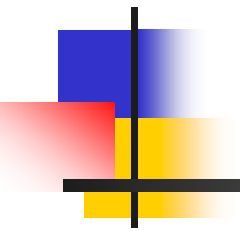
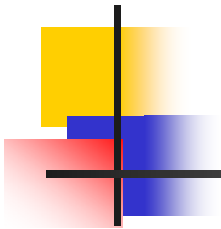
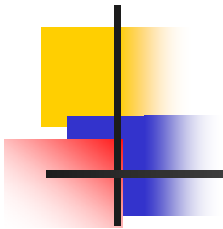


XML Schemas



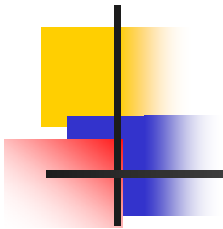
Objectives

- To learn:
  - what an XML Schema is
  - how XML Schema will replace DTD
  - how to use the XML Schema language in your applications



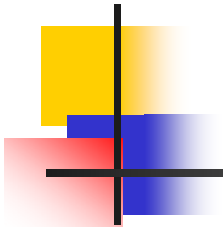
- Lecture outline

- Introduction
- The <schema> element
- A reference to DTD
- A reference to XML schema
- XSD simple elements
- XSD data types
- XSD attributes
- XSD restrictions/facets
- XSD complex elements
- Other XSD components



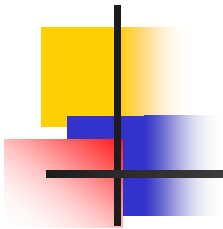
- Introduction

- What is an XML Schema?
- XML Schemas are the Successors of DTDs
- XSD How To
- Simple XML schema



-- What is an XML Schema?

- XML Schema was originally proposed by Microsoft, but became an official W3C recommendation in May 2001.
- The purpose of an XML Schema is to define the legal building blocks of an XML document, just like a DTD.
- An XML Schema:
  - defines elements that can appear in a document
  - defines attributes that can appear in a document
  - defines which elements are child elements
  - defines the order of child elements
  - defines the number of child elements
  - defines whether an element is empty or can include text
  - defines data types for elements and attributes
  - defines default and fixed values for elements and attributes



-- XML Schemas are the Successors of DTDs

- Many think that very soon XML Schemas will be used in most Web applications as a replacement for DTDs. Here are some reasons:
  - XML Schemas are extensible to future additions
  - XML Schemas are richer and more useful than DTDs
  - XML Schemas are written in XML
  - XML Schemas support data types
  - XML Schemas support namespaces

-- XSD (XML Schema Definition) How To

- Look at this simple XML document called "note.xml":
  - <?xml version="1.0"?>
    <note>
    <to>Tove</to>
    <from>Jani</from>
    <heading>Reminder</heading>
    <body>Don't forget me this weekend!</body>
    </note>
- This is a simple DTD file called "note.dtd" that defines the elements of the XML document above ("note.xml"):
  - <!ELEMENT note (to, from, heading, body)>
    <!ELEMENT to (#PCDATA)>
    <!ELEMENT from (#PCDATA)>
    <!ELEMENT heading (#PCDATA)>
    <!ELEMENT body (#PCDATA)>

-- Simple XML 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" 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>
```

- The <schema> element

- The <schema> 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:

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

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

-- XML <schema> element explained

- **xmlns:xs="http://www.w3.org/2001/XMLSchema**
  - indicates that the elements and data types used in the schema (schema, element, complexType, sequence, string, boolean, etc.) come from the "http://www.w3.org/2001/XMLSchema" namespace.
  - It also specifies that the elements and data types that come from the "http://www.w3.org/2001/XMLSchema" namespace should be prefixed with **xs:**
- **targetNamespace="http://www.w3schools.com"**
  - indicates that the elements defined by this schema (note, to, from, heading, body.) come from the "http://www.w3schools.com" namespace.
- **xmlns="http://www.w3schools.com"**
  - indicates that the default namespace is "http://www.w3schools.com".
- **elementFormDefault="qualified"**
  - indicates that any elements used by the XML instance document which were declared in this schema must be namespace qualified.

- A Reference to a DTD

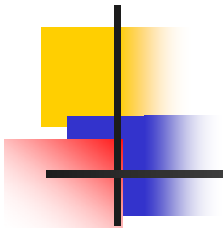
- <?xml version="1.0"?>

