Week 02: XML, schema and validation, DOM

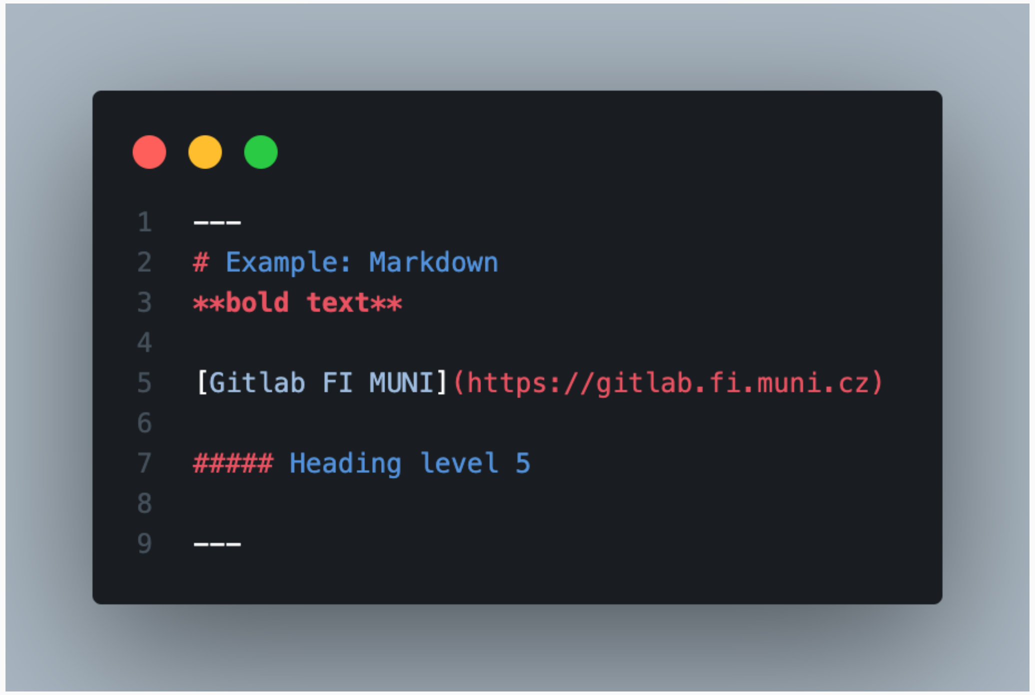
Agenda

- Markup languages
- XML basics
- XML schema
- DOM
- Short demo
- Hands on: Iteration 01

Let's dive into it!

Markup languages (recap)

- natural language + **special constructs** ("marks")
- for instance **HTML, Markdown, TeX**
- easily readable for both computers as well as humans


```
---
# Example: Markdown
**bold text**

[Gitlab FI MUNI](https://gitlab.fi.muni.cz)

##### Heading level 5

---
```

Example: Markdown

bold text

[Gitlab FI MUNI](https://gitlab.fi.muni.cz)

Heading level 5

XML

- eXtensible Markup Language
- data exchange format
- translations
- web scrapping
- .xml file extension

