Hierarchical data formats: XML, XML Schema

Jakub Klímek

XML

Example of a human readable message

```
Dear John Doe,

the balance on your bank account 111333444/1123 as of 3rd of January 2021 is 25000 CZK.



Best regards,

Your bank
1234 5th Avenue
+420123456789
```

Example of a tagged message - 1/3

```
Dear <customer>John Doe</customer>,

the balance on your bank account <accountNumber>111333444/1123</accountNumber> as of
<balanceDate>3rd of January 2021</balanceDate> is <balance>25000 CZK</balance>.


Best regards,

<bankName>Your bank</bankName>
<streetAddress>1234 5th Avenue</streetAddress>
<phone>+420123456789</phone>
```

Example of a tagged message - 2/3

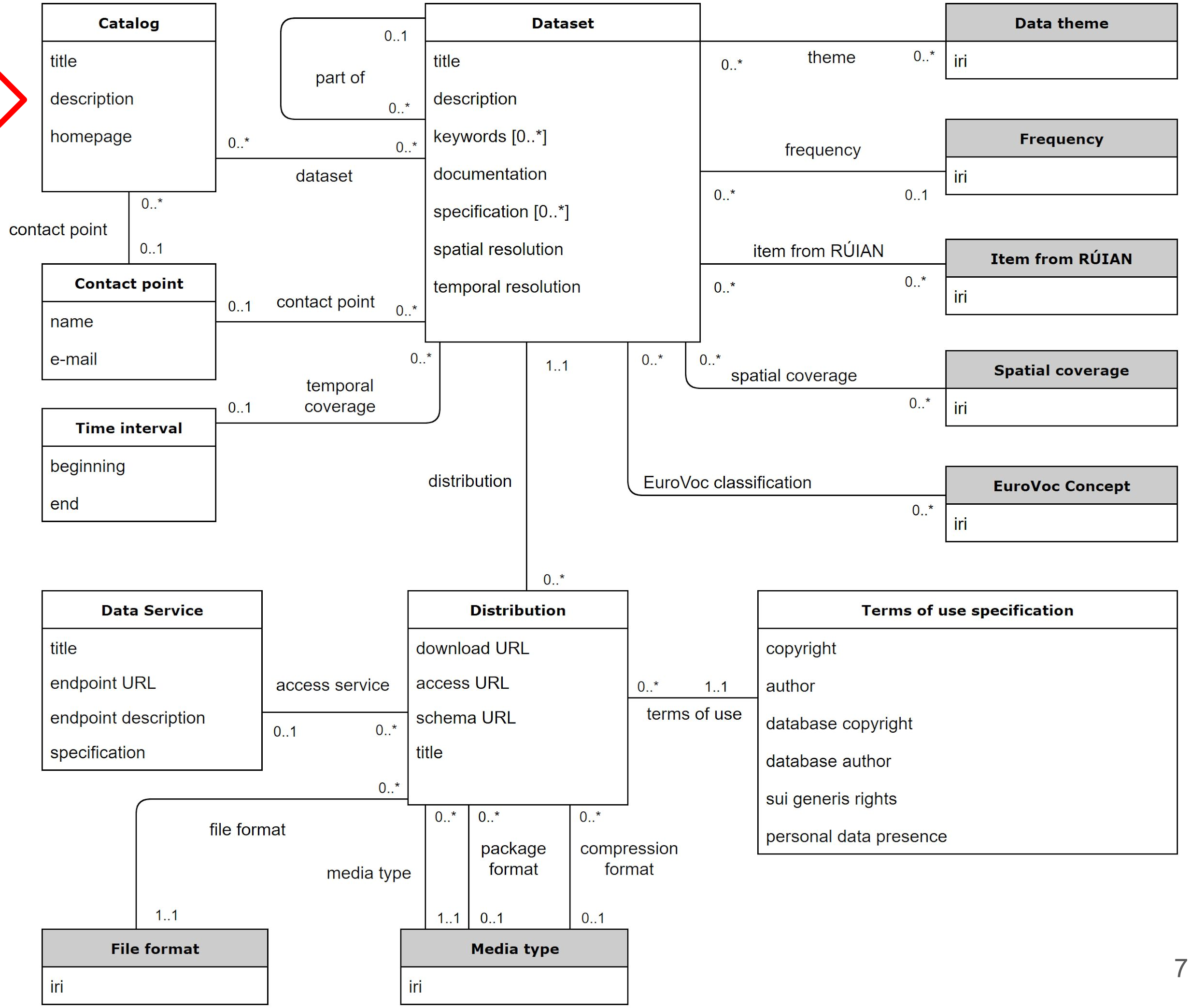
```
Dear <customer><firstName>John</firstName> <lastName>Doe</lastName></customer>,

the balance on your bank account <accountNumber>111333444/1123</accountNumber> as of
<balanceDate>3rd of January 2021</balanceDate> is <balance><value>25000</value>
<currency>CZK</currency></balance>.

Best regards,

<bankName>Your bank</bankName>
<streetAddress>1234 5th Avenue</streetAddress>
<phone>+420123456789</phone>
```

Example of a tagged message - 3/3

```xml
<?xml version="1.0" encoding="UTF-8"?>
<message>
Dear <customer><firstName>John</firstName> <lastName>Doe</lastName></customer>,

the balance on your bank account <accountNumber>111333444/1123</accountNumber> as of
<balanceDate>3rd of January 2021</balanceDate> is <balance><value>25000</value>
<currency>CZK</currency></balance>.

Best regards,

<bankName>Your bank</bankName>
<streetAddress>1234 5th Avenue</streetAddress>
<phone>+420123456789</phone>
</message>
```

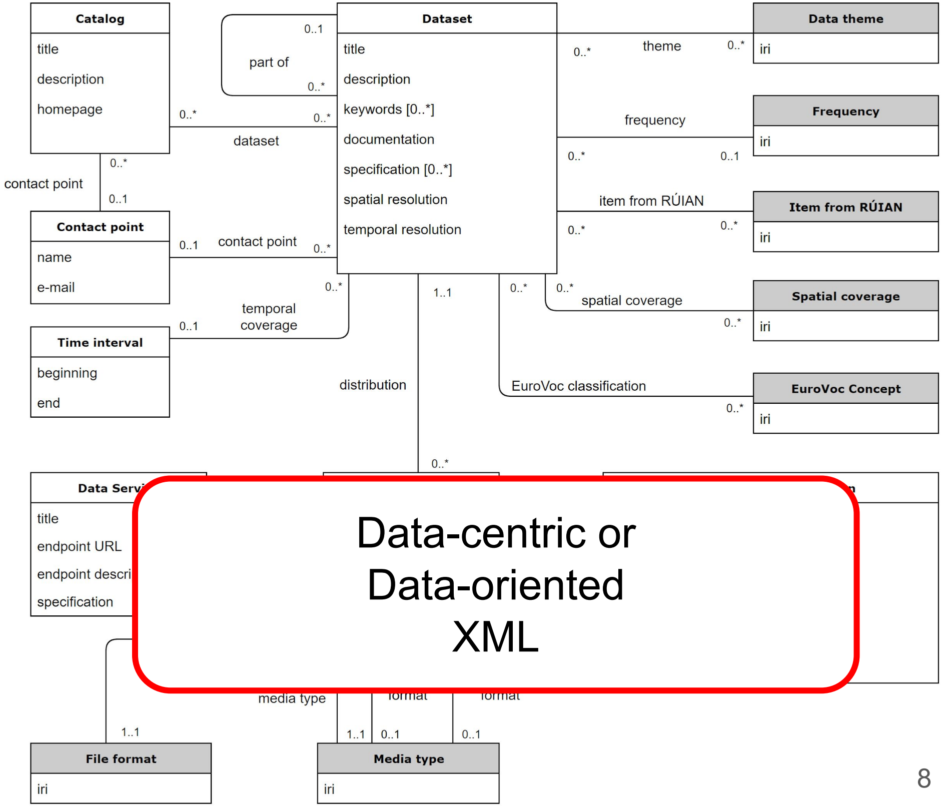
Document-centric or
Document-oriented
XML