```
<!DOCTYPE note SYSTEM "http://www.w3schools.com/dtd/note.dtd">

<note>
      <to>Tove</to>
      <from>Jani</from>
      <heading>Reminder</heading>
      <body>Don't forget me this weekend!</body>
</note>
```

- A Reference to an XML Schema ...

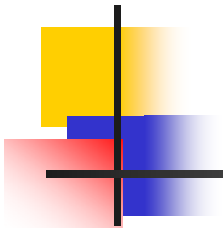
- <?xml version="1.0"?>

```
<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>
```



... -- A Reference to an XML Schema

- xmlns="http://www.w3schools.com"
  - specifies the default namespace declaration. This declaration tells the schema-validator that all the elements used in this XML document are declared in the "http://www.w3schools.com" namespace.
- xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  - XML Schema Instance namespace
- xsi:schemaLocation="http://www.w3schools.com note.xsd"
  - this schemaLocation attribute has two values. The first value is the namespace to use. The second value is the location of the XML schema to use for that namespace



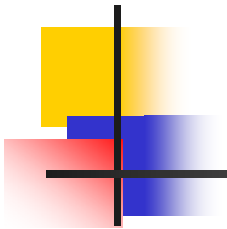
- XSD Simple Elements

- XML Schemas define the elements of your XML files.
- A simple element is an XML element that:
  - can contain only text (boolean, date, string, etc).
  - It cannot contain any other elements or attributes
- The syntax for defining a simple element is:
  - **<xs:element name="xxx" type="yyy"/>**
- Example:

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

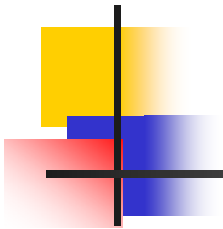
  - Corresponding simple element definition:

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



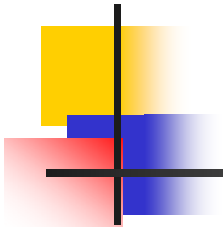
-- Declare Default and Fixed Values for Simple Elements

- Simple elements can have a default value OR a fixed value set.
- 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. You cannot specify another value. In the following example the fixed value is "red":
  - `<xs:element name="color" type="xs:string" fixed="red"/>`



- Common XML Schema Data Types

- XML Schema has a lot of built-in data types. Here is a list of the most common types:
  - xs:string
  - xs:decimal
  - xs:integer
  - xs:boolean
  - xs:date
  - xs:time

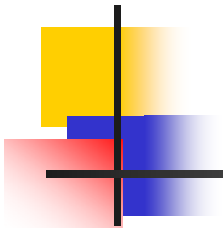


- XSD attributes

- Simple elements cannot have attributes.
- If an element has attributes, it is considered to be of complex type.
- But The attribute itself is always declared as a simple type..
- The syntax for defining an attribute is:
  - `<xs:attribute name="xxx" type="yyy"/>`
- Example:
  - An XML element with an attribute
    - `<lastname lang="EN">Smith</lastname>`
  - A corresponding simple attribute definition:
    - `<xs:attribute name="lang" type="xs:string"/>`

-- Declare Default and Fixed Values for Attributes

- Attributes can have a default value OR a fixed value specified
- A default value is automatically assigned to the attribute when no other value is specified. In the following example the default value is "EN":
  - `<xs:attribute name="lang" type="xs:string" default="EN"/>`
- A fixed value is also automatically assigned to the attribute. You cannot specify another value. In the following example the fixed value is "EN":
  - `<xs:attribute name="lang" type="xs:string" fixed="EN"/>`



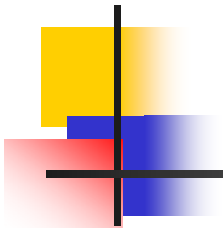
-- Creating Optional and Required Attributes

- All attributes are optional by default.
- To explicitly specify that the attribute is optional, use the "use" attribute
  - <xs:attribute name="lang" type="xs:string" use="optional"/>
- To make an attribute required:
  - <xs:attribute name="lang" type="xs:string" use="required"/>

- XSD restrictions/facets ...

- Restrictions are used to control acceptable values for XML elements or attributes.
- Restrictions are used to control acceptable values for XML elements or attributes.
- This example defines an element called "age" with a restriction. The value of age cannot be lower than 0 or greater than 100:

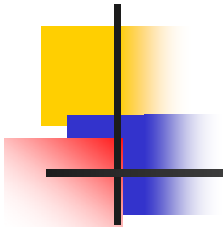
```
<xs:element name="age">
      <xs:simpleType>
            <xs:restriction base="xs:integer">
                  <xs:minInclusive value="0"/>
                  <xs:maxInclusive value="100"/>
            </xs:restriction>
      </xs:simpleType>
</xs:element>
```



... - XSD restrictions/facets

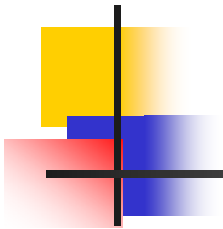
- Restrictions on a Set of Values
  - To limit the content of an XML element to a set of acceptable values, we would use the enumeration constraint
  - This example defines an element called "car":

