

# The main problem of DTD' s...

---

**They are not written in XML!**

**Solution:**

**Another XML-based standard: XML Schema**

**For more info see:**

**<http://www.w3.org/XML/Schema>**



# XML Schema (W3C)

---

Thanks to Jussi Pohjolainen  
TAMK University of Applied Sciences

---

w3schools.com

# **XML NAMESPACES**

# XML Namespaces

---

- ❑ The idea behind XML namespaces is to avoid **element name conflicts**.
- ❑ Example of name conflict (w3schools.com)

<table>	<table>
<tr>	<name>African Coffee Table</name>
<td>Apples</td>	<width>80</width>
<td>Bananas</td>	<length>120</length>
</tr>	</table>
</table>	

Same tag-name, different content and meaning!

# Solving Name Conflict

---

```
<h:table>
  <h:tr>
    <h:td>Apples</h:td>
    <h:td>Bananas</h:td>
  </h:tr>
</h:table>
```

Prefix **h** has xhtml-related elements and prefix **f** has furniture-related elements

```
<f:table>
  <f:name>African Coffee Table</f:name>
  <f:width>80</f:width>
  <f:length>120</f:length>
</f:table>
```

# xmlns - attributes

---

- ❑ When using prefixes in XML, a so-called namespace for the prefix must be defined.
- ❑ The namespace is defined by the `xmlns` attribute in the **start tag of an element**.

# xmlns - attribute

---

```
<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="http://www.w3schools.com/furniture">
    <f:name>African Coffee Table</f:name>
    <f:width>80</f:width>
    <f:length>120</f:length>
  </f:table>
</root>
```

# xmlns - attribute

---

```
<root
  xmlns:h="http://www.w3.org/TR/html4/"
  xmlns:f="http://www.w3schools.com/furniture">

  <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>

</root>
```



# Namespace name

---

- ❑ The name of namespace should be unique: `<h:table xmlns:h="http://www.w3.org/TR/html4/">`
- ❑ It is just a string, but it should be declared as URI.
- ❑ Using URI *reduces* the possibility of different namespaces using **duplicate identifiers**.

## Example:

### An XHTML + MathML + SVG Profile

---

- ❑ An XHTML+MathML+SVG profile is a profile that combines XHTML 1.1, MathML 2.0 and SVG 1.1 together.
- ❑ This profile enables mixing XHTML, MathML and SVG in the same document using **XML namespaces mechanism**.

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC
    "-//W3C//DTD XHTML 1.1 plus MathML 2.0 plus SVG 1.1//EN"
    "http://www.w3.org/2002/04/xhtml-math-svg/xhtml-math-svg-flat.dtd">
<html xmlns
    = "http://www.w3.org/1999/xhtml"
    xmlns:svg
    = "http://www.w3.org/2000/svg">
```

```
<head>
  <title>Example of XHTML, SVG and MathML</title>
</head>
<body>
```

```
<h2>MathML</h2>
<p>
  <math xmlns="http://www.w3.org/1998/Math/MathML">
    <mfrac>
      <mi>a</mi>
      <mi>b</mi>
    </mfrac>
  </math>
</p>
```

```
<h2>SVG</h2>
```

```
<p>
  <svg:svg width="50px" height="50px">
    <svg:circle cx="25px" cy="25px" r="20px" fill="green" />
  </svg:svg>
</p>