We “hang” the data from the Catalog entity



Example of data-centric XML data

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
    <title>My catalog</title>
    <description>This is my dummy catalog</description>
    <contact-point>
        <name>John Doe</name>
        <e-mail>mailto:john@doe.org</e-mail>
    </contact-point>
    <datasets>
        <dataset>
            <title>My first dataset</title>
        </dataset>
        <dataset>
            <title>My second dataset</title>
        </dataset>
    </datasets>
</catalog>
```



Data-centric or Data-oriented XML

XML - eXtensible Markup Language, 1.0 and 1.1

Extensible Markup Language (XML) 1.0

- W3C Recommendation, First edition, **1998**
- widely adopted
- element and attribute names use Unicode 2.0 and list **permitted** characters

Extensible Markup Language (XML) 1.1

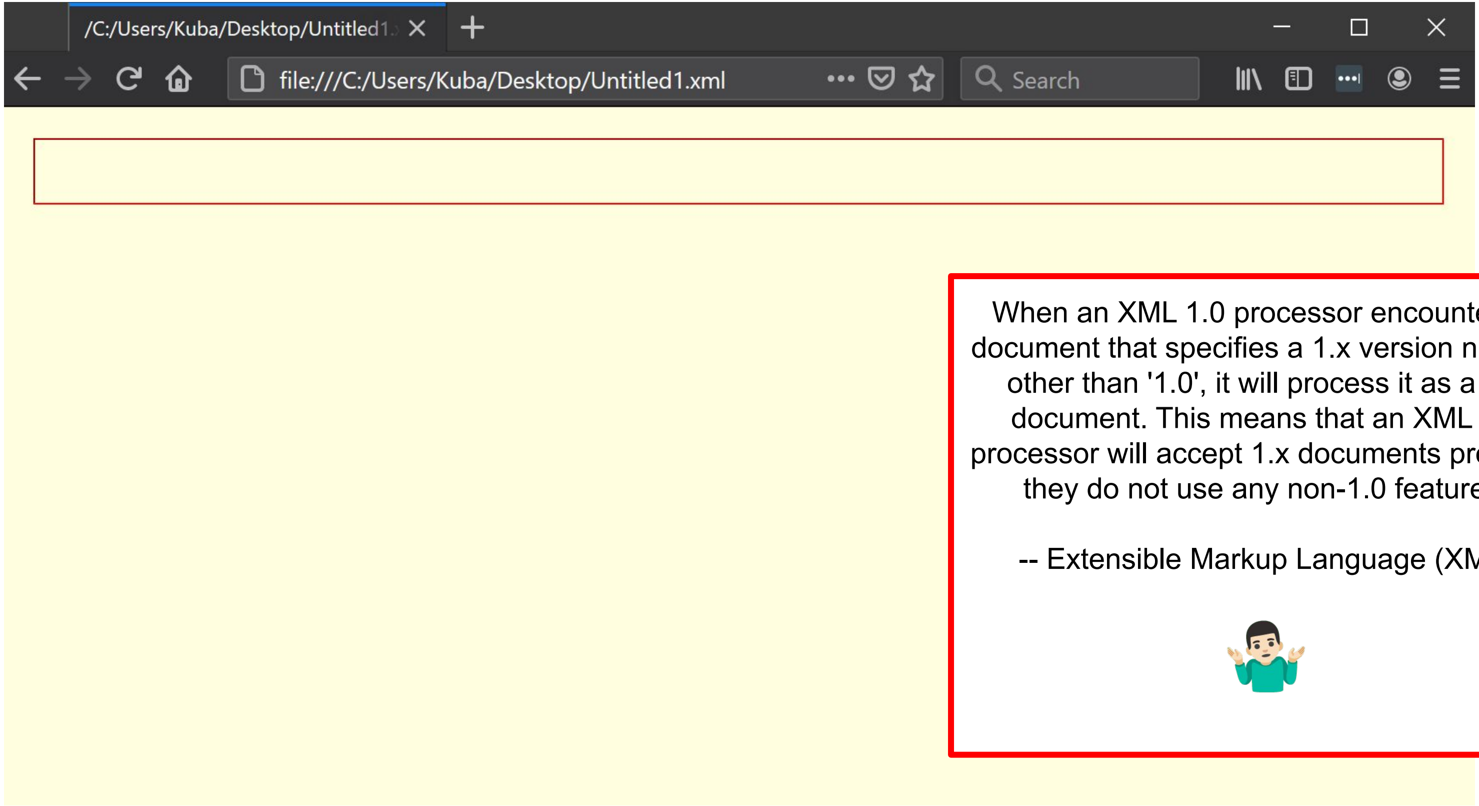
- W3C Recommendation, Second edition, **2006**
- not so widely adopted
- element and attribute names use Unicode and list **forbidden** characters
    - Therefore unicode version independent

Extensible Markup Language (XML) 1.0 (Fifth Edition)

- W3C Recommendation, Fifth edition, **2008**
- relaxes element and naming restrictions (towards XML 1.1)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML declaration -->
<!-- root element -->
<root-element>
   <!-- an empty element -->
   <element/>
   <!-- attributes of an element -->
   <element attribute1="value"
            attribute2="another value">...</element>
   <!-- an element with subelements -->
   <element attribute1="value" attribute2="value">
    <subelement>TEXT CONTENT</subelement>
    <subelement>...</subelement>
   </element>
</root-element>
```

XML 1.1 in Firefox in 2021


/C:/Users/Kuba/Desktop/Untitled1.
file:///C:/Users/Kuba/Desktop/Untitled1.xml
Search

> When an XML 1.0 processor encounters a document that specifies a 1.x version number other than '1.0', it will process it as a 1.0 document. This means that an XML 1.0 processor will accept 1.x documents provided they do not use any non-1.0 features.
>
> -- Extensible Markup Language (XML) 1.1

🤷‍♂️

XML - eXtensible Markup Language

In Czech **legacy** systems also: `iso-8859-2`, `windows-1250`, `utf-16`

Prolog (XML declaration)

- `version` (`1.0` or `1.1`)
- `encoding` (`utf-8`, … - case-insensitive)

start tag
case-sensitive

One root element

Empty element
“/” at the end

Comments
<!-- multiline -->

Attributes
unordered, case-sensitive, unique within tag

Nested elements
subelements, ordered

end tag
“/” at the beginning

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XML declaration -->
<!-- root element -->
<root-element>
   <!-- an empty element -->
   <element/>
   <!-- attributes of an element -->
   <element attribute1="value"
            attribute2="another value">...</element>
   <!-- an element with subelements -->
   <element attribute1="value" attribute2="value">
    <subelement>TEXT CONTENT</subelement>
    <subelement>...</subelement>
   </element>
</root-element>
```

XML - Mixed content

```xml
<?xml version="1.0" encoding="UTF-8"?>
<message>
Dear <customer><firstName>John</firstName> <lastName>Doe</lastName></customer>,

the balance on your bank account <accountNumber>111333444/1123</accountNumber> as of
<balanceDate>3rd of January 2021</balanceDate> is <balance><value>25000</value>
<currency>CZK</currency></balance>.

Best regards,

<bankName>Your bank</bankName>
<streetAddress>1234 5th Avenue</streetAddress>
<phone>+420123456789</phone>
</message>
```

Mixed content

Elements containing both text and nested elements

ordered

Basic syntax errors in XML documents

```
<?xml version="1.0" encoding="UTF-8"?>
<elementA>
   <!-- a bad enclosing symbol -->
   <elementB>...</elementb>
   <!-- an element without an enclosing symbol -->
   <elementD>
   <!-- a bad element nesting -->
   <elementE>
    <elementF>
        …
    </elementE>
   </elementF>
</elementA>
<!-- another root element -->
<elementG />
```

Case mismatch
B vs. b

`elementD` started, but not ended
missing `</elementD>`

`elementE` started first and also ended first
first, the nested `elementF` needs to be closed

There can be only one root

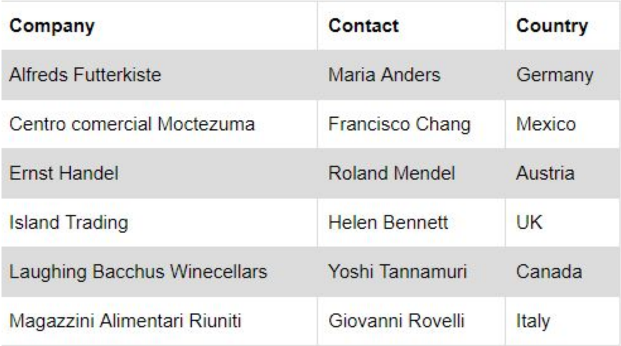
XML document well-formedness

XML document is **well-formed** iff it complies with XML syntax rules.

XML namespaces

https://www.w3schools.com/xml/xml_namespaces.asp

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



Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

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

XML namespaces

https://www.w3schools.com/xml/xml_namespaces.asp

```
<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>
```

XML namespaces, Qualified name (QName)

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

XML namespaces

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

XML namespaces - default namespace

```
<root xmlns="http://www.w3.org/TR/html4/">
 <table>
  <tr>
    <td>Apples</td>
    <td>Bananas</td>
  </tr>
 </table>
</root>
```

XML CDATA sections

```
<?xml version="1.0" encoding="UTF-8"?>
<elementA>
    <![CDATA[<greeting>Hello, world!</greeting>]]>
</elementA>
```

Everything between `<![CDATA[` and `]]>` is treated as string

Natural language specification - `xml:lang` attribute

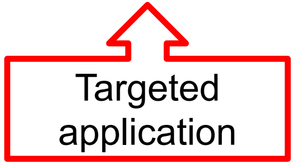
```
<?xml version="1.1" encoding="UTF-8"?>
<document>
    <p xml:lang="en">The quick brown fox jumps over the lazy dog.</p>
    <p xml:lang="en-GB">What colour is it?</p>
    <p xml:lang="en-US">What color is it?</p>
    <sp who="Faust" desc='leise' xml:lang="de">
        <l>Habe nun, ach! Philosophie,</l>
        <l>Juristerei, und Medizin</l>
        <l>und leider auch Theologie</l>
        <l>durchaus studiert mit heißem Bemüh'n.</l>
    </sp>
</document>
```

Language tags specified by IETF BCP 47
consisting of
RFC 4646: Tags for Identifying Languages, and
RFC 4647: Matching of Language Tags

(same as in RDF, HTTP headers, …)

XML processing instruction (PI)