```
<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>
```



-- Constraint description

- **Enumeration** Defines a list of acceptable values
- **fractionDigits** Specifies the maximum number of decimal places allowed
- **Length** Specifies the exact number of characters or list items allowed.
- **maxExclusive** Specifies the upper bounds for numeric values
- **maxInclusive** Specifies the upper bounds for numeric values
- **maxLength** Specifies the maximum number of characters or list items allowed.
- **minExclusive** Specifies the lower bounds for numeric values
- **minInclusive** Specifies the lower bounds for numeric values
- **minLength** Specifies the minimum number of characters or list items allowed.
- etc



- XSD complex elements

- 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
- Note: Each of these elements may contain attributes as well!

-- Example of complex elements

- A complex XML element, "product", which is empty:

```
<product pid="1345"/>
```

- A complex XML element, "employee", which contains only other elements:

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

- A complex XML element, "food", which contains only text:

```
<food type="dessert">Ice cream</food>
```

- A complex XML element, "description", which contains both elements and text:

```
<description>
   It happened on <date lang="norwegian">03.03.99</date> ....
</description
```

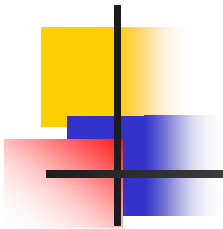
-- How to Define a Complex Element

- Complex XML element:

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

- XSD:

```
<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>
```



-- XSD Complex Types Indicators

- We have seven types of indicators:
  - Order indicators:
    - All
    - Choice
    - Sequence
  - **Occurrence indicators:**
    - maxOccurs
    - minOccurs
  - **Group indicators:**
    - Group name
    - attributeGroup name

-- Order indicators

- All Indicator :
  - specifies by default that the child elements can appear in any order and that each child element must occur once and only once.

```
<xs:element name="person">
    <xs:complexType>
        <xs:all>
            <xs:element name="firstname" type="xs:string"/>
            <xs:element name="lastname" type="xs:string"/>
        </xs:all>
    </xs:complexType>
</xs:element>
```

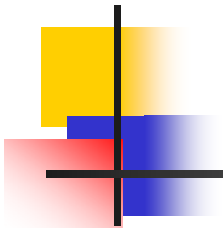
- Choice Indicator:
  - specifies that either one child element or another can occur.
- Sequence Indicator:
  - specifies that the child elements must appear in a specific order.

-- Occurrence Indicators

- maxOccurs indicator:
  - specifies the maximum number of times an element can occur:

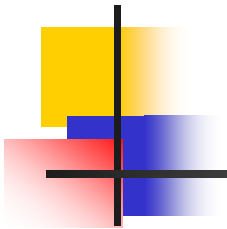
```
<xs:element name="person">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="full_name" type="xs:string"/>
            <xs:element name="child_name" type="xs:string" maxOccurs="10"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

- minOccurs Indicator:
  - specifies the minimum number of times an element can occur



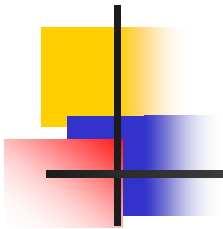
- Other XSD components

- Group indicators:
  - are used to define related sets of elements.
- The <any> element:
  - enables us to extend the XML document with elements not specified by the schema.
- The <anyAttribute> element:
  - enables us to extend the XML document with attributes not specified by the schema.
- Global and local definitions
- XSD Element Substitution
- XSD data types



- References

- W3School DTD Tutorial
  - http://www.w3schools.com/schema/default.asp
- MSXML 4.0 SDK
- http://www.topxml.com
- http://www.xml.org
- http://www.xml.com
- Several online presentations



- Reading list

- W3 Schools DTD Tutorial
  - http://www.w3schools.com/schema/default.asp

END