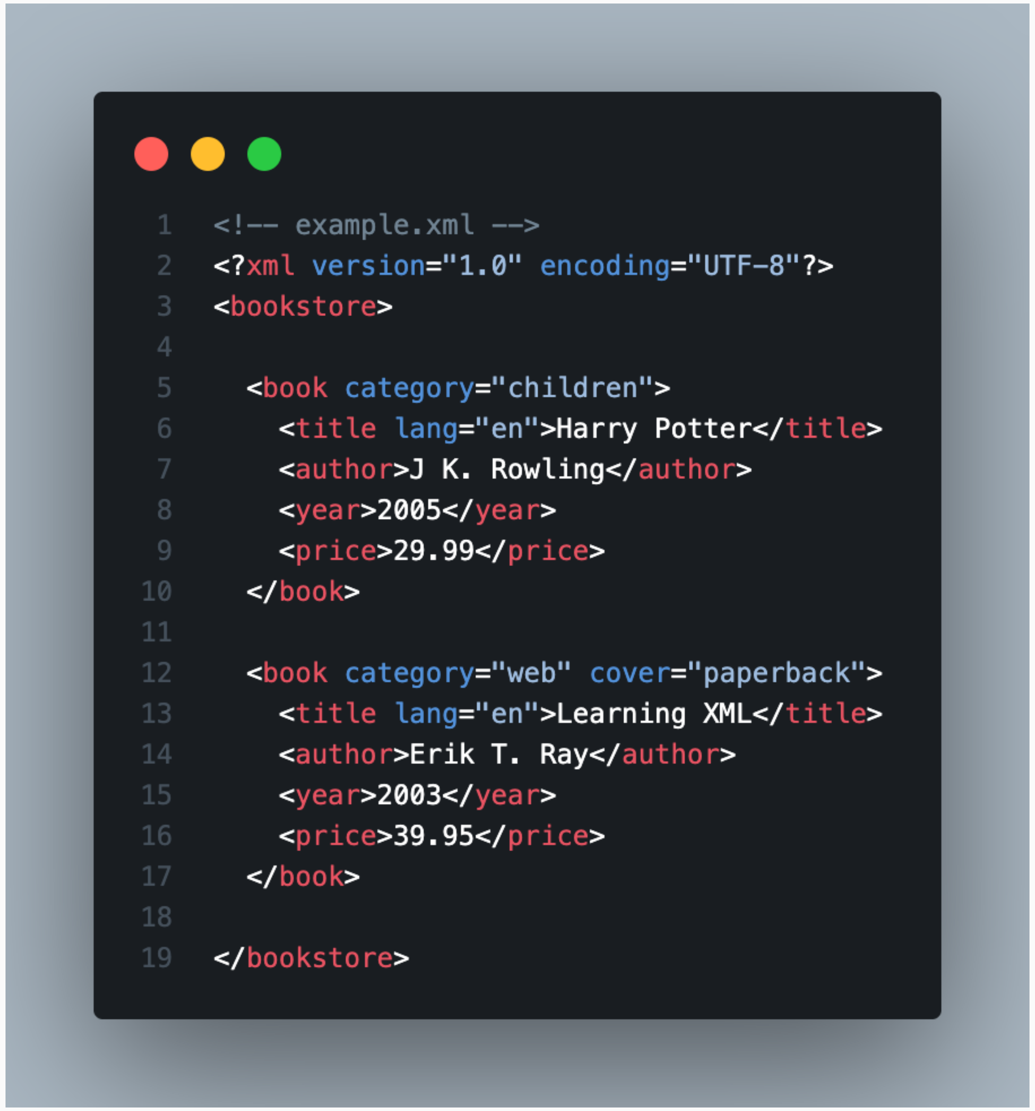
```xml
<!-- example.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>

  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>

  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>

</bookstore>
```

XML document structure



```xml
<!-- example.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>

  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>

  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>

</bookstore>
```

- comment
- processing instruction
- root element
- child/nested elements
- start/end tags
- text node
- attribute

Note: Elements are also nodes.