```
<?xml-stylesheet type="text/xsl" href="style.xsl"?>
```



Targeted application

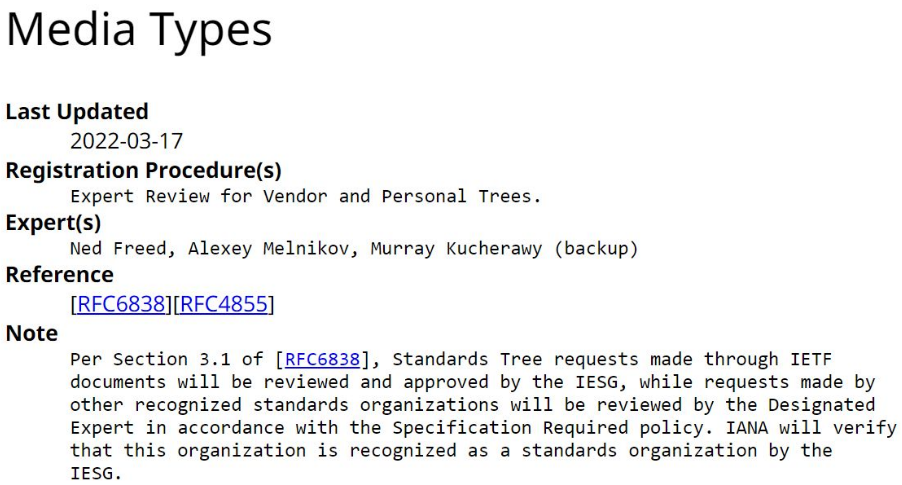
PI not part of XML data

Must be passed to applications by XML parser

Syntax similar to XML declaration in prolog, but **prolog is not a PI**.

XML processing instruction (PI) - Example

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type="text/xsl" href="media-types.xsl"?>
<?xml-model href="media-types.rng"
schematypens="http://relaxng.org/ns/structure/1.0" ?>
<registry xmlns="http://www.iana.org/assignments" id="media-types">
  <title>Media Types</title>
  <category>Multipurpose Internet Mail Extensions (MIME) and Media
Types</category>
  <updated>2022-03-17</updated>
  <registration_rule>Expert Review for Vendor and Personal
Trees.</registration_rule>
  <expert>Ned Freed, Alexey Melnikov, Murray Kucherawy
(backup)</expert>
  <xref type="rfc" data="rfc6838"/>
  <xref type="rfc" data="rfc4855"/>
  <note>Per Section 3.1 of <xref type="rfc" data="rfc6838"/>,
Standards Tree requests made through IETF
documents will be reviewed and approved by the IESG, while requests
made by
other recognized standards organizations will be reviewed by the
Designated
Expert in accordance with the Specification Required policy. IANA
will verify
that this organization is recognized as a standards organization by
the
IESG.</note>
```



Media Types

Last Updated

2022-03-17

Registration Procedure(s)

Expert Review for Vendor and Personal Trees.

Expert(s)

Ned Freed, Alexey Melnikov, Murray Kucherawy (backup)

Reference

[RFC6838][RFC4855]

Note

Per Section 3.1 of [RFC6838], Standards Tree requests made through IETF documents will be reviewed and approved by the IESG, while requests made by other recognized standards organizations will be reviewed by the Designated Expert in accordance with the Specification Required policy. IANA will verify that this organization is recognized as a standards organization by the IESG.

https://www.iana.org/assignments/media-types/media-types.xml

XML entities, entity references, character references

Predefined XML entities

<	&lt;
>	&gt;
&	&amp;
'	&apos;
"	&quot;

There are also other types of entities (omitted in this lecture)

Character references

<	&#60; - decimal
<	&#x3C; - hexadecimal

```
<?xml version="1.0" encoding="UTF-8"?>
<elementA>
    This is content of &lt;elementA&gt; with
    entity references.
</elementA>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<elementB>
    This is content of &#60;elementB&#62; with
    character references.
</elementB>
```

XML constructs not covered here

DTD - Document Type Definition

```
<!DOCTYPE TVSCHEDULE [
<!ELEMENT TVSCHEDULE (CHANNEL+)>
<!ELEMENT CHANNEL (BANNER,DAY+)>
<!ELEMENT BANNER (#PCDATA)>
<!ELEMENT DAY (DATE,(HOLIDAY|PROGRAMSLOT+)+)>
<!ELEMENT HOLIDAY (#PCDATA)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT PROGRAMSLOT (TIME,TITLE,DESCRIPTION?)>
<!ELEMENT TIME (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT DESCRIPTION (#PCDATA)>

<!ATTLIST TVSCHEDULE NAME CDATA #REQUIRED>
<!ATTLIST CHANNEL CHAN CDATA #REQUIRED>
<!ATTLIST PROGRAMSLOT VTR CDATA #IMPLIED>
<!ATTLIST TITLE RATING CDATA #IMPLIED>
<!ATTLIST TITLE LANGUAGE CDATA #IMPLIED>
]>
```

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TVSCHEDULE SYSTEM "tvschedule.dtd">
<TVSCHEDULE>
```

custom XML entities

- part of DTD
- used to reference text or data from a place in XML document

Bad example - XML (XHTML) for tabular data

```
<?xml version="1.0" encoding="UTF-8"?>
<table>
    <tr>
        <th>Name</th>
        <th>Age</th>
        <th>Coffees per day</th>
    </tr>
    <tr>
        <td>John</td>
        <td>20</td>
        <td>2</td>
    </tr>
    <tr>
        <td>Jane</td>
        <td>18</td>
        <td>1</td>
    </tr>
    <tr>
        <td>Steve</td>
        <td>31</td>
        <td>5</td>
    </tr>
</table>
```

Name	Age	Coffees per day
John	20	2
Jane	18	1
Steve	31	5

Obviously, it CAN be done. But is it right?

- XML elements do not reflect the meaning of data
  - they describe the generic structure of a table here
  - good maybe for printing and human reading

Bad example - XML for tabular data

```xml
<?xml version="1.0" encoding="UTF-8"?>
<consumers>



    <consumer>
        <name>John</name>
        <age>20</age>
        <coffees-per-day>2</coffees-per-day>
    </consumer>
    <consumer>
        <name>Jane</name>
        <age>18</age>
        <coffees-per-day>1</coffees-per-day>
    </consumer>
    <consumer>
        <name>Steve</name>
        <age>31</age>
        <coffees-per-day>5</coffees-per-day>
    </consumer>
</consumers>
```

Name	**Age**	**Coffees per day**
John	20	2
Jane	18	1
Steve	31	5

Obviously, it CAN be done. But is it right?

- Better usage of XML elements, but still
    - XML’s hierarchical nature is not used
        - it is not necessary here
    - We unnecessarily force data consumers to use more complex tools => CSV would fit better here

Example - XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<consumers>
    <consumer>
        <name>John</name>
        <drinks>
            <coffee-type when="morning">V60</coffee-type>
            <coffee-type when="after lunch">Batch brew</coffee-type>
            <coffee-type when="afternoon">Flat white</coffee-type>
        </drinks>
        <age>20</age>
        <coffees-per-day>2</coffees-per-day>
    </consumer>
    <consumer>
        <name>Jane</name>
        <drinks>
            <coffee-type when="morning">Aeropress</coffee-type>
            <coffee-type when="after lunch">Cappuccino</coffee-type>
            <coffee-type when="afternoon">Double espresso</coffee-type>
        </drinks>
        <age>18</age>
        <coffees-per-day>1</coffees-per-day>
    </consumer>
</consumers>
```

❌ - violates 1NF and still misses information

Name	Drinks	Age	Coffees per day
John	V60, Batch brew, Flat white	20	2
Jane	Aeropress, Cappuccino, Double espresso	18	1

Name	Drinks	When	Age	Coffees per day
John	V60	morning	20	2
John	Batch brew	after lunch	20	2
John	Flat white	afternoon	20	2

“flattening” - starts being redundant ❌

Example - XML

```xml
<?xml version="1.0" encoding="UTF-8"?>
<consumers>
    <consumer>
        <name>John</name>
        <drinks>
            <coffee-type when="morning">V60</coffee-type>
            <coffee-type when="after lunch">Batch brew</coffee-type>
            <coffee-type when="afternoon">Flat white</coffee-type>
        </drinks>
        <age>20</age>
        <coffees-per-day>2</coffees-per-day>
    </consumer>
    <consumer>
        <name>Jane</name>
        <drinks>
            <coffee-type when="morning">Aeropress</coffee-type>
            <coffee-type when="after lunch">Cappuccino</coffee-type>
            <coffee-type when="afternoon">Double espresso</coffee-type>
        </drinks>
        <age>18</age>
        <coffees-per-day>1</coffees-per-day>
    </consumer>
</consumers>
```

Name	**Age**	**Coffees per day**
John	20	2
Jane	18	1

Name	**Drinks**	**When**
John	V60	morning
John	Batch brew	after lunch
John	Flat white	afternoon

Starts getting complex for sharing ❌

Another bad example from real world

```xml
<Records>
    <Record>
        <header>Sídlo</header>
        <recorded>2014-11-05</recorded>
        <type>
            <code>SIDLO</code>
            <label>sídlo</label>
        </type>
        <address>
            <countryName>Česká republika</countryName>
            <city>Brno</city>
            <partOfCity>Bohunice</partOfCity>
            <street>Neužilova</street>
            <descNumber>201</descNumber>
            <orNumber>35</orNumber>
            <zip>62500</zip>
            <region>Brno-město</region>
        </address>
    </Record>
    <Record>
        <header>Identifikační číslo</header>
        <recorded>2014-11-05</recorded>
        <textValue>3543609</textValue>
        <type>
            <code>ICO</code>
            <label>identifikační číslo</label>
        </type>
    </Record>
</Records>
```

What is this data about?

