ZENTRUM FÜR
INFORMATIONSMODELLIERUNG
AUSTRIAN CENTRE FOR
DIGITAL HUMANITIES

KARL-FRANZENS-UNIVERSITÄT GRAZ
UNIVERSITY OF GRAZ

Beyond TEI

Digital Editions with XPath & XSLT for the Web & in LaTeX

Sarah Lang

Harvard, April/May 2022

Overview

1. The workshop
2. Motivation
3. Digital Scholarly Editing
4. Digital Archives
5. Annotating with XML markup
6. Text Encoding Initiative
7. Markup & annotation
8. Markup is all around you!
9. Primer on data structures & metadata formats
10. Intro to HTML (& Bootstrap)
11. Intro to LaTeX (& `reledmac`)
12. Navigating XML using XPath
13. XML transformations using XSLT

The workshop

Goals

1. get to know XPath & XSLT (and learn how to use it)
2. understand the role of XML/TEI, XPath and XSLT in Digital Editing
3. be able to use XSLT to generate HTML and LaTeX output from TEI
4. Two days isn't enough for you to master XSLT!

Schedule

Day 1, morning	XML, TEI and Digital Editing → repetition of the basics, making sure we're all on the same page, understanding why we're even learning XSLT.
Day 1, afternoon	Navigating XML documents using XPath, introduction to HTML (& Bootstrap) and LaTeX (& `reledmac`)
Day 2	Transforming XML documents into HTML & LaTeX output formats using XSLT

Single point of entry for all workshop-related materials: LaTeX Ninja blogpost & Github Repository (‘additional resources’ directory)

Introductions

Please introduce yourselves!

1. Name, pronouns, field/topic of study
2. Why did you come to this workshop?
3. Previous experience with Digital Humanities (DH) or editing?

Contact

@SarahALang_
@latex_ninja

sarahalang.com
latex-ninja.com

sarah.lang@uni-graz.at

Sarah Lang (she/they)

- originally from Germany, now in Graz (Austria)
- Studied Latin, French & History (teacher's education) in Graz & Montpellier (France), then Archaeology Bachelor, Master in Religious Studies & Philosophy
- got a DH certificate & started working at Zentrum für Informationsmodellierung (ZIM) / Centre for Information Modelling in Graz
  - Moral Weeklies/Spectators → *gams.uni-graz.at/mws*
  - Graz Repository of Ancient Fables (GRaF) → *gams.uni-graz.at/graf*
  - *PhD thesis:* Decoding alchemical *Decknamen* digitally. A Polysemantic Annotation and Machine Reasoning Algorithm for the Corpus of Iatrochymist Michael Maier (1568–1622)
- *Now:* teaching in Graz, Passau & Vienna; PostDoc in Graz. *Research interests:* history of science (alchemy), Neo-Latin, text mining and computer vision

Motivation

TEI, now what?

Why are we doing this workshop? The motivation from our abstract:

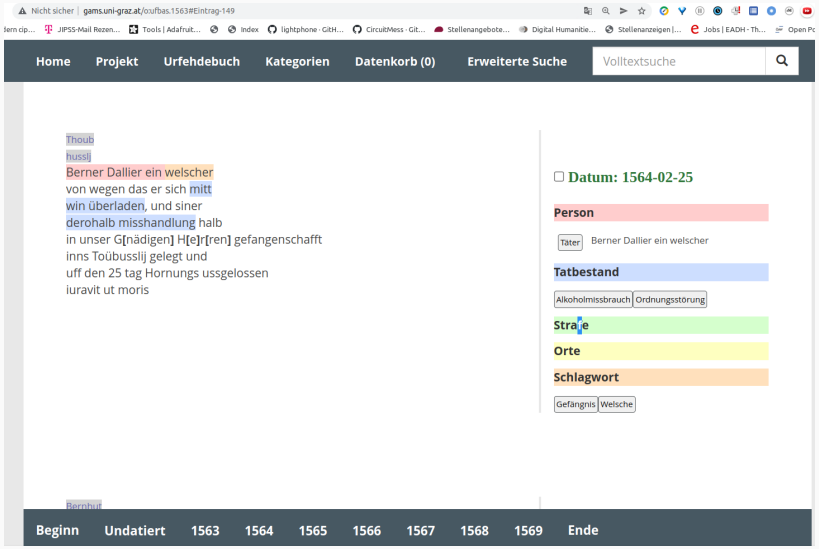
- […] the Text Encoding Initiative (TEI) for XML has become the gold standard for scholarly editions of texts.
- But what happens after an edition is encoded in TEI?
- While it is an **ideal format for archiving digital data**, it is less than ideal for viewing and interacting with the edited text.