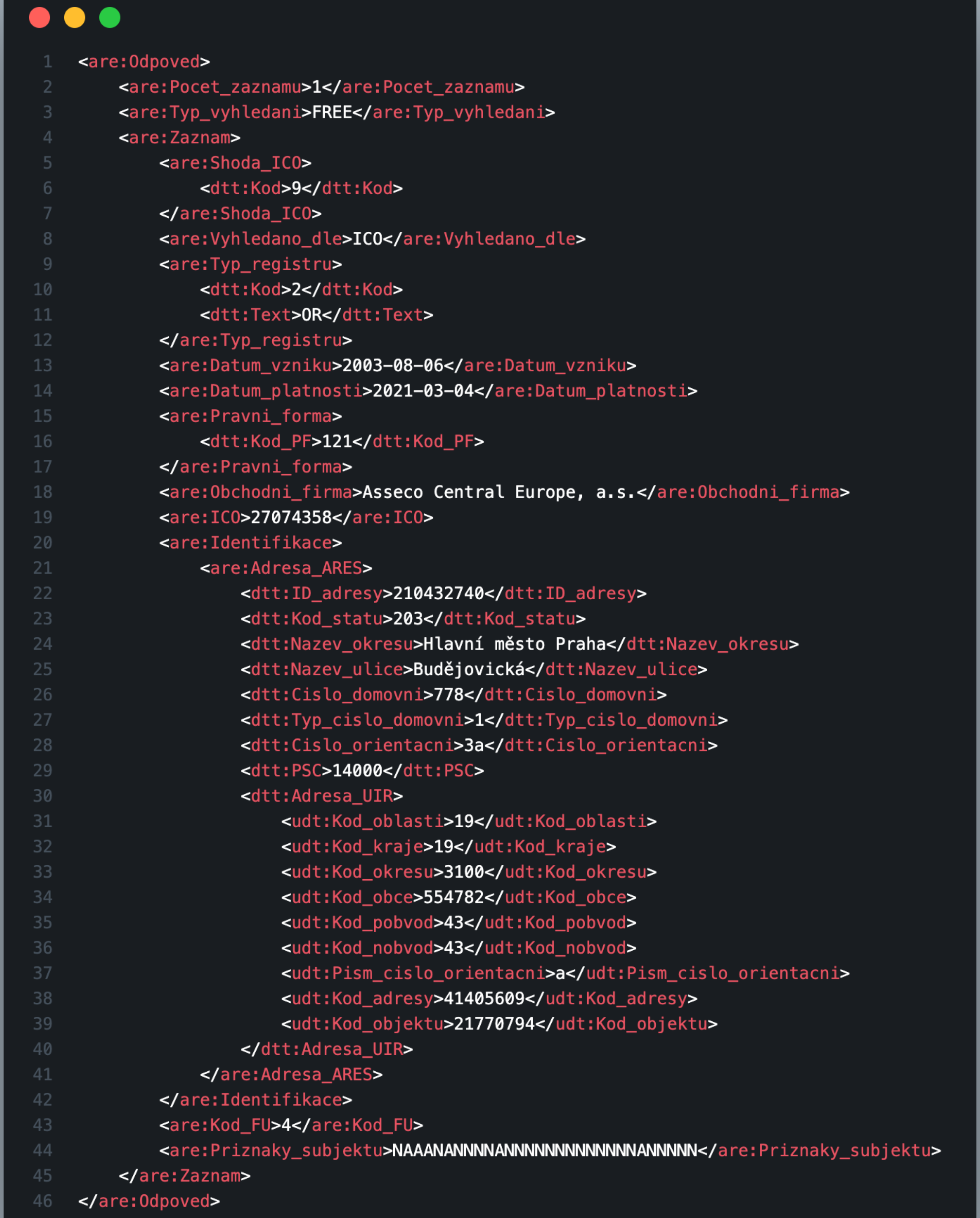
Basic rules

- all elements must have an **end tag** OR be empty and **self-closing**
- all elements must be properly **nested** (overlapping is not allowed)
- all attribute values must be enclosed in **quotes**
- each document must have a unique **root element**

Naming conventions and names of elements are free to choose.

But remember,

"with great power comes great responsibility."



```
<are:Odpoved>
    <are:Pocet_zaznamu>1</are:Pocet_zaznamu>
    <are:Typ_vyhledani>FREE</are:Typ_vyhledani>
    <are:Zaznam>
        <are:Shoda_ICO>
            <dtt:Kod>9</dtt:Kod>
        </are:Shoda_ICO>
        <are:Vyhledano_dle>ICO</are:Vyhledano_dle>
        <are:Typ_registru>
            <dtt:Kod>2</dtt:Kod>
            <dtt:Text>OR</dtt:Text>
        </are:Typ_registru>
        <are:Datum_vzniku>2003-08-06</are:Datum_vzniku>
        <are:Datum_platnosti>2021-03-04</are:Datum_platnosti>
        <are:Pravni_forma>
            <dtt:Kod_PF>121</dtt:Kod_PF>
        </are:Pravni_forma>
        <are:Obchodni_firma>Asseco Central Europe, a.s.</are:Obchodni_firma>
        <are:ICO>27074358</are:ICO>
        <are:Identifikace>
            <are:Adresa_ARES>
                <dtt:ID_adresy>210432740</dtt:ID_adresy>
                <dtt:Kod_statu>203</dtt:Kod_statu>
                <dtt:Nazev_okresu>Hlavní město Praha</dtt:Nazev_okresu>
                <dtt:Nazev_ulice>Budějovická</dtt:Nazev_ulice>
                <dtt:Cislo_domovni>778</dtt:Cislo_domovni>
                <dtt:Typ_cislo_domovni>1</dtt:Typ_cislo_domovni>
                <dtt:Cislo_orientacni>3a</dtt:Cislo_orientacni>
                <dtt:PSC>14000</dtt:PSC>
                <dtt:Adresa_UIR>
                    <udt:Kod_oblasti>19</udt:Kod_oblasti>
                    <udt:Kod_kraje>19</udt:Kod_kraje>
                    <udt:Kod_okresu>3100</udt:Kod_okresu>
                    <udt:Kod_obce>554782</udt:Kod_obce>
                    <udt:Kod_pobvod>43</udt:Kod_pobvod>
                    <udt:Kod_nobvod>43</udt:Kod_nobvod>
                    <udt:Pism_cislo_orientacni>a</udt:Pism_cislo_orientacni>
                    <udt:Kod_adresy>41405609</udt:Kod_adresy>
                    <udt:Kod_objektu>21770794</udt:Kod_objektu>
                </dtt:Adresa_UIR>
            </are:Adresa_ARES>
        </are:Identifikace>
        <are:Kod_FU>4</are:Kod_FU>
        <are:Priznaky_subjektu>NAAANANNNNANNNNNNNNNNNNNANNNNN</are:Priznaky_subjektu>
    </are:Zaznam>
</are:Odpoved>
```