… generic records in a registry

What should the data be about?

Companies, their IDs, HQ addresses, ...

Another bad example from real world

```xml
<Records>
    <Record>
        <header>Sídlo</header>
        <recorded>2014-11-05</recorded>
        <type>
            <code>SIDLO</code>
            <label>sídlo</label>
        </type>
        <address>
            <countryName>Česká republika</countryName>
            <city>Brno</city>
            <partOfCity>Bohunice</partOfCity>
            <street>Neužilova</street>
            <descNumber>201</descNumber>
            <orNumber>35</orNumber>
            <zip>62500</zip>
            <region>Brno-město</region>
        </address>
    </Record>
    <Record>
        <header>Identifikační číslo</header>
        <recorded>2014-11-05</recorded>
        <textValue>3543609</textValue>
        <type>
            <code>ICO</code>
            <label>identifikační číslo</label>
        </type>
    </Record>
</Records>
```

```xml
<Companies>
    <Company>
        <Identifier recorded="2014-11-05">3543609</Identifier>
        <Address type="HQ" recorded="2014-11-05">
            <countryName>Česká republika</countryName>
            <city>Brno</city>
            <partOfCity>Bohunice</partOfCity>
            <street>Neužilova</street>
            <descNumber>201</descNumber>
            <orNumber>35</orNumber>
            <zip>62500</zip>
            <region>Brno-město</region>
        </Address>
    </Company>
</Companies>
```

What is this data about?

Principal ways of XML processing in apps - DOM

```xml
<?xml version="1.0" encoding="UTF-8"?>
<consumers>
    <consumer>
        <name>John</name>
        <drinks>
            <coffee-type when="morning">V60</coffee-type>
            <coffee-type when="after lunch">Batch brew</coffee-type>
            <coffee-type when="afternoon">Flat white</coffee-type>
        </drinks>
        <age>20</age>
        <coffees-per-day>2</coffees-per-day>
    </consumer>
    <consumer>
        <name>Jane</name>
        <drinks>
            <coffee-type when="morning">Aeropress</coffee-type>
            <coffee-type when="after lunch">Cappuccino</coffee-type>
            <coffee-type when="afternoon">Double espresso</coffee-type>
        </drinks>
        <age>18</age>
        <coffees-per-day>1</coffees-per-day>
    </consumer>
</consumers>
```

How can we process XML data in an application?

1. **Document object model (DOM)**

Loads the entire XML document into memory

- - does not work for streams
- - does not work for large files
- + supports arbitrary querying (a.k.a. random access)

e.g. XPath `/consumers/consumer[1]/name`

Principal ways of XML processing in apps - SAX

```xml
<?xml version="1.0" encoding="UTF-8"?>
<consumers>
    <consumer>
        <name>John</name>
        <drinks>
            <coffee-type when="morning">V60</coffee-type>
            <coffee-type when="after lunch">Batch brew</coffee-type>
            <coffee-type when="afternoon">Flat white</coffee-type>
        </drinks>
        <age>20</age>
        <coffees-per-day>2</coffees-per-day>
    </consumer>
    <consumer>
        <name>Jane</name>
        <drinks>
            <coffee-type when="morning">Aeropress</coffee-type>
            <coffee-type when="after lunch">Cappuccino</coffee-type>
            <coffee-type when="afternoon">Double espresso</coffee-type>
        </drinks>
        <age>18</age>
        <coffees-per-day>1</coffees-per-day>
    </consumer>
</consumers>
```

How can we process XML data in an application?

2. **Simple API for XML (SAX)**

Processes the XML file as a stream of events

+ works for streams
+ works for large files
- does not support effective arbitrary querying (a.k.a. random access)

SAX parser **pushes** events to your code

- until it reads the whole file

Principal ways of XML processing in apps - SAX

```xml
<?xml version="1.0" encoding="UTF-8"?>
<consumers>
    <consumer>
        <name>John</name>
        <drinks>
            <coffee-type when="morning">V60</coffee-type>
            <coffee-type when="after lunch">Batch brew</coffee-type>
            <coffee-type when="afternoon">Flat white</coffee-type>
        </drinks>
        <age>20</age>
        <coffees-per-day>2</coffees-per-day>
    </consumer>
    <consumer>
        <name>Jane</name>
        <drinks>
            <coffee-type when="morning">Aeropress</coffee-type>
            <coffee-type when="after lunch">Cappuccino</coffee-type>
            <coffee-type when="afternoon">Double espresso</coffee-type>
        </drinks>
        <age>18</age>
        <coffees-per-day>1</coffees-per-day>
    </consumer>
</consumers>
```

SAX parser **pushes** events to your code:

1. Element start (name = "consumers")
2. Element start (name = "consumer")
3. Element start (name = "name")
4. Text value (value = "John")
5. Element end (name = "name")
6. Element start (name="drinks")
7. Element start (name="coffee-type")
8. Attribute (name="when" value="morning")
9. Text value (value="V60")
10. Element end (name="coffee-type")
11. ...

Principal ways of XML processing in apps - StAX

```xml
<?xml version="1.0" encoding="UTF-8"?>
<consumers>
    <consumer>
        <name>John</name>
        <drinks>
            <coffee-type when="morning">V60</coffee-type>
            <coffee-type when="after lunch">Batch brew</coffee-type>
            <coffee-type when="afternoon">Flat white</coffee-type>
        </drinks>
        <age>20</age>
        <coffees-per-day>2</coffees-per-day>
    </consumer>
    <consumer>
        <name>Jane</name>
        <drinks>
            <coffee-type when="morning">Aeropress</coffee-type>
            <coffee-type when="after lunch">Cappuccino</coffee-type>
            <coffee-type when="afternoon">Double espresso</coffee-type>
        </drinks>
        <age>18</age>
        <coffees-per-day>1</coffees-per-day>
    </consumer>
</consumers>
```

How can we process XML data in an application?

3. **Streaming API for XML (StAX)**

Processes the XML file as a stream of events

+ works for streams
+ works for large files
- does not support effective arbitrary querying (a.k.a. random access)

Your code **pulls** events from the StAX parser

+ you can stop processing when done

What can be done with XML?

Exploitation

- Definition of specific formats

Parsing

- DOM, SAX, StAX, LINQ

Validation

- XML Schema (XSD)
  - most wide-spread
- RelaxNG (out of scope)
  - XSD alternative
- DTD (out of scope)
  - basically deprecated
- Schematron (out of scope)
  - rule-based validation

Querying

- XPath
- XQuery (out of scope)

Transformation

- XSLT

Persistence

- XML databases (out of scope)
- SQL/XML (out of scope)

Message transfer

- Web services (SOAP, WSDL)

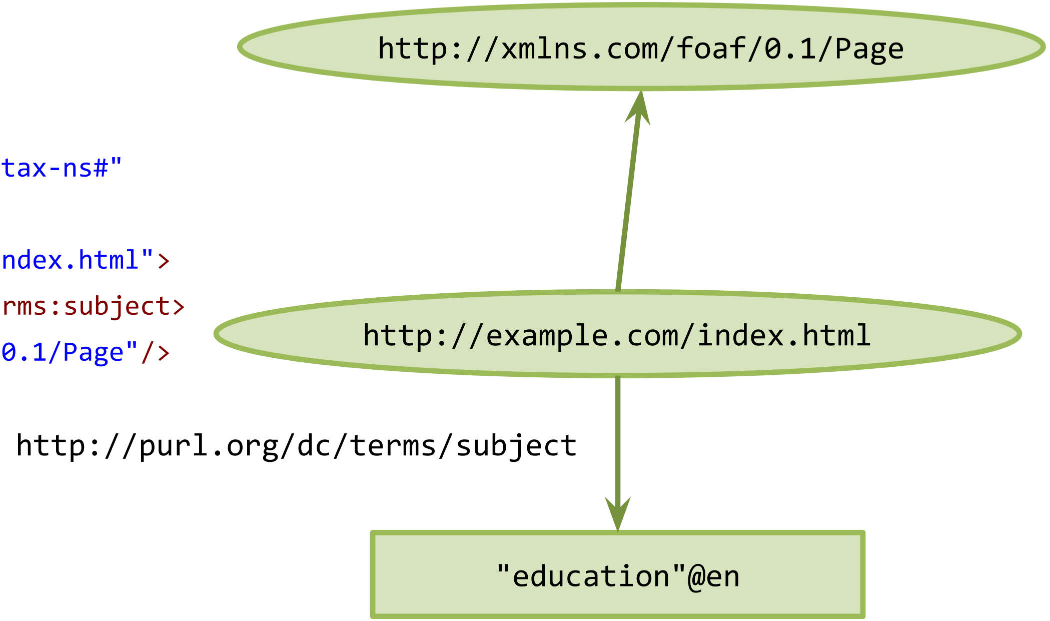
RDF 1.1 XML Syntax (RDF/XML)