Goals for the next session

1. understand the terms from the abstract:
   - ✖ digital edition
   - ✖ single source principle
   - ✖ ~~creating different representations from our data~~ → (why) do we even need to create new presentations from our data? Aren't there tools for that?

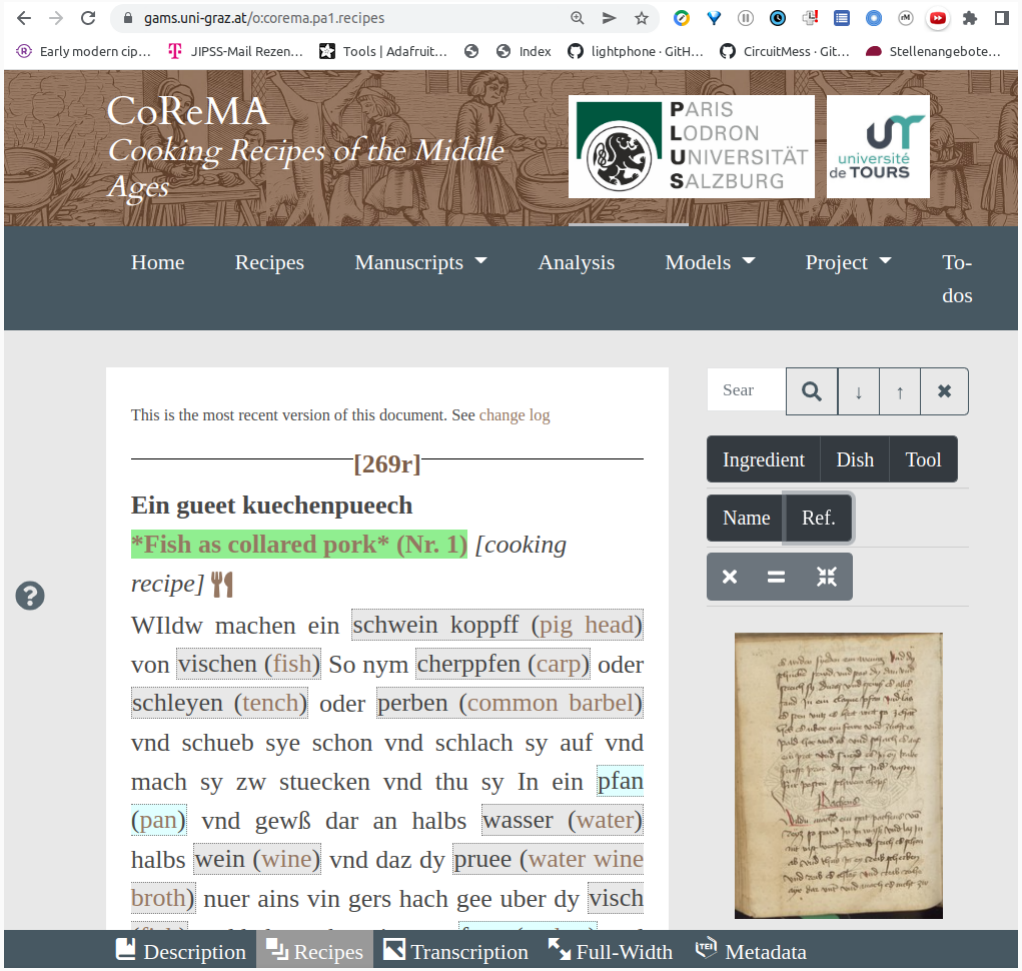
Teaser i

GAMS ufbas



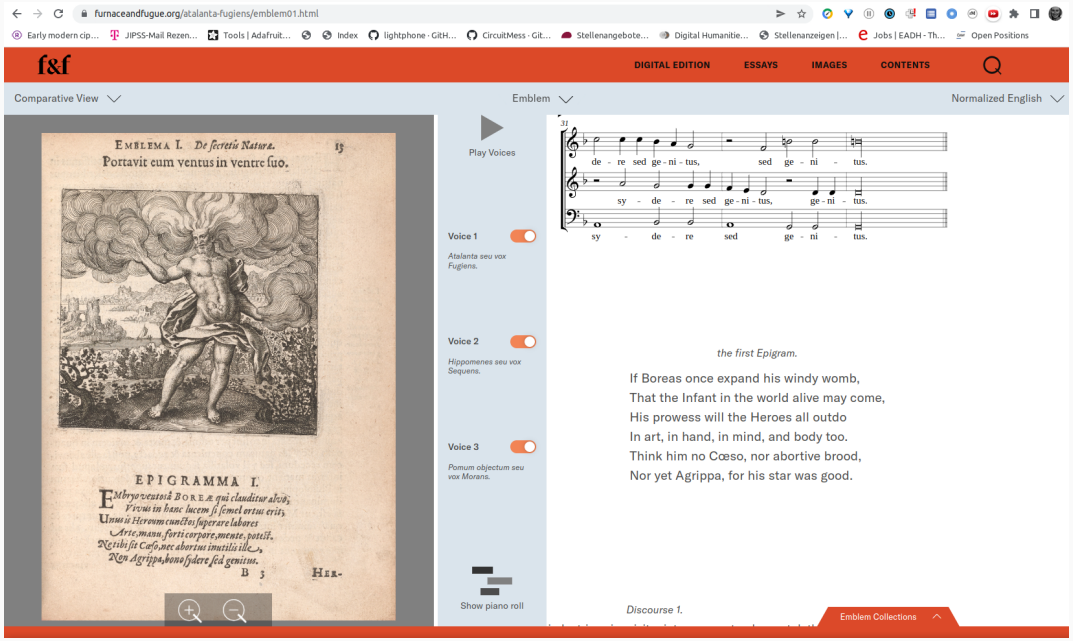
Teaser ii

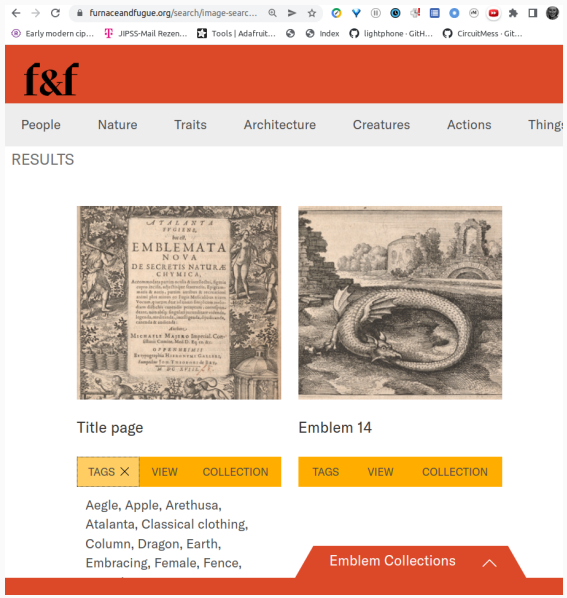
CoReMA → Look at: *http://gams.uni-graz.at/corema*



Teaser iii

Furnace and Fugue (music in MEI, text and images)





From TEI to Edition

What did all those examples have in common?

- all are digital editions
- all are based on data in TEI-XML

That's what we want. – How do we get there?

Remember

- XML is great for capturing & long-term archiving data in its whole complexity
  - → ...but also ends up chaotic with a lot of detail
  - → theoretically human-readable
  - → in practice, it's better to create a new representation of the data to read