To avoid

Element or attribute?

```xml
<book>
  <category>children</category>
  <title>
    <lang>en</lang>
    Harry Potter
  </title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

vs.

```xml
<book category="children">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

Element or attribute?

Attribute value:

- cannot be further structured
- "atomic" value
- keeps an additional info of another element
- for instance, **lang**

Note: do not use too many attributes on a single element. It can get hard to read very quickly. Let's say that 3-4 attributes are maximum.

Element:

- even if nested, it is a meaningful structure itself
- usually structured
- contains any number of nodes
- for instance, **person**

XML Schema

- the structure of an XML document
- what elements and attributes are **allowed** to appear, in which **order**, how many times...
- **data types** of elements and attributes
- **default** or **fixed** values for elements/attributes
- **namespaces**
- follows XML syntax itself
- **.xsd** file extension

*Note: we **validate** an XML document against a schema. So a document is **valid** if it conforms to a given schema.*

```xml
<!-- example.xsd -->
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

<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>
```

Definition Header

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

Element

This is a simple element. It can contain only **text**.

```
<xs:element name="element_name" type="element_type" />
```

Limiting occurences

```
<xs:element name="element_name" minOccurs="1" maxOccurs="unbounded" />
```

Types: simple types

Basic types
xs:string
xs:decimal
xs:integer
xs:boolean
xs:date
xs:time
...

Types: simple types

- definition inside of a **xs:simpleType** element
- user defined types
- restrictions, unions, enumerations

```xml
<!-- Example 01: base restriction -->
<xs:simpleType name="typeName">
   <xs:restriction base="baseTypeName"> ... </xs:restriction>
</xs:simpleType>

<!-- Example 02: content length restriction -->
<xs:simpleType name="typeName">
    <xs:restriction base="xs:string">
        <xs:maxLength value="32"/>
    </xs:restriction>
</xs:simpleType>

<!-- Example 03: regex restriction -->
<xs:simpleType name="isbnType">
    <xs:restriction base="xs:string">
        <xs:pattern value="[0-9]{10}"/>
    </xs:restriction>
</xs:simpleType>
```

```xml
<!-- Example 04: enumeration restriction with usage -->
<xs:simpleType name="grade">
    <xs:restriction base="xs:NMTOKEN">
        <xs:enumeration value="A" />
        <xs:enumeration value="B" />
        <xs:enumeration value="C" />
    </xs:restriction>
</xs:simpleType>

<xs:element name="Course">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="Grade" type="grade" />
        </xs:sequence>
    </xs:complexType>
</xs:element>
```

Attributes

```xml
<!-- Example 01: with default value -->
<xs:attribute name="lang" type="xs:string" default="EN" />

<!-- Example 02: with required flag -->
<xs:attribute name="lang" type="xs:string" use="required" />
```

Complex types

- user defined
- definition inside of a **xs:complexType** element
- **xs:sequence** => all child elements, the **order is specified**
- **xs:all** => all child elements, the **order is not important**
- **xs:choice** => **only one** child element
- ...

```xml
<!-- Sequence -->
<xs:element name="elementName">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="elem1" type="xs:string"/>
            <xs:element name="elem2" type="xs:string"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>

<!-- All -->
<xs:element name="elementName">
    <xs:complexType>
        <xs:all>
            <xs:element name="elem1" type="xs:string"/>
            <xs:element name="elem2" type="xs:string"/>
        </xs:all>
    </xs:complexType>
</xs:element>
```

Is the following XML element valid? For **xs:sequence**? For **xs:all**?

```xml
<elementName>
    <elem2>World</elem2>
    <elem1>Hello</elem1>
</elementName>
```