The oldest RDF serialization

- RDF 1.1 XML Syntax, W3C Recommendation, **2014**
- the only standard one for RDF 1.0 in **2004**
  - meant to allow XML-based systems transfer RDF data
  - difficult for users to read and process
    - serialization of graph-based RDF model into hierarchical XML document
    - reason for many (unwarranted) antipathies towards RDF

RDF/XML - basic RDF triples

```xml
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:dcterms="http://purl.org/dc/terms/"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
>
  <rdf:Description rdf:about="http://example.com/index.html">
    <dcterms:subject xml:lang="en">Education</dcterms:subject>
    <rdf:type rdf:resource="http://xmlns.com/foaf/0.1/Page"/>
  </rdf:Description>
</rdf:RDF>
```


http://xmlns.com/foaf/0.1/Page
http://example.com/index.html
http://purl.org/dc/terms/subject
"education"@en

RDF/XML - typed literals

Typed literals

```
<rdf:Description rdf:about="SubjectResource">
    <PredicateResource rdf:datatype="LiteralType">
        ObjectLiteral
</PredicateResource>
</rdf:Description>

<!-- Alternatively using XML entities -->

<!DOCTYPE rdf:RDF [<!ENTITY my "TypePrefix">]>
…
<… rdf:datatype="&my;TypeName">…</…>
```

RDF/XML - blank nodes

```
<rdf:RDF …>
    <rdf:Description rdf:about="SubjectResource">
        <PredicateResource rdf:nodeID="BlankNode"/>
    </rdf:Description>
    <rdf:Description rdf:nodeID="BlankNode">
        …
    </rdf:Description>
    …
</rdf:RDF>
```

RDF/XML - XML literals

In RDF graph, this literal has datatype `rdf:XMLLiteral`

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:dc="http://purl.org/dc/elements/1.1/"
            xml:base="http://www.example.com/books">

  <rdf:Description rdf:ID="book12345">
     <dc:title rdf:parseType="Literal">
       <span xml:lang="en">
         The <em>&lt;br /&gt;</em> Element Considered Harmful.
       </span>
     </dc:title>
  </rdf:Description>

</rdf:RDF>
```

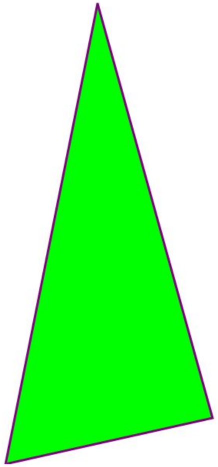
Specific XML formats example - SVG

Scalable Vector Graphics (SVG) 1.1 (Second Edition)
W3C Recommendation **2011**

Scalable Vector Graphics (SVG) 2
W3C Candidate Recommendation **2018**

Explained later in the course

```
<svg height="210" width="500">
  <polygon points="200,10 250,190 160,210"
style="fill:lime;stroke:purple;stroke-width:1" />
</svg>
```

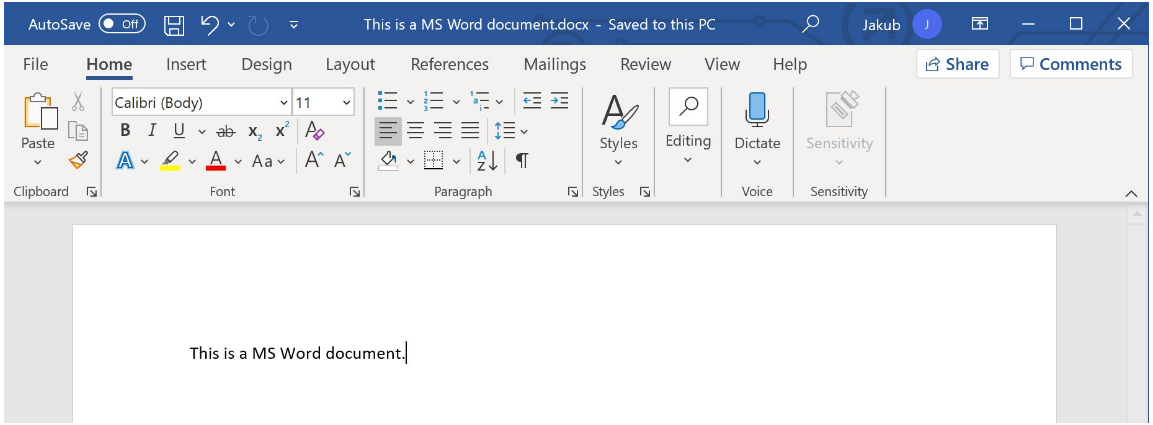


Specific XML formats example - OOXML

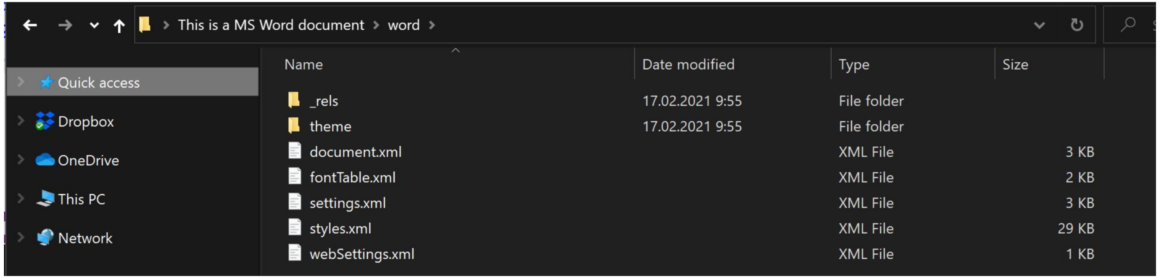
ECMA-376 Office Open XML
ISO/IEC 29500
5th edition, December 2016

.docx, .xlsx, .pptx, …

Zipped XML format



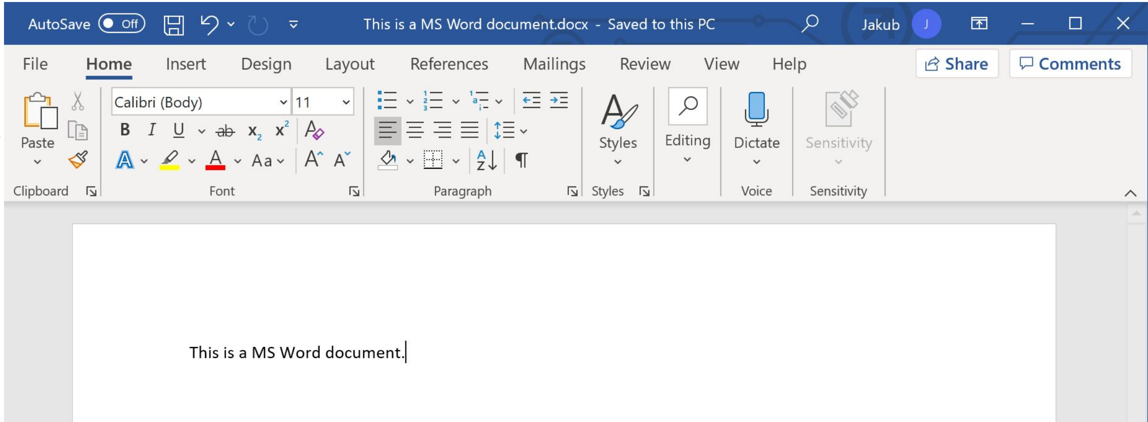
This is a MS Word document.



Name	Date modified	Type	Size
_rels	17.02.2021 9:55	File folder	
theme	17.02.2021 9:55	File folder	
document.xml		XML File	3 KB
fontTable.xml		XML File	2 KB
settings.xml		XML File	3 KB
styles.xml		XML File	29 KB
webSettings.xml		XML File	1 KB

Specific XML formats example - OOXML

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<w:document xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas" xmlns:cx="http://schemas.microsoft.com/office/drawing/2014/chartex" xmlns:cx1="http://schemas.microsoft.com/office/drawing/2015/9/8/chartex"
xmlns:cx2="http://schemas.microsoft.com/office/drawing/2015/10/21/chartex" xmlns:cx3="http://schemas.microsoft.com/office/drawing/2016/5/9/chartex" xmlns:cx4="http://schemas.microsoft.com/office/drawing/2016/5/10/chartex" xmlns:cx5="http://schemas.microsoft.com/office/drawing/2016/5/11/chartex"
xmlns:cx6="http://schemas.microsoft.com/office/drawing/2016/5/12/chartex" xmlns:cx7="http://schemas.microsoft.com/office/drawing/2016/5/13/chartex" xmlns:cx8="http://schemas.microsoft.com/office/drawing/2016/5/14/chartex" xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
xmlns:aink="http://schemas.microsoft.com/office/drawing/2016/ink" xmlns:am3d="http://schemas.microsoft.com/office/drawing/2017/model3d" xmlns:o="urn:schemas-microsoft-com:office:office" xmlns:r="http://schemas.openxmlformats.org/officeDocument/2006/relationships"
xmlns:m="http://schemas.openxmlformats.org/officeDocument/2006/math" xmlns:v="urn:schemas-microsoft-com:vml" xmlns:wp14="http://schemas.microsoft.com/office/word/2010/wordprocessingDrawing" xmlns:wp="http://schemas.openxmlformats.org/drawingml/2006/wordprocessingDrawing"
xmlns:w10="urn:schemas-microsoft-com:office:word" xmlns:w="http://schemas.openxmlformats.org/wordprocessingml/2006/main" xmlns:w14="http://schemas.microsoft.com/office/word/2010/wordml"
xmlns:w15="http://schemas.microsoft.com/office/word/2012/wordml" xmlns:w16cex="http://schemas.microsoft.com/office/word/2018/wordml/cex" xmlns:w16cid="http://schemas.microsoft.com/office/word/2016/wordml/cid" xmlns:w16="http://schemas.microsoft.com/office/word/2018/wordml"
xmlns:w16se="http://schemas.microsoft.com/office/word/2015/wordml/symex" xmlns:wpg="http://schemas.microsoft.com/office/word/2010/wordprocessingGroup" xmlns:wpi="http://schemas.microsoft.com/office/word/2010/wordprocessingInk" xmlns:wne="http://schemas.microsoft.com/office/word/2006/wordml"
xmlns:wps="http://schemas.microsoft.com/office/word/2010/wordprocessingShape" mc:Ignorable="w14 w15 w16se w16cid w16 w16cex wp14">
    <w:body>
        <w:p w14:paraId="357D7129" w14:textId="66D7F77B" w:rsidR="00451A6F" w:rsidRDefault="00D55538">
            <w:r>
                <w:t>This is a MS Word document.</w:t>
            </w:r>
        </w:p>
        <w:sectPr w:rsidR="00451A6F">
            <w:pgSz w:w="11906" w:h="16838"/>
            <w:pgMar w:top="1417" w:right="1417" w:bottom="
            <w:cols w:space="708"/>
            <w:docGrid w:linePitch="360"/>
        </w:sectPr>
    </w:body>
</w:document>
```



