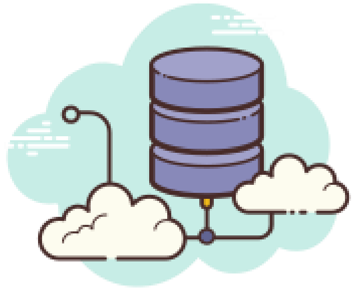
INF115 Lecture 16:
XML and JSON
Part 2

Adriaan Ludl
Department of Informatics
University of Bergen

Spring Semester 2021

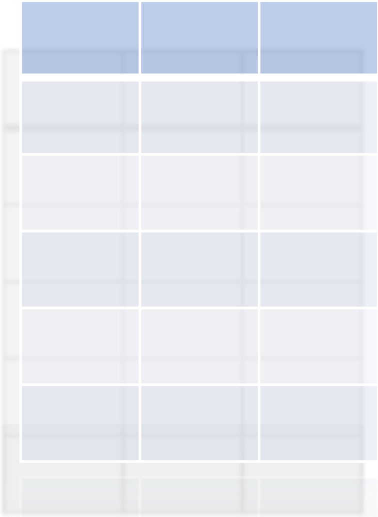
Chapter 14: *XML and JSON – Part 2*



Learning Goals:

- Create **XML documents** with ***elements*** and ***attributes***.
- Describe the **proper structure** of ***XML documents*** with DTD and XML Schema.

- **Use XML style sheets** and **query language**.
- Be able to **create JSON documents**.
- Know the **use of XML and JSON in web services**.



XML vs other database ?

- What can XML do that a CSV (tables, spreadsheets) cannot do ?
- Text human readable vs binary ?
- Example: Save a computer game state as an XML file
  - https://sourceforge.net/projects/gamexml/

XPath – a query language for XML

We want to create rules that « **rewrite** » XML documents,

So it is useful to be able to talk about parts of an XML document.

- Example: Refer to the contents of the <title> element
  that are below the first instance of the <book> element.

XPath is a **notation** to refer to parts of an XML document

- XPath is based on **path expressions**
- XPath contains a **library of standard features**
- XPath is a **W3C Standard**