</body>
</html>
```



MathML

$\frac{a}{b}$

SVG



# W3C SCHEMA



# XML Schema (W3C)

---

- ❑ Language for defining set of rules for XML – documents.
- ❑ W3C Recommendation (2001)
- ❑ More specific than DTD
  - Datatypes!
- ❑ Is XML-language and it uses *xml namespaces*

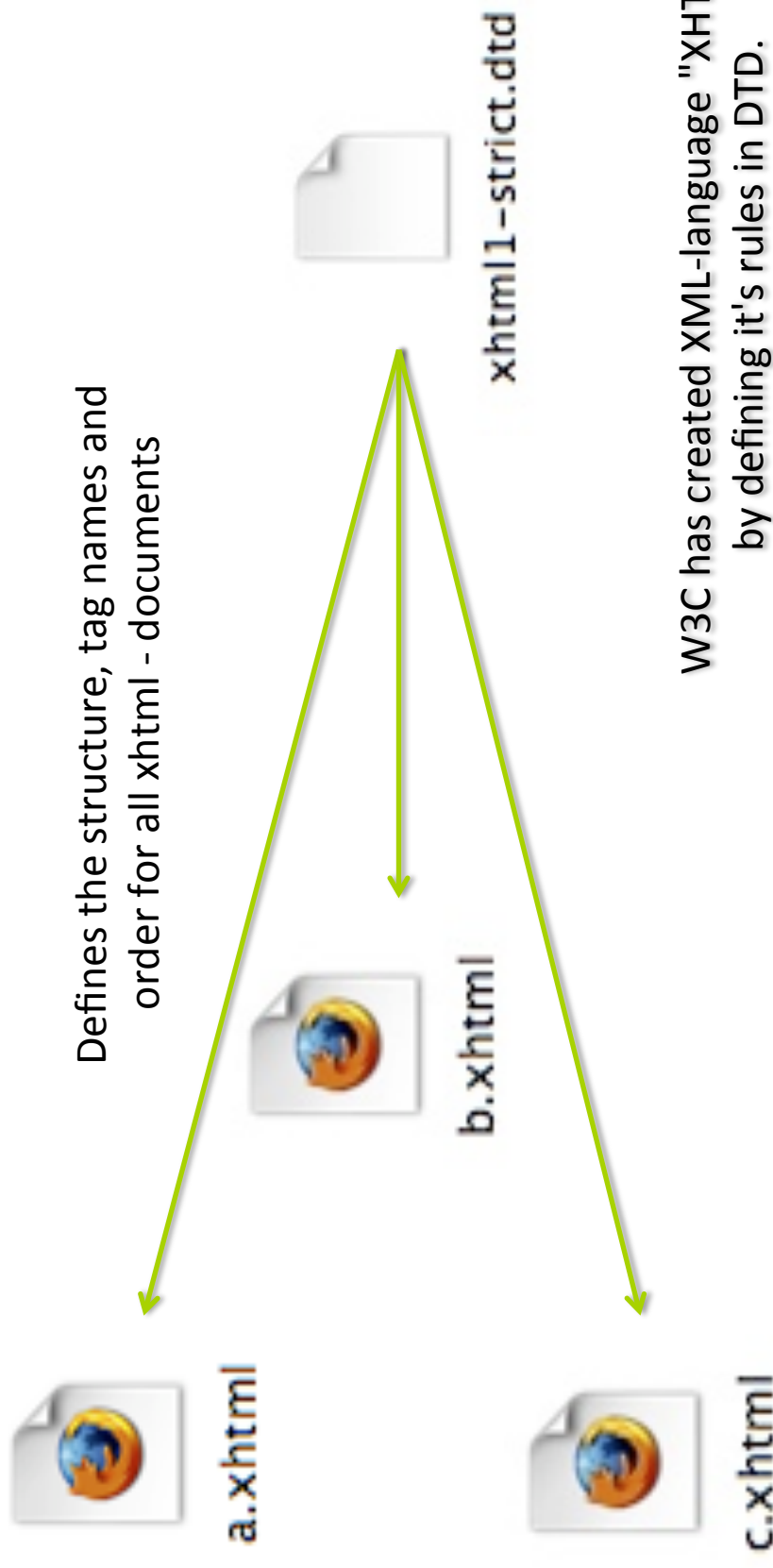