This is a MS Word document.

Specific XML formats example - Atom, RSS

RFC 4287 - The Atom Syndication Format

Web feeds

Alternative to:

- RSS 0.9, 1.0, 1.1
  - RDF Site Summary
  - based on early RDF/XML draft
- RSS 2.x
  - Really Simple Syndication
  - pure XML based

```xml
<feed xmlns="http://www.w3.org/2005/Atom">

    <title>Example Feed</title>
    <subtitle>A subtitle.</subtitle>
    <link href="http://example.org/feed/" rel="self" />
    <link href="http://example.org/" />
    <id>urn:uuid:60a76c80-d399-11d9-b91C-0003939e0af6</id>
    <updated>2003-12-13T18:30:02Z</updated>

    <entry>
        <title>Atom-Powered Robots Run Amok</title>
        <link href="http://example.org/2003/12/13/atom03" />
        <link rel="alternate" type="text/html" href="http://example.org/2003/12/13/atom03.html"/>
        <link rel="edit" href="http://example.org/2003/12/13/atom03/edit"/>
        <id>urn:uuid:1225c695-cfb8-4ebb-aaaa-80da344efa6a</id>
        <updated>2003-12-13T18:30:02Z</updated>
        <summary>Some text.</summary>
        <content type="xhtml">
            <div xmlns="http://www.w3.org/1999/xhtml">
                <p>This is the entry content.</p>
            </div>
        </content>
        <author>
            <name>John Doe</name>
            <email>johndoe@example.com</email>
        </author>
    </entry>

</feed>
```

XML Schema

XML Schema - Example

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns:vc="http://www.w3.org/2007/XMLSchema-versioning"
           elementFormDefault="qualified" attributeFormDefault="unqualified"
           vc:minVersion="1.1">
    <xs:complexType name="TypeAddress">
        <!-- specification of content -->
        <xs:sequence>
            <xs:element name="Street" type="xs:string"/>
            <xs:element name="Number" type="xs:integer"/>
            <xs:element name="City" type="xs:string"/>
        </xs:sequence>
        <!-- specification of attributes -->
        <xs:attribute name="Country" type="xs:string" default="CZ"/>
    </xs:complexType>
    <xs:element name="Address" type="TypeAddress"/>
</xs:schema>
```

XSD and XML documents / instances

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  ... <!-- XML schema definition  --> …
</xs:schema>
```

```
<?xml version="1.0" encoding="utf-8"?>
<root_element_of_XML_document
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="schema2.xsd">
  ... <!-- XML document --> …
</root_element_of_XML_document>
```

Link to XML Schema

XML document validity

XML document is **valid** iff it validates against an XML schema.

(for instance XML Schema, but also DTD, Relax NG, Schematron, ...)

XML Schema

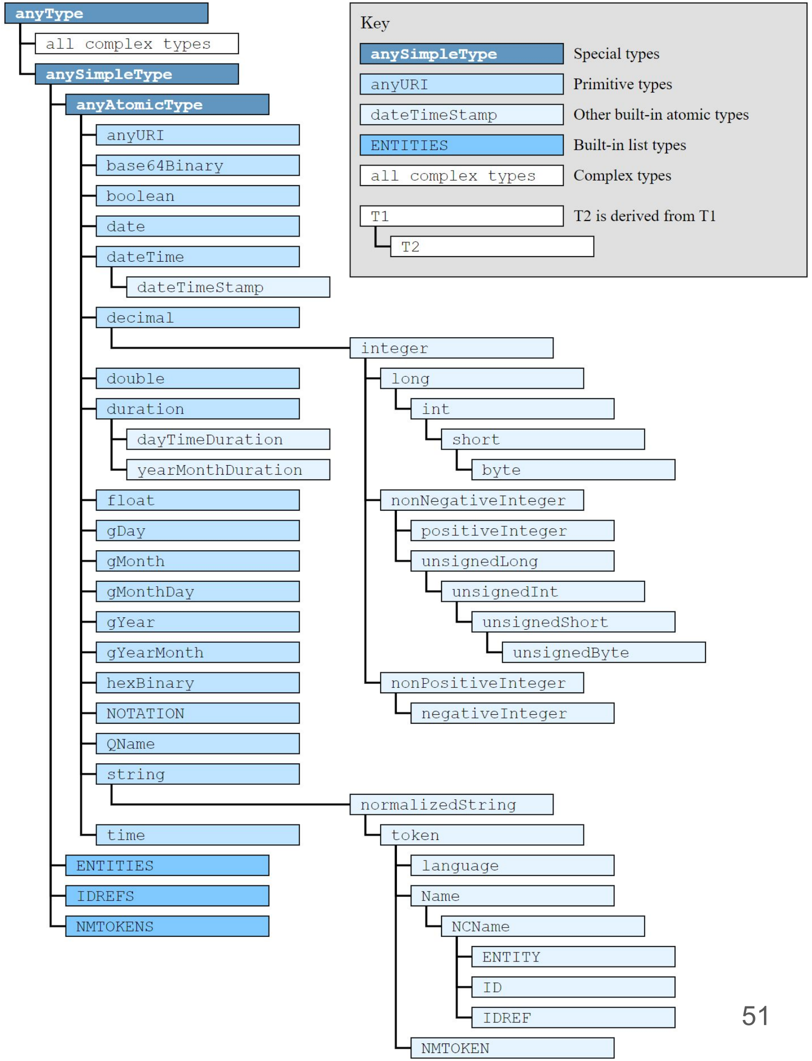
- W3C Recommendations
  - Structures: https://www.w3.org/TR/xmlschema11-1/
  - Data types: https://www.w3.org/TR/xmlschema11-2/
- Currently XML Schema 1.1, 2012
  - as with XML 1.0/1.1, lots of parsers and validators still only support XML Schema 1.0

XML Schema 1.1 Simple types

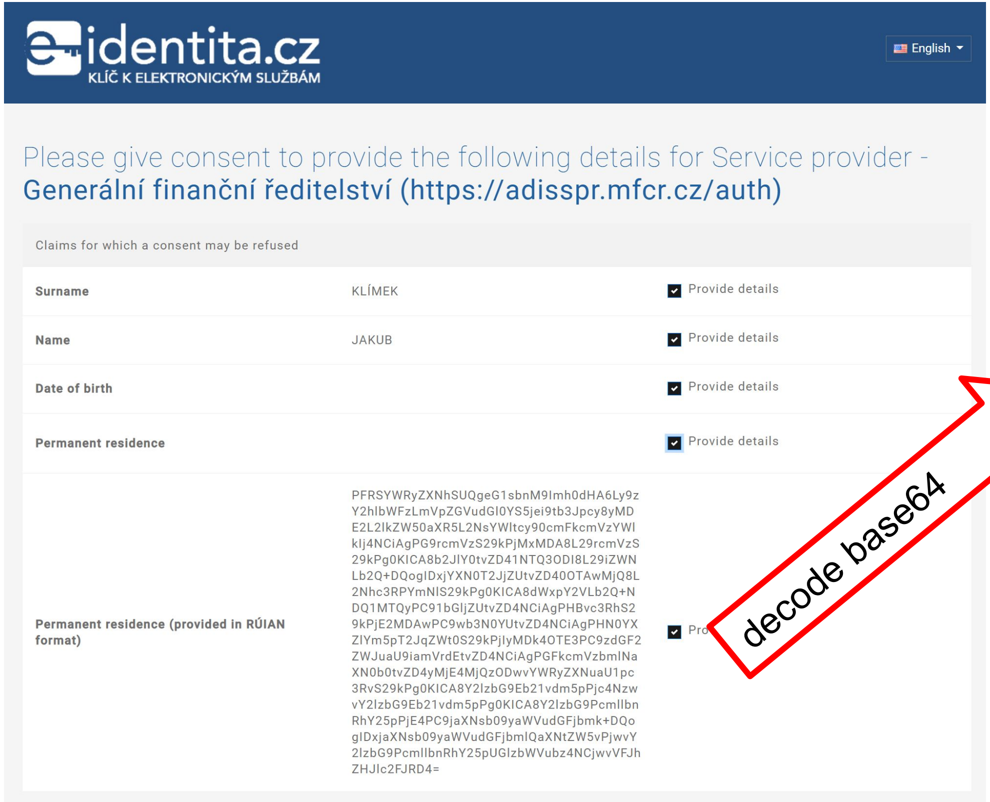
[W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes]