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<book>
  <author>Tove Jansson</ author >
  <title>Trollvinter</ title >
</book>
```

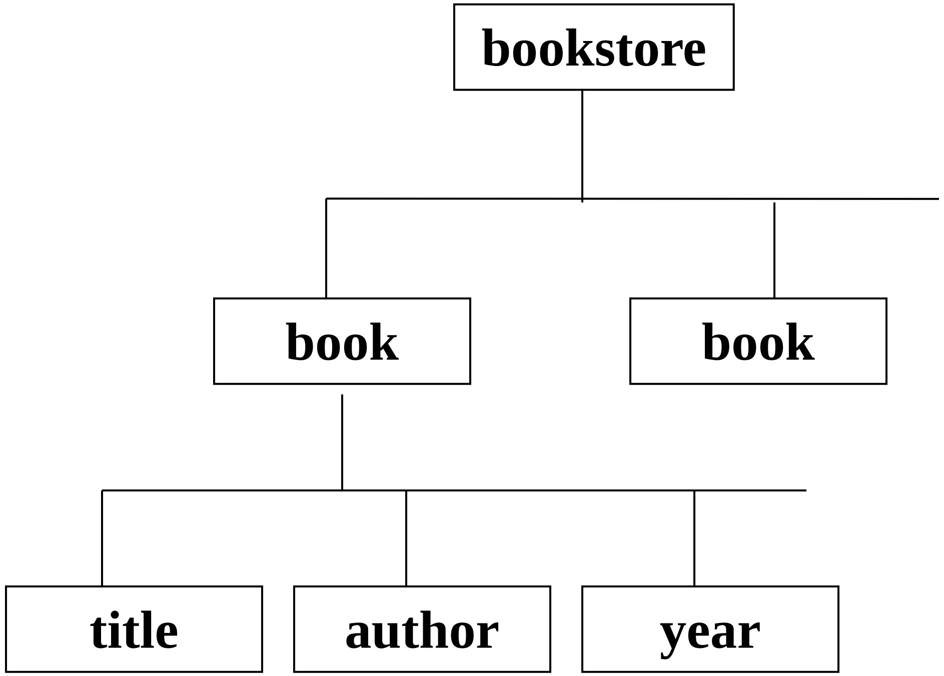
Tree structures of XML documents

Nodes in tree can be described as

- Parents, children, siblings

In general a node can have

- Ancestors
- Descendents



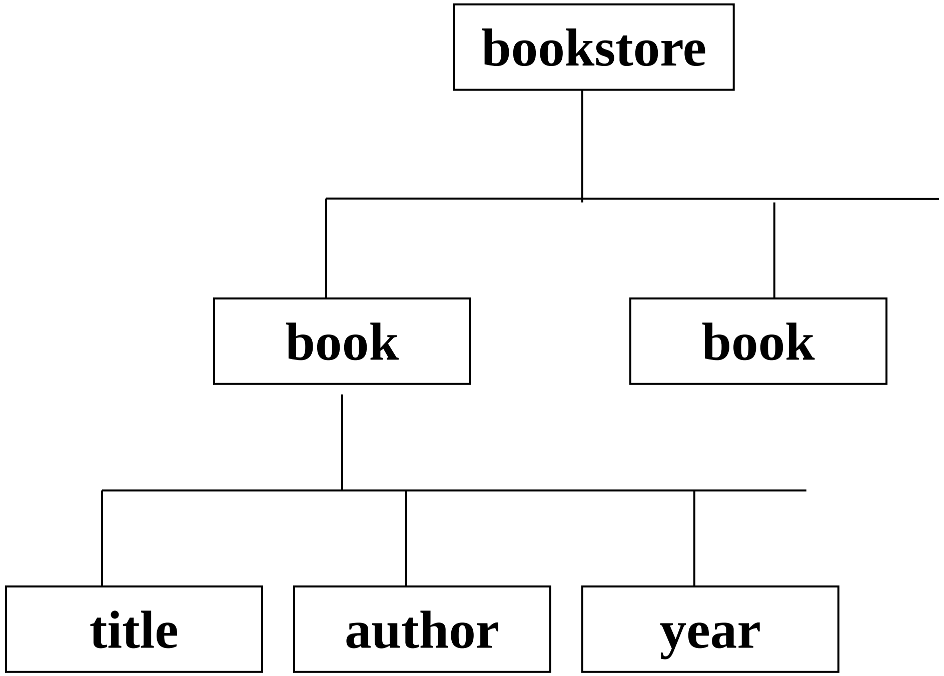
Tree structures of XML documents

Nodes in tree can be described as

- Parents, children, siblings

In general a node can have

- Ancestors
- Descendents



- Thus, a **path expression** refers to the path under which an item can be found.

XPath *path expressions*

Path Expression	Result
bookstore	Selects **all nodes with the name** "bookstore"
/bookstore	Selects the **root element bookstore** **Note:** If the path starts with a *slash* (/) it always represents an *absolute path* to an element!
bookstore/book	Selects all book elements that are **children** of bookstore
//book	Selects all book elements **no matter where they are in the document**
bookstore//book	Selects all book elements that are **descendants** of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects **all attributes** that are **named lang**

From https://www.w3schools.com/xml/xpath_syntax.asp

XPath *path expressions* (2)

Path Expression	**Result**
/bookstore/*	Selects **all the child element nodes of the bookstore** element
//*	Selects **all elements in the document**
//title[@*]	Selects all title elements which have **at least one attribute** of any kind
//title[@lang]	Selects all the **title elements that have an attribute named lang**
//title[@lang='en']	Selects all the title elements that have a "lang" attribute **with a value of "en"**
/bookstore/book[3]	Selects the third book element that is a child of the bookstore element.
/bookstore//book..	Selects the **parent node of the book element** under bookstore
/table/message@msg	Selects **all msg attributes of messages** directly under table
text()	Selects all text nodes directly under the active node

From https://www.w3schools.com/xml/xpath_syntax.asp

XML style sheet

- XSL = eXstensible Stylesheet Language
- **XSLT** = XSL **Transformations**
- XSLT rewrites (transforms) an XML document into another XML document - or HTML
- XSLT uses XPath to navigate XML documents

Websites:

Learn-yourself pages:	www.w3schools.com/xsl
Standard / Specification:	www.w3.org/Style/XSL/

Transform XML with XSLT

With XSLT we can create **rules for converting XML to HTML**

❖ **General format of such rules**:

- **XML pattern => HTML code + "XML extraction"**
- The right side of the rule consists of HTML interspersed with code to pick elements from the XML document

❖ **Rules can make structural changes**

- Can select some items
- Can sort items

❖ **How is XPath used in XSL?**

- Used as a **pattern on the left side of rules**
- Also used **to pick out items on the right**

Transform XML with XSLT

With XSLT we can create **rules for converting XML to HTML**

❖ **General format of such rules**:

- **XML pattern => HTML code + "XML extraction"**
- The right side of the rule consists of HTML interspersed with code to pick elements from the XML document

❖ **Rules can make structural changes**

- Can select some items
- Can sort items

❖ **How is XPath used in XSL?**

- Used as a **pattern on the left side of rules**
- Also used **to pick out items on the right**

Transform XML with XSLT

With XSLT we can create **rules for converting XML to HTML**

- **General format of such rules**:
  - **XML pattern => HTML code + "XML extraction"**
  - The right side of the rule consists of HTML interspersed with code to pick elements from the XML document
- **Rules can make structural changes**
  - Can select some items
  - Can sort items
- **How is XPath used in XSL?**
  - Used as a **pattern on the left side of rules**
  - Also used **to pick out items on the right**

Example of an XSLT rule

- **Pick** <pnr>, <fornavn>, <etternavn> from <**person**>.
- **Set** <pnr> as heading (h2).
- **Put** first name and last name in a paragraph (p), with a blank character in between.

