XSD
(XML Schema)

XSD

- **Purpose is** to define the legal building blocks of an XML document – like DTD
- It defines the document structure with a list of legal elements

- It defines:
  - elements, attributes,
  - value types, ranges
  - element order
  - …

Example - DTD

```
<?xml version="1.0"?>
    <!DOCTYPE note [
        <!ELEMENT note (to, from, heading, body)>
        <!ELEMENT to (#PCDATA)>
        <!ELEMENT from (#PCDATA)>
        <!ELEMENT heading (#PCDATA)>
        <!ELEMENT body (#PCDATA)>
]>
<note>
    <to> Tove </to>
    <from> Jani </from>
    <heading> Reminder </heading>
    <body> Don't forget me this weekend </body>
</note>
```

Example - XSD

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://www.w3schools.com"
    xmlns="http://www.w3schools.com">
<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>
```

Why XSD instead of DTD?

- XML Schemas:
  - are extensible to future additions
  - are richer and more useful than DTDs
  - are written in XML
  - support data types
  - support namespaces

W3C recommendation

- was originally proposed by Microsoft, but…
- became an official W3C recommendation in May 2001

- BUT… it is used very little until now!!!

Validating

- Browsers can only test if XML document is well formed
- Browsers can’t check XML document validity
  - without external help (plug-in, tool)

Notation

XSD file (XML schema)

XML file

Reference to

Reference to XSD

```
<?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">
…
```

Reference to DTD

```
<?xml version="1.0"?>
<!DOCTYPE note SYSTEM "note.dtd">
…
```

How to – XSD file

```
<?xml version="1.0"?>
    <xs:schema>
    ...
    </xs: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>
    ...
</xs:schema>
```

How to – XML file

```
<?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

- Contain only text
- Pattern

```
<xs:element name="xxx" type="yyy"/>
```

- Example

```
<xs:element name=“firstName" type=“xs:string"/>
```

```
<firstName> Linas </firstName>
```

Simple elements

```
<lastname>Refsnes</lastname>
<age>34</age>
<dateborn>1968-03-27</dateborn>
```

```
<xs:element name="lastname" type="xs:string"/>
<xs:element name="age" type="xs:integer"/>
<xs:element name="dateborn" type="xs:date"/>
```

Complex elements

- empty elements
- elements that contain
  - only other elements
  - only text
  - both other elements and text

Complex elements (2)

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

```
<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>
```

Common XML Schema Data Types

- Most common types:
  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

- There are more ☺

Supports Data Types

- It is easier to :
  - describe permissible document content
  - validate the correctness of data
  - work with data from a database
  - define data facets (restrictions on data)
  - define data patterns (data formats)
  - convert data between different data types

Example – data types

- The date "**03-11-2004**" can be interpreted as:
  - 3rd of November
  - 11th of March

- XML data type “date” requires the format YYYY-MM-DD

```
<date type="date"> 2004-03-11 </date>
```

```
<xs:element name=“start" type="xs:date"/>
```

Data types (groups)

- String
- Date and Time
- Numeric
- Miscellaneous

Examples (date type)

```
<xs:element name="start" type="xs:date"/>
```

```
<start> 2008-10-10Z </start>
<start> 2008-10-10+02:00 </start>
<start> 2002-05-30T09:30:10+06:00 </start>
```

```
<xs:element name="period" type="xs:duration"/>
```

```
<period> P5Y </period>
<period> P5Y2M10DT15H04M20S </period>
```

Default and Fixed Values

- Default value

```
<xs:element name="color" type="xs:string" default="red"/>
```

- Fixed value

```
<xs:element name="color" type="xs:string" fixed="red"/>
```

XSD Attributes

```
<xs:attribute name=“lang" type=“xs:string"/>
```

```
<lastname lang=“LT”>Būtėnas</lastname>
```

- Only complex elements can have attributes !
- Defining attributes:
  - Type xs:string, xs:decimal, xs:integer, …
  - Default <value>
  - Fixed <value>
  - Use <required>