Most common built-in simple data types:

- `boolean`
- `anyURI`
- `date`, `time`, `dateTime`, `dateTimeStamp`
- `decimal`, `integer`, `double`
- `hexBinary`, `base64Binary`
- `gYear`



Base64 encoding - common in eGovernment



Please give consent to provide the following details for Service provider - **Generální finanční ředitelství (https://adisspr.mfcr.cz/auth)**

Claims for which a consent may be refused

Surname	KLÍMEK	☑ Provide details
Name	JAKUB	☑ Provide details
Date of birth		☑ Provide details
Permanent residence		☑ Provide details
Permanent residence (provided in RÚIAN format)	PFRSYWRyZXNhSUQgeG1sbnM9Imh0dHA6Ly9z Y2hlbWFzLmVpZGVudGl0YS5jei9tb3Jpcy8yMD E2L2lkZW50aXR5L2NsYWltcy90cmFkcmVzYWl kIj4NCiAgPG9rcmVzS29kPjMxMDA8L29rcmVzS 29kPg0KICA8b2JlY0tvZD41NTQ3ODI8L29iZWN Lb2Q+DQogIDxjYXN0T2JjZUtvZD40OTAwMjQ8L 2Nhc3RPYmNlS29kPg0KICA8dWxpY2VLb2Q+N DQ1MTQyPC91bGljZUtvZD4NCiAgPHBvc3RhS2 9kPjE2MDAwPC9wb3N0YUtvZD4NCiAgPHN0YX ZlYm5pT2JqZWt0S29kPjIyMDk4OTE3PC9zdGF2 ZWJuaU9iamVrdEtvZD4NCiAgPGFkcmVzbmlNa XN0b0tvZD4yMjE4MjQzODwvYWRyZXNuaU1pc 3RvS29kPg0KICA8Y2lzbG9Eb21vdm5pPjc4Nzw vY2lzbG9Eb21vdm5pPg0KICA8Y2lzbG9Pcmllbn RhY25pPjE4PC9jaXNsb09yaWVudGFjbmk+DQo gIDxjaXNsb09yaWVudGFjbmlQaXNtZW5vPjwvY 2lzbG9PcmllbnRhY25pUGlzbWVubz4NCjwvVFJh ZHJlc2FJRD4=	☑ Pro

decode base64

```
<TRadresaID
xmlns="http://schemas.eidentita.cz/moris/2016/ide
ntity/claims/tradresaid">
  <okresKod>3100</okresKod>
  <obecKod>554782</obecKod>
  <castObceKod>490024</castObceKod>
  <uliceKod>444142</uliceKod>
  <postaKod>17000</postaKod>
  <stavebniObjektKod>22097917</stavebniObjektKod>
  <adresniMistoKod>22128438</adresniMistoKod>
  <cisloDomovni>878</cisloDomovni>
  <cisloOrientacni>81</cisloOrientacni>

<cisloOrientacniPismeno></cisloOrientacniPismeno>
</TRadresaID>
```

base64Binary

XML Schema - Basic Principles

Definitions of:

- Data types
  - Simple (`simpleType`)
  - Complex (`complexType`)
- Elements (`element`)
  - Groups of elements (`group`)
- Attributes (`attribute`)
  - Groups of attributes (`attributeGroup`)

XSD - Element definition

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Catalog"/>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema01.xsd"
    />
```

✅

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema01.xsd"
    >
        <Dataset>
            <test/>
        </Dataset>
    </Catalog>
```

✅

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog1
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema01.xsd"
    />
```

❌

XSD - Element definition

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Catalog"/>
    <xs:element name="Catalog1"/>
</xs:schema>
```

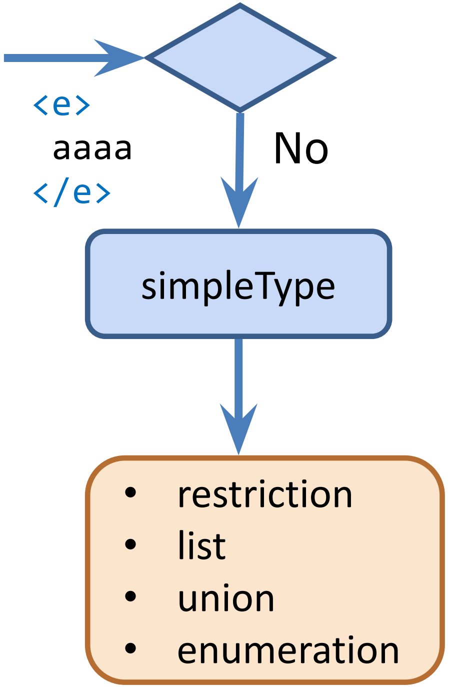
```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema02.xsd"
    />
```

✅

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog1
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema02.xsd"
    />
```

✅

subelements or attributes?



```
<e>
  aaaa
</e>
```

No

simpleType

- restriction
- list
- union
- enumeration

XSD - Element simple type usage

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Catalog" type="xs:boolean"/>
</xs:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema03.xsd"
    />
```

❌

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema03.xsd"
    >false</Catalog>
```

✅

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema03.xsd"
    >
        <Dataset>
            <test/>
        </Dataset>
    </Catalog>
```

❌

XSD - Simple type restriction

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:element name="Catalog">
        <xs:simpleType>
            <xs:restriction base="xs:integer">
                <xs:minInclusive value="42"/>
            </xs:restriction>
        </xs:simpleType>
    </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema04.xsd"
    >42</Catalog>
```

✅

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema04.xsd"
    >22</Catalog>
```

❌

XSD - Named type definition and usage

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name="atLeast42">
        <xs:restriction base="xs:integer">
            <xs:minInclusive value="42"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="Catalog" type="atLeast42"/>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema05.xsd"
    >42</Catalog>
```

✅

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema05.xsd"
    >22</Catalog>
```

❌

XSD - List

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name="listOfNumbers">
        <xs:list itemType="xs:integer"/>
    </xs:simpleType>
    <xs:element name="Catalog" type="listOfNumbers"/>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema05-2.xsd">
    42 33 11
</Catalog>
```

✅

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema05-2.xsd">

    42 test 11
</Catalog>
```

❌

XSD - Type union

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name="numberOrBoolean">
        <xs:union memberTypes="xs:boolean xs:integer"/>
    </xs:simpleType>
    <xs:element name="Catalog" type="numberOrBoolean"/>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema05-3.xsd">
    42
</Catalog>
```

✅

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema05-3.xsd">
    true
</Catalog>
```

✅

XSD - Enumeration

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
    <xs:simpleType name="number135">
        <xs:restriction base="xs:integer">
            <xs:enumeration value="1"/>
            <xs:enumeration value="3"/>
            <xs:enumeration value="5"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="Catalog" type="number135"/>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema05-3.xsd">
    42
</Catalog>
```

❌

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema05-4.xsd">
    3
</Catalog>
```

✅

subelements or attributes?