```
<xsl:template match="person">
  <h2>Person <xsl:value-of select="pnr"/></h2>
  <p>
    Navn:
    <xsl:value-of select="fornavn"/>
    <xsl:text>█</xsl:text>
    <xsl:value-of select="etternavn"/>
  </p>
</xsl:template>
```

Quizz on ***XML and JSON*** (part 3)

Please answer the practice quizz on mitt.uib now ☺

(you can take it again later if you want)

Link:

➢ https://mitt.uib.no/courses/27455/quizzes

XML and databases

Applications:

- XML for **dynamic web data**
- XML as the **transfer format** for data

Store XML in databases:

- As CLOB or XMLType values
- Loading data

Queries against XML data:

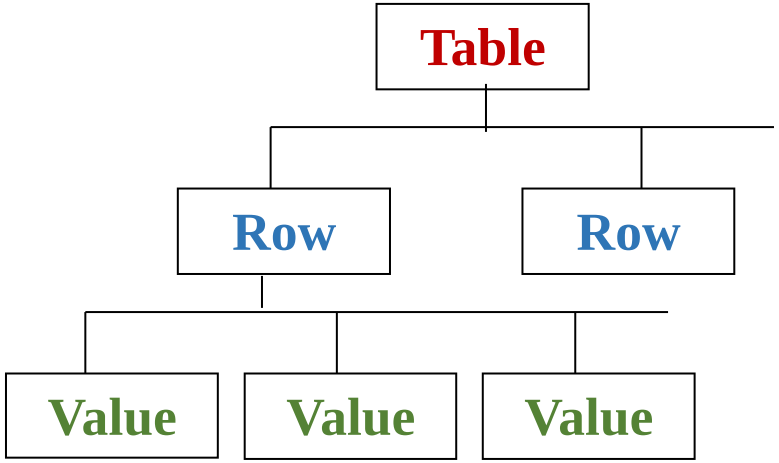
- **XPath**
- XQuery is a query language with syntax similar to SQL

Generate XML from table data

Structured data in XML *variant 1*

XML representation of **row collections:**

- **Root element** = **table** name
- **Element name** = name of database **row** or **column**
- **Text** in the element (between the tags) = **value**



```
<?xml version="1.0" encoding="UTF-8"?>
<Studenter>
  <Student>
    <Studentnr>1</Studentnr>
    <Navn>Ole Hansen</Navn>
    …
  </Student>
  <Student>
    <Studentnr>2</Studentnr>
  …
  </Student>
</Studenter>
```

Structured data in XML *variant 2*

As an alternative we put **data in attributes**

