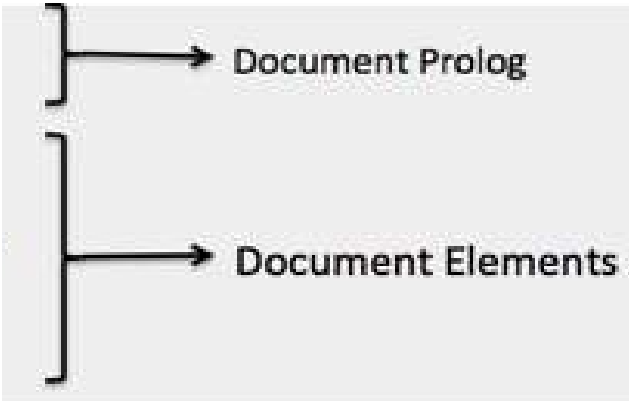
XML – Example – Address data

```
<?xml version="1.0"/>
<address>
    <name>Alice Lee</name>
    <email>alee@aol.com</email>
    <phone>212-346-1234</phone>
    <birthday>1985-03-22</birthday>
</address>
```

XML Example - 2

- Example :

```
<?xml version = "1.0"?>
<contact-info>
   <name>Vikas</name>
   <company>TechMentors</company>
   <phone>9999888877</phone>
</contact-info>
```



- **Document Prolog** comes at the top of the document, before the root element.
- **Document Elements -** Document Elements are the building blocks of XML. These divide the document into a hierarchy of sections, each serving a specific purpose

XML with attributes

```
<books>
    <book isbn="123">
        <title> Second Chance </title>
        <author> Matthew Dunn </author>
    </book>
</books>
```

XML Basic Rules

- XML is case sensitive
- All start tags must have end tags
- Elements must be properly nested
- XML declaration is the first statement
- Every document must contain a root element
- Attribute values must have quotation marks
- Certain characters are reserved for parsing

Documement Type Definition - DTD

- A Document Type Definition (DTD) allows the developer to create a set of rules to specify legal content and place restrictions on an XML file

- If the XML document does not follow the rules contained within the DTD, a parser generates an error

- An XML document that conforms to the rules within a DTD is said to be valid

Why use DTD?

- A single DTD ensures a common format for each XML document that references it
- An application can use a standard DTD to verify that data that it receives from the outside world is valid

DTD and XML Association

address.dtd

```
<!DOCTYPE address.dtd
[
<!ELEMENT address (person,street,city, zip)>
                        <!ELEMENT person  (#PCDATA)>
                        <!ELEMENT street  (#PCDATA)>
                        <!ELEMENT city    (#PCDATA)>
                        <!ELEMENT pin     (#PCDATA)>

]>
```

address.xml

```
                                <?xml version="1.0" encoding="UTF-8"?>
                                <!DOCTYPE address SYSTEM “address.dtd">
                                <address>
                                        <person>Ram</person>
                                        <street>Road No. 1</street>
                                        <city> Pune </city>
                                        <pin>411033</pin>
                                </address>
```

XmlSchemaDefinition

- An XML Schema describes the structure of an XML document.
- XML Schema supports data types
- Default and fixed values for elements and attributes can be set
- **Why to use XSD?**

  It is easier to validate the correctness of data
  It is easier to define data facets (restrictions on data)
  It is easier to define data patterns (data formats)
  It is easier to convert data between different data types

XSD Example

students.xsd

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

<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
   <xs:element name = 'class'>
      <xs:complexType>
         <xs:sequence>
             <xs:element name = 'student' type = 'StudentType' minOccurs = '0'
                maxOccurs = 'unbounded' />
         </xs:sequence>
      </xs:complexType>
   </xs:element>

   <xs:complexType name = "StudentType">
      <xs:sequence>
         <xs:element name = "firstname" type = "xs:string"/>
         <xs:element name = "lastname" type = "xs:string"/>
         <xs:element name = "nickname" type = "xs:string"/>
         <xs:element name = "marks" type = "xs:positiveInteger"/>
      </xs:sequence>
      <xs:attribute name = 'rollno' type = 'xs:positiveInteger'/>
   </xs:complexType>
</xs:schema>
```

XML Using XSD

students.xml

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

<class>
   <student rollno = "393">
      <firstname>Dinkar</firstname>
      <lastname>Kad</lastname>
      <nickname>Dinkar</nickname>
      <marks>85</marks>
   </student>

   <student rollno = "493">
      <firstname>Vaneet</firstname>
      <lastname>Gupta</lastname>
      <nickname>Vinni</nickname>
      <marks>95</marks>
   </student>

   <student rollno = "593">
      <firstname>Jasvir</firstname>
      <lastname>Singh</lastname>
      <nickname>Jazz</nickname>
      <marks>90</marks>
   </student>
</class>
```