Relational vs. Non-relational data model

Relational (ERD model)

- atomic, flat
- general view
- no data duplication => usage of unique **keys** as reference
- data relations => entity relations, foreign keys
- usage of **ids**

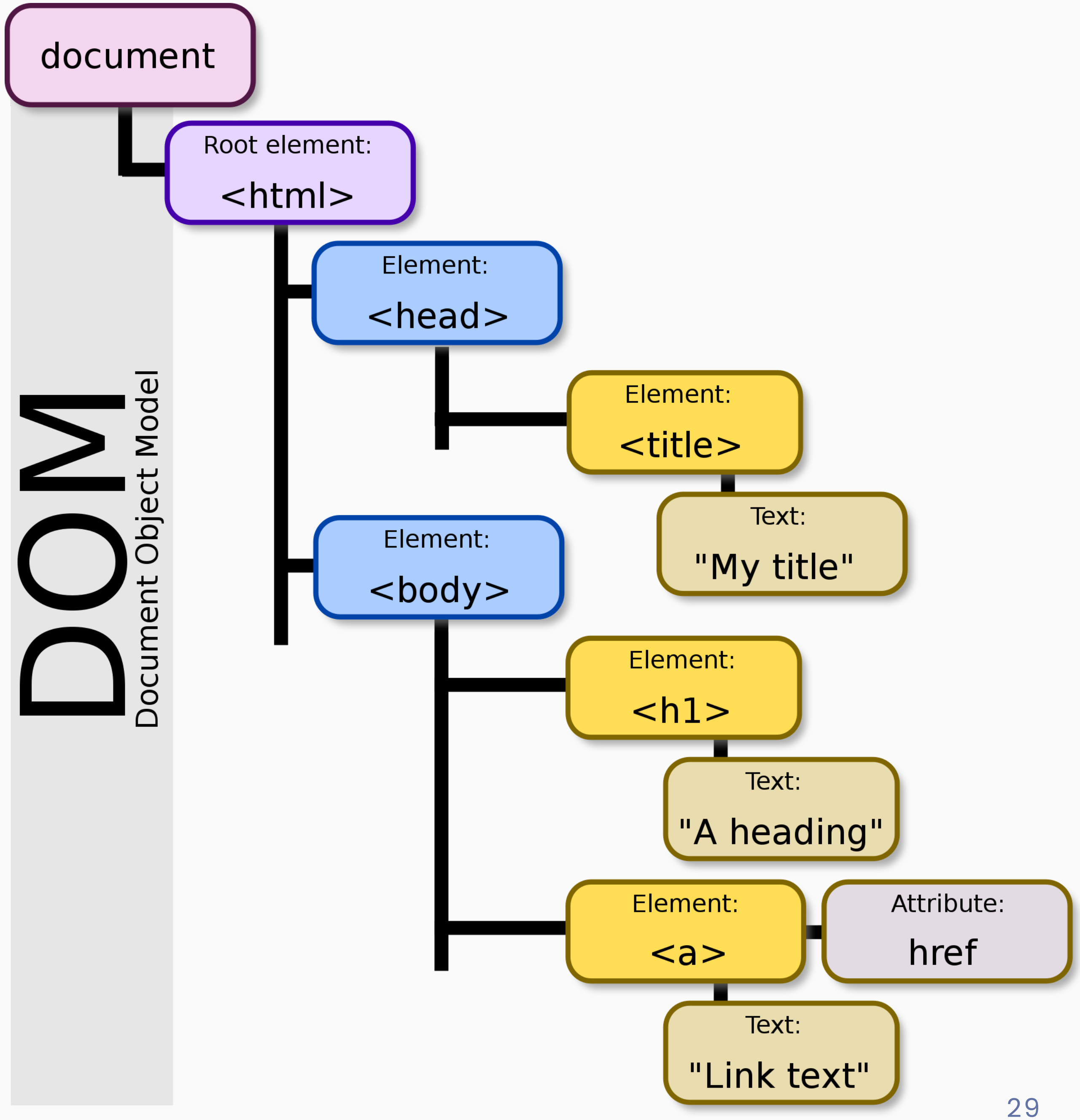
Non-relational (XML document)

- structured, nested
- specific view
- data duplication (sometimes only partially)
- data relations => elements nested one in another
- **ids** not necessary