```
<Student Studnr="012345", Fornavn="Ole",
       Etternavn="Hansen", Gate="Bøgata",
       Husnr="23", Postnr="3800",
       Poststed="Bø">
</Student>
```

- Element *name* = **table** name
- Attribute *name* = name of table **column**
- Attribute *value* = **value**

Semi-structured data

Characteristics

- Not a table structure, more a **document structure**
- **Form-free** (does not follow the given type structure)

Requirements

- Want to **store such data** in «databases»
- Want to be able to **search** (ad hoc) with a **query language**

How to store semi-structured data?

- ➢ As **XML documents** in files
- ➢ **Normalized table structure** in relational database
- ➢ **Hybrid solution**: XML documents as values in relational bases
- ➢ **Object relational databases** with XML support
- ➢ **XML** databases
- ➢ **NoSQL** databases

Semi-structured data

Characteristics

- Not a table structure, more a **document structure**
- **Form-free** (does not follow the given type structure)

Requirements

- Want to **store such data** in «databases»
- Want to be able to **search** (ad hoc) with a **query language**

How to store semi-structured data?

- ➢ As **XML documents** in **files**
- ➢ **Normalized table structure** in relational database
- ➢ **Hybrid solution**: XML documents as values in relational databases
- ➢ **Object relational databases** with XML support
- ➢ **XML** databases
- ➢ **NoSQL** databases

15 minute break!
Lecture resumes at 11:00

XML databases

Questions/Issues:

- Is XML suitable for **permanent data storage**?
- Can XML documents replace database systems?

What is required of an XML database?

- Be able to **describe logical data structure in XML** with, among other things:
  - **Valid** elements and attributes
  - **Data types** and **element structure**
  - **Primary keys** and **foreign keys**, **validation rules**
- An XML-based **query language** for searching XML data
- Ability to **update XML data**

Example Loading XML into MySQL

Create an XML file, saved at “D:\\test.xml”:

```
<plugin plugin_name="tree">
  <title>Test</title>
  <description>some description</description>
     <file>test.tmp</file>
…
</plugin>
```

- Load it into MySQL:

```
LOAD XML LOCAL INFILE "D:\\test.xml"
INTO TABLE mytable
ROWS IDENTIFIED BY '<plugin>';
```

XMLType in Oracle

A dedicated data type **XMLType** for storing XML documents:

```
CREATE TABLE XML_DOCUMENTS (
  DOC_NO      NUMBER,
  MY_DOC      XMLTYPE
);
```

Insert XML data :

```
INSERT INTO XML_DOCUMENTS (DOC_NO, MY_DOC)
  VALUES ( 1,
    XMLTYPE.createXML(
      '<?xml version="1.0" encoding="UTF-8"?>
       <melding dato="30.04.2019" rom="5-116">
          <avsender fnavn="Kari" enavn="Lie"/>
          <beskjed>Møte om 5 min!</beskjed>
      </melding>'
    )
  );
```

https://mariadb.com/kb/en/connect-xml-table-type/

The Datatype XMLType

- **Useful operations** on XMLType :
  - EXTRACT
  - EXTRACTVALUE
  - EXISTSNODE
  - UPDATEXML
- Retrieve a **subtree** :

```
SELECT EXTRACT(MY_DOC, '\bok\kap')
FROM XML_DOCUMENTS;
```

- **Update parts** of an XML document :

```
UPDATE XML_DOCUMENTS
SET MY_DOC = UPDATEXML(MY_DOC,'\bok\kap\text()',
                       'Ny tittel')
WHERE EXISTSNODE(MY_DOC,\bok\kapittel[nr="2"])=1;
```

The Datatype XMLType

- **Useful operations** on XMLType :
  - EXTRACT
  - EXTRACTVALUE
  - EXISTSNODE
  - UPDATEXML
- Retrieve a **subtree** :

```
SELECT EXTRACT(MY_DOC, ‘\book\chap')
FROM XML_DOCUMENTS;
```

- **Update parts** of an XML document :

```
UPDATE XML_DOCUMENTS
SET MY_DOC = UPDATEXML(MY_DOC,'\bok\kap\text()',
                              'Ny tittel')
WHERE EXISTSNODE(MY_DOC,\bok\kapittel[nr="2"])=1;
```

The Datatype XMLType

- **Useful operations** on XMLType :
  - EXTRACT
  - EXTRACTVALUE
  - EXISTSNODE
  - UPDATEXML
- Retrieve a **subtree** :

```
SELECT EXTRACT(MY_DOC, ‘\book\chap')
FROM XML_DOCUMENTS;
```

- **Update parts** of an XML document :

```
UPDATE XML_DOCUMENTS
SET MY_DOC = UPDATEXML(MY_DOC,’\book\chap\text()',
                       ‘New title')
WHERE EXISTSNODE(MY_DOC,\bok\kapittel[nr="2"])=1;
```

Generate XML from a database ("regular tables")

- Functions for “*decorating*” a query result with markup tags