# Schema vs. DTD ([W3Schools.com](http://W3Schools.com))

---

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

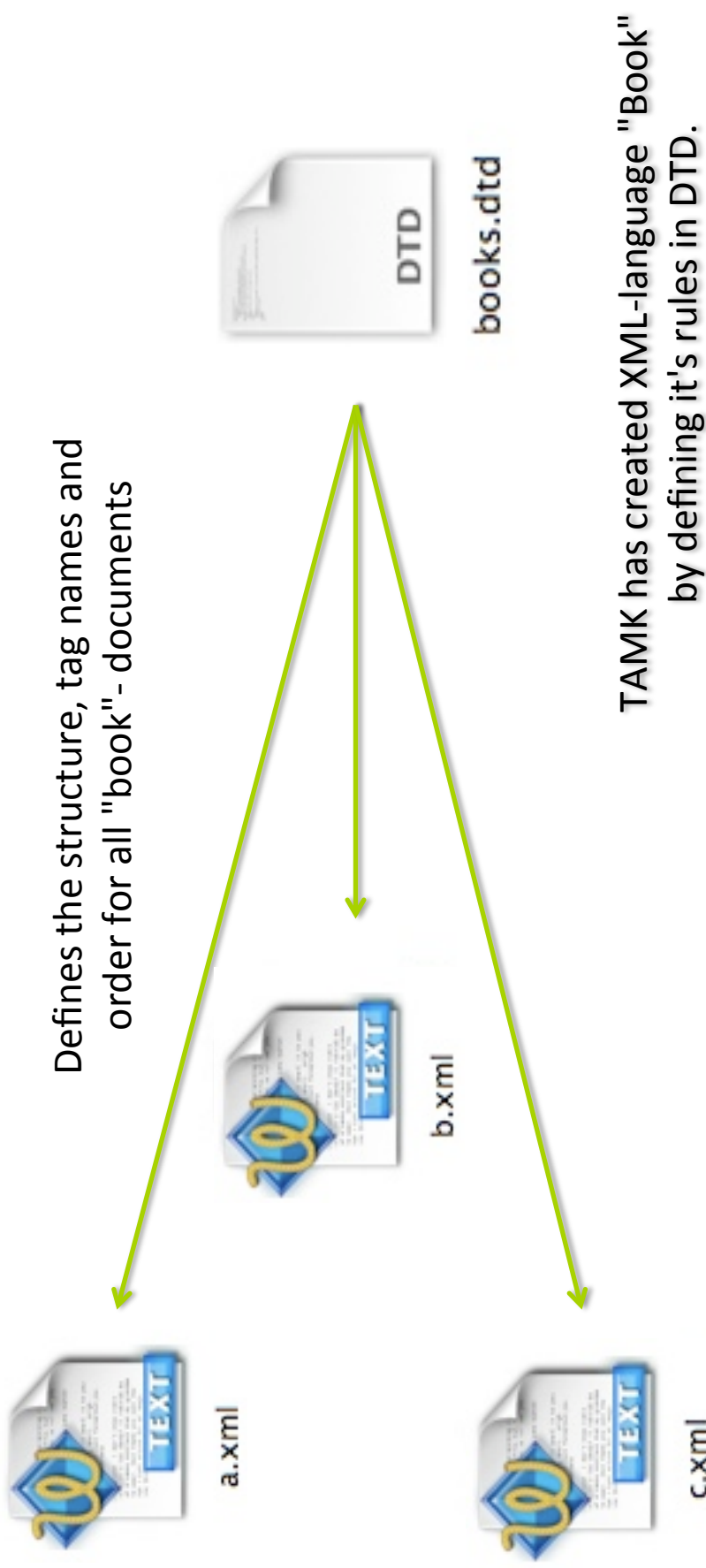
# DTD Linking

---



# DTD Linking

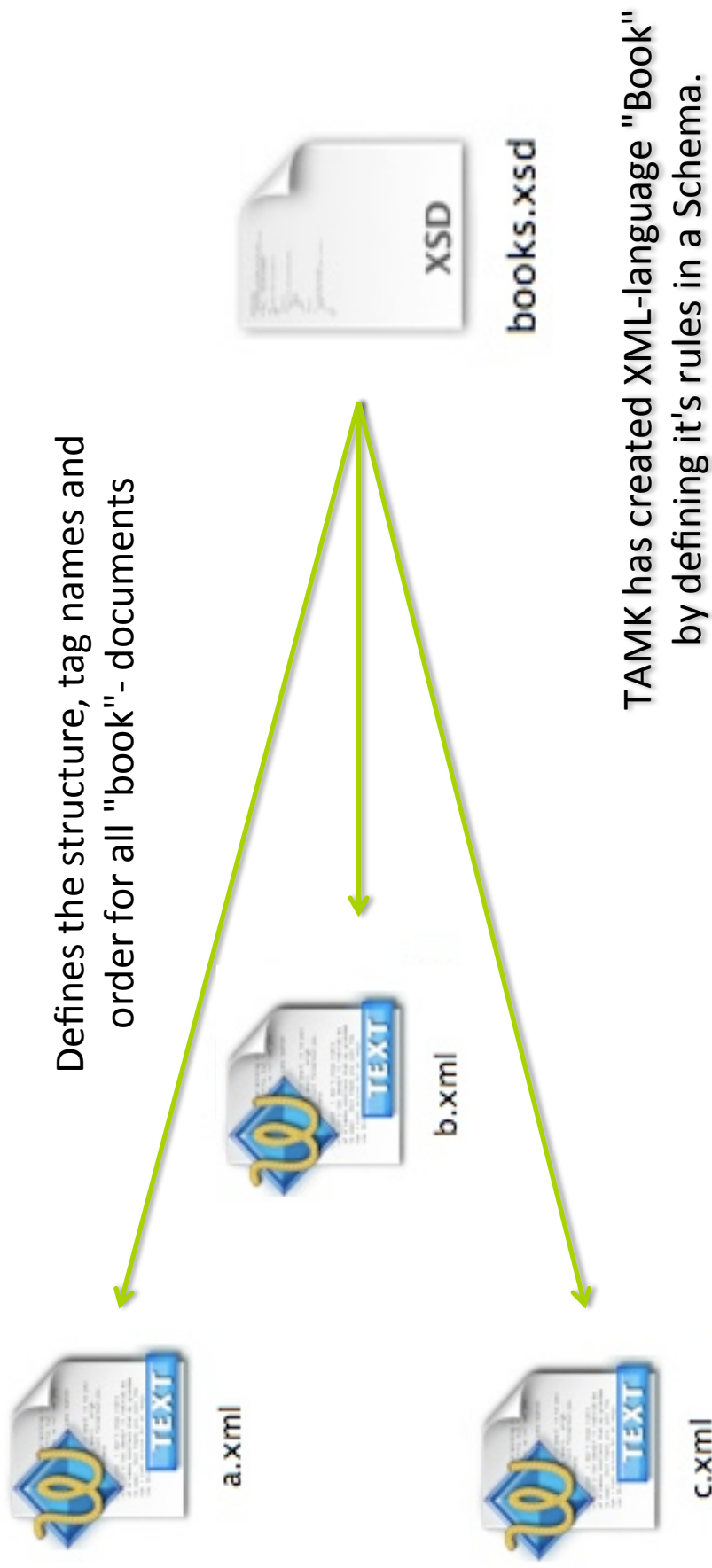
---





# Schema Linking

---



# Linking?

---

- ❑ The basic idea with linking to Schema:

```
<?xml version="1.0"?>  
<root schemaLocation="note.xsd">  
    <foo>...</foo>  
</root>
```

- ❑ The problem with this is that now it is set that attribute "schemaLocation" is part of your XML-language

# Linking and Namespace Usage

---

## □ Linking with namespace

```
<?xml version="1.0"?>
<root
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="note.xsd">
  <foo>...</foo>
</root>
```

- Now the "schemaLocation" – attribute is in it's own namespaces (xsi) and does not belong to the "main" language.

# Simple Schema

---

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="complete_name" type="complete_name_type"/>

  <xsd:complexType name="type="complete_name_type">
    <xsd:sequence>
      <xsd:element name="nome" type="xsd:string"/>
      <xsd:element name="cognome" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

# Let's remove namespaces...

---

```
<?xml version="1.0"?>
<schema>
  <element name="complete_name" type="complete_name_type"/>

  <complexType name="complete_name_type">
    <sequence>
      <element name="nome" type="string"/>
      <element name="cognome" type="string"/>
    </sequence>
  </complexType>
</schema>
```

It doesn't look so confusing  
after all?

# The Basics: Element

---

- ❑ You define the name for the elements by using element-element. 😊
  - `<element name="foo" type="bar" />`
- ❑ Type?
  - 44 Built-in schema datatypes
  - string, double, time, date, etc.
  - [See all the datatypes](#)

# Usage of Datatypes

---

```
<xsd:element name="firstname"  
  type="xsd:string" />
```

```
<xsd:element name="ableToSwim"  
  type="xsd:boolean" />
```

```
<xsd:element name="date"  
  type="xsd:date" />
```

# minOccurs and maxOccurs

---

- The amount of elements
  - In DTD: \*, ?, +
  - In Schema: minOccurs, maxOccurs
    - Example

```
<xsd:element name="date" type="xsd:date" minOccurs="1" maxOccurs="2" />
```
- Default and special values
  - default minOccurs: 1
  - default maxOccurs: same as minOccurs
  - maxOccurs="unbounded" : unlimited



# Defining new Datatypes

---

- ❑ If the built-in datatypes are not enough, you can build your own datatypes.
- ❑ This does not necessarily work:
  - `<xsd:element name="grade" type="xsd:integer" />`
- ❑ There are **two ways** of specifying your own datatype
  - Named Data Type
  - Anonymous Data Type