Yes

complexType

subelements?

```
<e>
 aaaa
</e>
```

No

simpleType

- restriction
- list
- union
- enumeration

```
<e a="##">
 aaaa
</e>
```

No

simpleContent

text content
attributes
no subelements

- restriction
- extension

XSD - Complex type - simple content + attribute

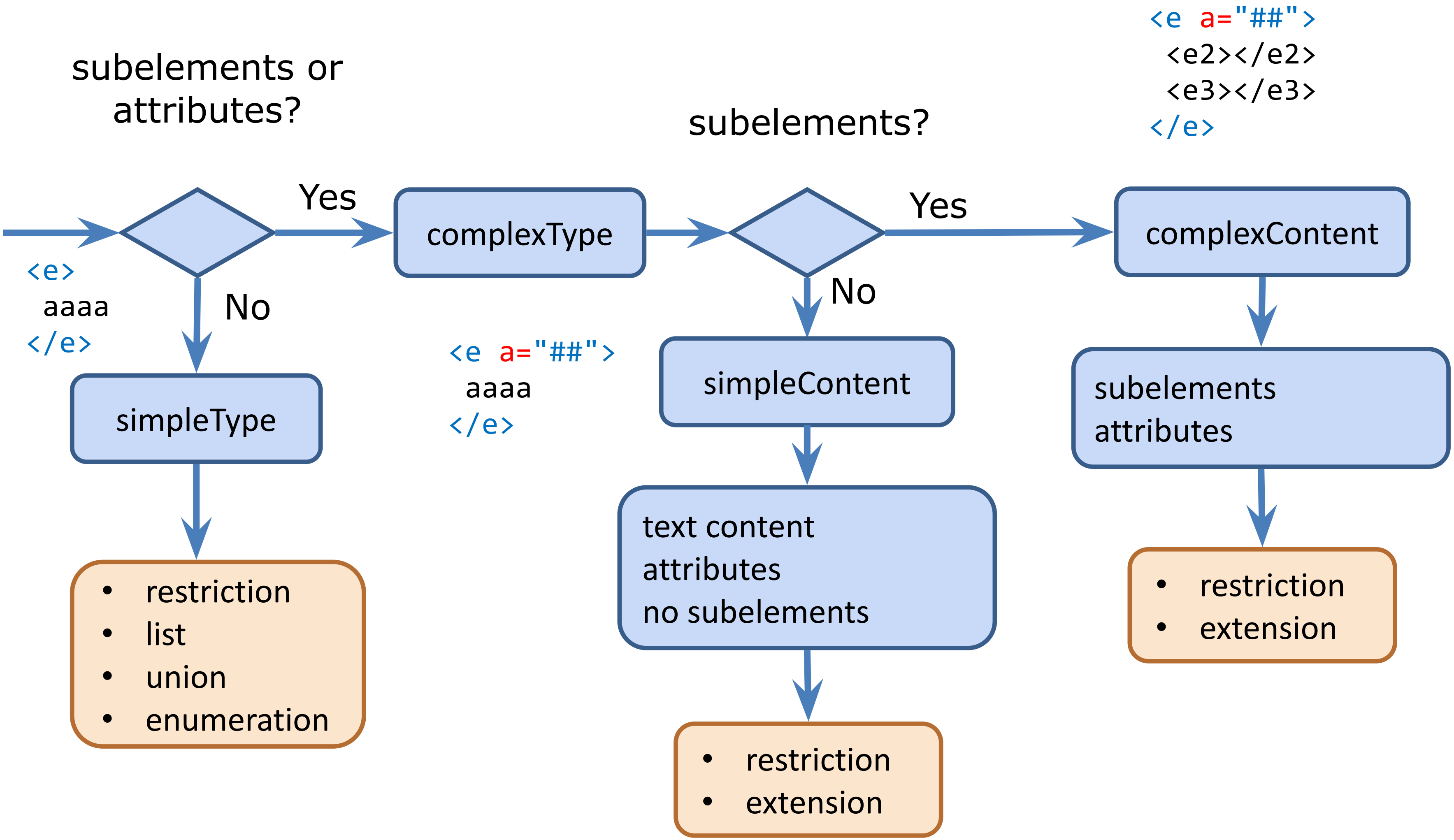
```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Catalog">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="numberOfPublishers"
                type="xs:integer"
                use="required"/>
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema06.xsd"
    >22</Catalog>
```

❌

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema06.xsd"
    numberOfPublishers="42"
    >someText</Catalog>
```

✅



subelements or attributes?

```
<e>
 aaaa
</e>
```

Yes

No

complexType

simpleType

- restriction
- list
- union
- enumeration

subelements?

```
<e a="##">
 aaaa
</e>
```

Yes

No

simpleContent

text content
attributes
no subelements

- restriction
- extension

```
<e a="##">
 <e2></e2>
 <e3></e3>
</e>
```

complexContent

subelements
attributes

- restriction
- extension

XSD - Complex type - sequence

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Catalog">
    <xs:complexType>
      <xs:sequence>
        <xs:element
          name="Dataset"
          minOccurs="1"
          maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema07.xsd"
    >22</Catalog>
```

❌

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema07.xsd"
    >
  <Dataset>test</Dataset>
  <Dataset><Distribution/></Dataset>
  <Dataset></Dataset>
  <Dataset/>
</Catalog>
```

✅

XSD - Complex type - sequence & element order

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Catalog">
    <xs:complexType>
      <xs:sequence>
        <xs:element
          name="Dataset"
          minOccurs="1"
          maxOccurs="unbounded"/>
        <xs:element
          name="DataService"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema08.xsd"
    >
  <Dataset>test</Dataset>
  <Dataset></Dataset>
  <DataService/>
  <Dataset/> ❌
</Catalog>
```

XSD - Complex type - sequence & element order

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Catalog">
    <xs:complexType>
      <xs:sequence>
        <xs:element
          name="Dataset"
          minOccurs="1"
          maxOccurs="unbounded"/>
        <xs:element
          name="DataService"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema08.xsd"
    >
  <Dataset>test</Dataset>
  <Dataset></Dataset>
  <DataService/>
</Catalog>
```

✅

XSD - Complex type - sequence & element order

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Catalog">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element
          name="Dataset"
          minOccurs="1"
          maxOccurs="unbounded"/>
        <xs:element
          name="DataService"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema09.xsd"
    >
  <Dataset>test</Dataset>
  <Dataset></Dataset>
  <DataService/>
  <Dataset/> ❌
</Catalog>
```

XSD - Complex type - sequence & element order

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Catalog">
    <xs:complexType>
      <xs:sequence maxOccurs="unbounded">
        <xs:element
          name="Dataset"
          minOccurs="1"
          maxOccurs="unbounded"/>
        <xs:element
          name="DataService"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema09.xsd"
    >
  <Dataset>test</Dataset>
  <Dataset></Dataset>
  <DataService/>
  <Dataset/>
  <DataService/>
</Catalog>
```

✅ ❔

XSD - Complex type - sequence & element order

```
<xs:element name="Catalog">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Datasets" minOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element
              name="Dataset"
              minOccurs="1"
              maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="DataServices" minOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element
              name="DataService"
              minOccurs="0"
              maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema10.xsd">
  <Datasets>
    <Dataset>test</Dataset>
    <Dataset/>
    <Dataset/>
  </Datasets>
  <DataServices>
    <DataService/>
    <DataService/>
  </DataServices>
</Catalog>
```

✅

XSD - Complex type - sequence & element order

```xml
<xs:element name="Catalog">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="Datasets" minOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element
              name="Dataset"
              minOccurs="1"
              maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
      <xs:element name="DataServices" minOccurs="1">
        <xs:complexType>
          <xs:sequence>
            <xs:element
              name="DataService"
              minOccurs="0"
              maxOccurs="unbounded"/>
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema10.xsd">
  <Datasets>
    <Dataset>test</Dataset>
    <Dataset/>
    <Dataset/>
  </Datasets>
  <DataServices/>
</Catalog>
```

✅

XSD - Complex type - choice

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Catalog">
    <xs:complexType>
      <xs:choice>
        <xs:element name="Dataset"/>
        <xs:element name="DataService"/>
      </xs:choice>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema11.xsd">
  <Dataset/>
</Catalog>
```

✅

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema11.xsd">
  <DataService>test</DataService>
</Catalog>
```

✅

XSD - Complex type - all

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Catalog">
    <xs:complexType>
      <xs:all>
        <xs:element name="Dataset"/>
        <xs:element name="DataService"/>
      </xs:all>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema12.xsd">
  <DataService/>
  <Dataset/>
</Catalog>
```

✅

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema12.xsd">
  <Dataset/>
  <DataService/>
</Catalog>
```

✅

XSD - Element references

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="Dataset">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="name"/>
        <xs:element name="description"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="Catalog">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="Dataset"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalog
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="Schema13.xsd">
  <Dataset>
      <name>My dataset</name>
      <description>My dataset description</description>
  </Dataset>
</Catalog>
```

✅

XML Schema - namespaces

Untitled1.xsd

```
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://tempuri.org/"
elementFormDefault="qualified">
  <xs:element name="Add">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="intA"
                    type="xs:int"/>
        <xs:element name="intB"
                    type="xs:int"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

document.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<n1:Add
xmlns:n1="http://tempuri.org/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tempuri.org/ Untitled1.xsd">

  <n1:intA>1</n1:intA>
  <n1:intB>3</n1:intB>

</n1:Add>
```

XSD - namespaces

Untitled1.xsd

```
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://tempuri.org/">

  <xs:element name="Add">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="intA" type="xs:int"
                    form="qualified"/>
        <xs:element name="intB" type="xs:int"
                    form="qualified"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

document.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<n1:Add
xmlns:n1="http://tempuri.org/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tempuri.org/ Untitled1.xsd">

  <n1:intA>1</n1:intA>
  <n1:intB>3</n1:intB>

</n1:Add>
```

XSD - namespaces

Untitled1.xsd

```
<xs:schema
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://tempuri.org/">

  <xs:element name="Add">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="intA" type="xs:int"
                    form="unqualified"/>
        <xs:element name="intB" type="xs:int"
                    form="qualified"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

document.xml

```
<?xml version="1.0" encoding="UTF-8"?>

<n1:Add
xmlns:n1="http://tempuri.org/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://tempuri.org/ Untitled1.xsd">

  <intA>1</intA>
  <n1:intB>3</n1:intB>

</n1:Add>
```

More XML Schema - not covered here

- combining multiple schemas: `import`, `include`, `override`
- re-using existing types: `complexContent` extensions and restrictions
- `any` - allowing content unspecified by the XSD
- substitution groups
- Mixed content (text and elements)
- (element) `groups`, `attributeGroups`
- assertions - integrity constraints
- ID/IDREFs, key, key-refs
- ...