```
SELECT XMLElement("Navn", s.navn)
FROM student;
```

```
XMLELEMENT("NAVN",NAVN)
-----------------------
<Navn>EVA</Navn>
<Navn>OLA</Navn>
....
```

```
SELECT XMLForest(s.snr, s.navn, s.adresse, s.fdato)
FROM student s
WHERE s.snr = 1;
```

```
XMLFOREST(SNR,NAVN,ADRESSE,FDATO)
------------------------------------
<SNR>1</SNR>
<NAVN>EVA</NAVN>
<ADRESSE>AVEIEN 1</ADRESSE>
<FDATO>01.01.1972</FDATO>
```

XML for data transfer

Why use XML as a data transfer format?

- **Flexible**: Expandable syntax
- **Self-descriptive** (item name)
- "Open": **Readable** for machines and people
- Good tools for **automating** work

But:

- There are **many other good formats**
- Contains a lot of "unnecessary" data that **takes up space**
- "Self-descriptive" is a **subjective** point of view:
  - Humans can guess the meaning of data based on element names - machines cannot (yet ...)

XML for data transfer

Why use XML as a data transfer format?

- **Flexible**: Expandable syntax
- **Self-descriptive** (item name)
- "Open": **Readable** for machines and people
- Good tools for **automating** work

But:

- There are **many other good formats**
- Contains a lot of "unnecessary" data that **takes up space**
- "Self-descriptive" is a **subjective** point of view:
  - Humans can guess the meaning of data based on element names - machines cannot (yet …)

Quizz on *XML and JSON* (part 4)

Please answer the practice quizz on mitt.uib now ☺

(you can take it again later if you want)

Link:

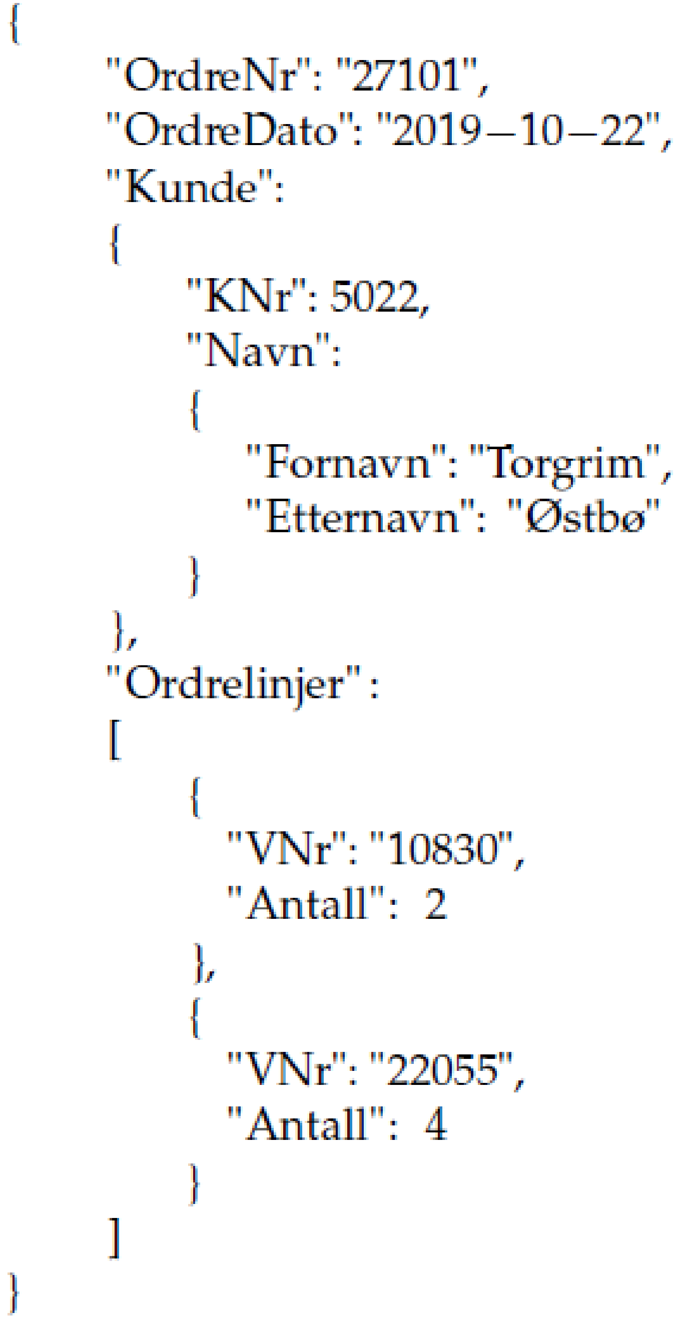
➢ https://mitt.uib.no/courses/27455/quizzes

JSON

JavaScript Object Notation

- **Text transfer format** widely used in **web solutions**.
- Can **describe semi-structured data**, in the same way as XML.
- **Easy to process** in many programming languages,

  including JavaScript.