Restrictions / Facets

- Restrictions can be used on:
  - Elements
  - Attributes

```
<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>
```

Example

<age> 32 </age>

<age> -1 </age>

Facets – enumeration

```
<xs:element name="car">
      <xs:simpleType>
       <xs:restriction base="xs:string">
            <xs:enumeration value="Audi"/>
            <xs:enumeration value="Golf"/>
            <xs:enumeration value="BMW"/>
       </xs:restriction>
      </xs:simpleType>
</xs:element>
```

Facets – pattern

```
<xs:element name="initials">
        <xs:simpleType>
         <xs:restriction base="xs:string">

<xs:pattern value="[a-zA-Z][a-zA-Z][a-zA-Z]"/>

         </xs:restriction>
        </xs:simpleType>
</xs:element>
```

Facets – defining your type

```
<xs:element name="car" type="carType"/>


<xs:simpleType name="carType">
        <xs:restriction base="xs:string">
                <xs:enumeration value="Audi"/>
                <xs:enumeration value="Golf"/>
                <xs:enumeration value="BMW"/>
        </xs:restriction>
</xs:simpleType>
```

<car> BMW </car>

<car> TATA</car>

Defining your complex 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>
```

Extending complex types (1)

```
<xs:element name="employee" type="fullpersoninfo"/>

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

… to be continued →

Extending complex types (2)

```
…

<xs:complexType name="fullpersoninfo">
   <xs:complexContent>
      <xs:extension base="personinfo">
         <xs:sequence>
            <xs:element name="address" type="xs:string"/>
            <xs:element name="city" type="xs:string"/>
            <xs:element name="country" type="xs:string"/>
         </xs:sequence>
      </xs:extension>
   </xs:complexContent>
</xs:complexType>
```

Complex type – empty

```
<xs:element name="product">
   <xs:complexType>
      <xs:complexContent>
         <xs:restriction base="xs:integer">
            <xs:attribute name="prodid" type="xs:positiveInteger"/>
         </xs:restriction>
      </xs:complexContent>
   </xs:complexType>
</xs:element>
```

- Both defines the same !

```
<xs:element name="product">
         <xs:complexType>
                  <xs:attribute name="prodid" type="xs:positiveInteger"/>
         </xs:complexType>
</xs:element>
```

Example

<product prodid="20"> </product>

<product prodid="20"/>

<product prodid="20">

Stalas

</product>

Complex type – text only

```
<xs:element name="shoesize">
   <xs:complexType>
      <xs:simpleContent>
         <xs:extension base="xs:integer">
               <xs:attribute name="country" type="xs:string" />
         </xs:extension>
      </xs:simpleContent>
   </xs:complexType>
</xs:element>
```

```
<shoesize country="france">35</shoesize>
```

Complex type – mixed

```
<xs:element name="letter">
   <xs:complexType mixed="true">
      <xs:sequence>
         <xs:element name="name" type="xs:string"/>
         <xs:element name="orderid" type="xs:positiveInteger"/>
         <xs:element name="shipdate" type="xs:date"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

```
<letter>
Dear Mr.<name>John Smith</name>.
Your order <orderid>1032</orderid>
will be shipped on <shipdate>2001-07-13</shipdate>.
</letter>
```

Indicators

- **Order indicators:**
  - All
  - Choice
  - Sequence
- **Occurrence indicators:**
  - maxOccurs
  - minOccurs
- **Group indicators:**
  - Group name
  - attributeGroup name

Element <Any>

```
<xs:element name="letter">
   <xs:complexType>
      <xs:sequence>
         <xs:element name="name" type="xs:string"/>
         <xs:element name="orderid" type="xs:integer"/>
         <xs:any minOccurs="0"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
```

```
<letter>
   <name>John Smith</name>
   <orderid>1032</orderid>
   <naujiMetai>2011-01-01</naujiMetai>
</letter>
```