Bonus topic

DOM

- **D**ocument **O**bject **M**odel
- interface (cross-platform, language-independent)
- represents a document as a tree structure
- each node contains an object
- nodes can have event handlers
- used by browsers to represent an HTML page
- applicable to XML as well



Demo

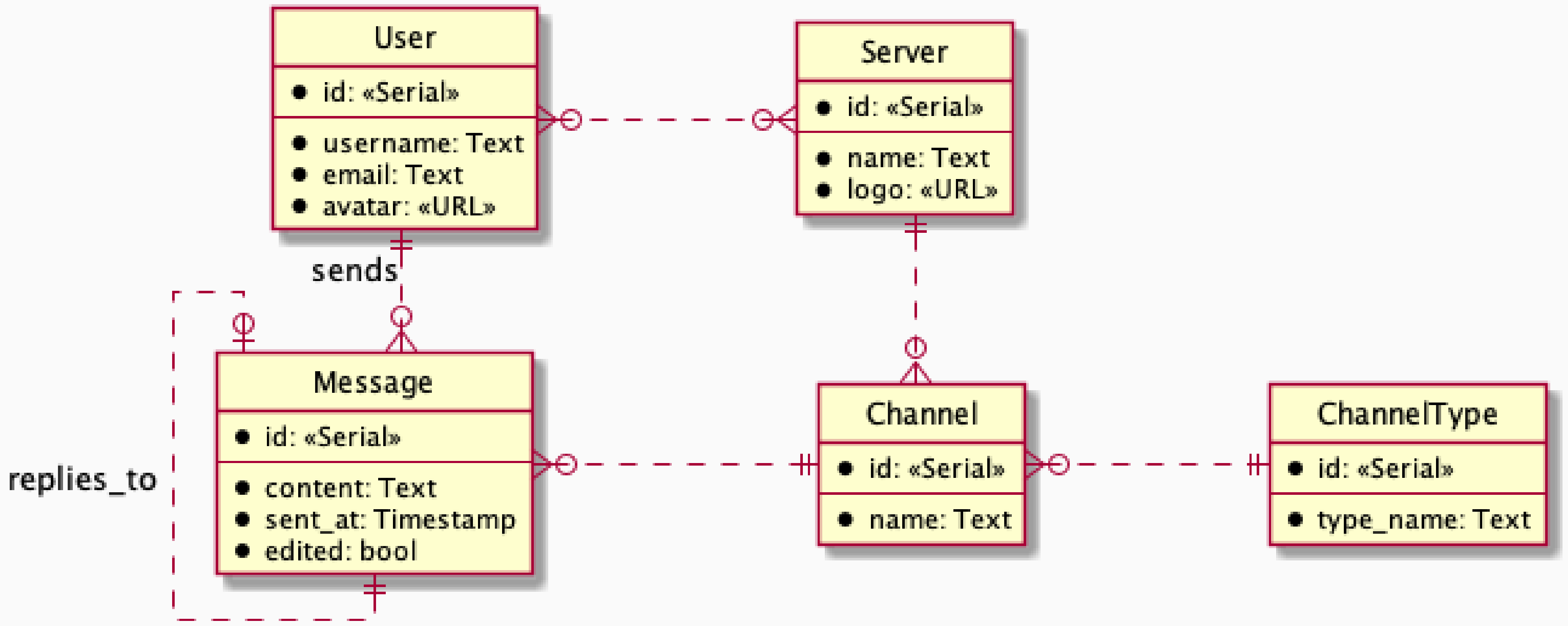
Modelling Discord using XML

1. data modelling
2. schema definition
3. document validation

Note: Demo code is available in the Interactive syllabus for seminars.

Starting point

ERD model from last week (slightly extended)



How to run a validation?

- online, for instance: freeformatter or utilities-online
- VS Code extension, for instance: XML extension from Red Hat (does also formatting and other)

Now, it's your turn :)

The assignment for **Iteration 01** can be found in Gitlab issues.

But before you start, make sure that your forked and cloned repo contains a **.gitlab-ci.yml** file.

If it doesn't, follow these steps:

```
git checkout main    # make sure you're on the main branch by switching to it
git pull upstream main  # download changes (the .gitlab-ci.yml file should be downloaded)
git push origin main    # update your main branch on the Gitlab side (your origin)
```

Now, you can continue as described in *How to download new iteration* on Gitlab Wiki.

If you struggle, don't hesitate to ask for help :)

Note: even though the nature of XML does not require indentation and proper formatting, please format it anyway. It makes the document much more readable.