```
{
      "OrdreNr": "27101",
      "OrdreDato": "2019−10−22",
      "Kunde":
      {
            "KNr": 5022,
            "Navn":
            {
                  "Fornavn": "Torgrim",
                  "Etternavn":  "Østbø"
            }
      },
      "Ordrelinjer" :
      [
            {
               "VNr": "10830",
               "Antall":  2
            },
            {
               "VNr": "22055",
               "Antall":  4
            }
      ]
}
```

Web Service

A set of **operations** ("subprograms") that an **application can "call on"** (use) over the Internet.

Motivation:

1. **Publish "live" data** that other systems can use.
2. **Integrate systems** from different vendors.
3. **Move data** (regularly) from one (database) system to another.

Based on **open protocols** (XML, JSON, HTTP)

They work across programming languages and platforms

- Example: A Java client can use a .NET service
- Two **technologies** for building web services: **SOAP** and **REST**
  - **SOAP**: *Simple Object Access Protocol* supports XML, Security, ACID Transactions
  - **REST**: *Representational State Transfer* scalability, more formats (XML, JSON …)

https://spf13.com/post/soap-vs-rest/

Web Service

A set of **operations** ("subprograms") that an **application can "call on"** (use) over the Internet.

Motivation:

1. **Publish "live" data** that other systems can use.
2. **Integrate systems** from different vendors.
3. **Move data** (regularly) from one (database) system to another.

Based on **open** **protocols** (XML, JSON, HTTP, …)

They work across programming languages and platforms:

- Example: A Java client can use a .NET service.

Two **technologies** for building web services: **SOAP** and **REST**

- **SOAP**: *Simple Object Access Protocol* supports XML, Security, ACID Transactions
- **REST**: *Representational State Transfer* offers scalability, more formats (XML, JSON …)

https://spf13.com/post/soap-vs-rest/

Example applications

Weather data:

- A local newspaper wants to integrate **local weather data** from *yr.no* on its **website**.
- The newspaper's **web solution calls up a web service** from *yr.no*, and gives **geographical parameters** for local towns.
- The **web service** of *yr.no* delivers fresh weather data in **XML format**, which the newspaper **visually adapts** to its web pages.

Student information system:

- Information about studies, courses, students and subject teachers is updated and stored in a student information system.
- Data is transferred daily to an e-learning system (e.g. Canvas), timetable system (eg TimeEdit) and online study reviews.

Example applications

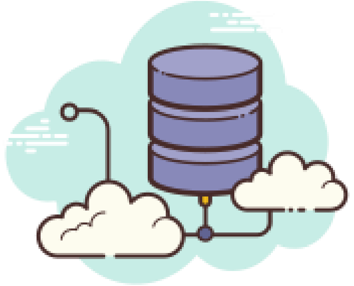
Weather data:

- A local newspaper will integrate **local weather data** from *yr.no* on its website.
- The newspaper's **web solution calls up a web service** from *yr.no*, and gives **geographical parameters** for local towns.
- The **web service** of *yr.no* delivers fresh weather data in **XML format**, which the newspaper visually adapts to its web pages.

Student information system:

- Information about **studies**, **courses**, **students** and **subject lecturers** is updated and stored in a student information system.
- **Data** is **transferred daily** to an e-learning system (e.g. Canvas), timetable system (e.g. TimeEdit) and online study counselling.

Summary: XML and JSON



- **XML** = **eXtensible Markup Language**
    - **Elements** and **attributes**
    - Preamble, namespace, processing instructions
  - Describe **valid** XML with **DTD** or **XML Schema**
  - Transformation and presentation: **XSLT**
  - **XML and databases**
    - **Structured** and **semi-structured** data
    - **Storing** XML **data** in databases
    - XML as a **transfer format**
    - **Query** languages on XML
- **JSON** = **JavaScript Object Notation**
  - Describe **valid** JSON: **JSON Schema**
- **Web services**: SOAP, REST