- **Desired result:** view/presentation, visualization and interaction with the data as a website (i.e. HTML data)
- → we need to transform the data

TEI Publication Tools

There is a plethora of out-of-the-box solutions to make editions out of TEI data. → TEI List of Resources for Publishing and Delivery Tools, for example:

1. teiPublisher
2. EVT
3. TEICHI
4. Versioning Machine
5. Boilerplate
6. Oxgarage
7. GAMS ...

TEI Publisher: The Instant Publishing Toolbox i

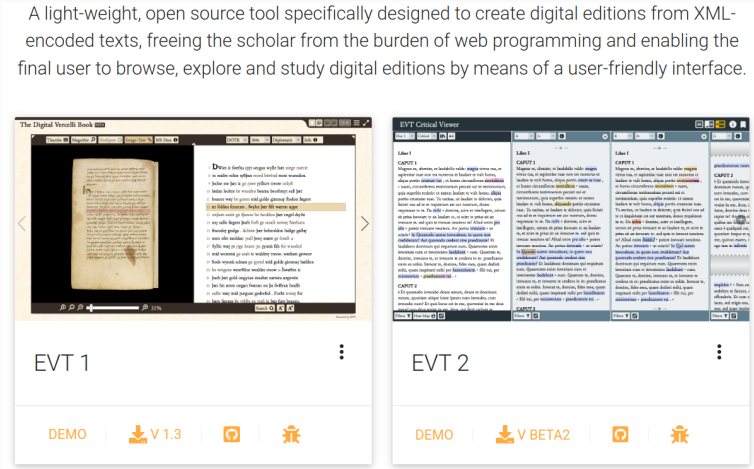
- *https://teipublisher.com/index.html*
- *https://wiki.tei-c.org/index.php/TEI_Publisher*
- *https://wiki.tei-c.org/index.php/TeiPublisher*
- *http://hisoma.huma-num.fr/exist/apps/tei-publisher/index.html*
- "Testimonial": Florian Chiffoleau (2020), Blog Post: *Publication of my digital edition – Working with TEI Publisher.*

TEI Publisher: The Instant Publishing Toolbox ii

Demos: Van Gogh Letters ● Early English Books Online (EEBO)

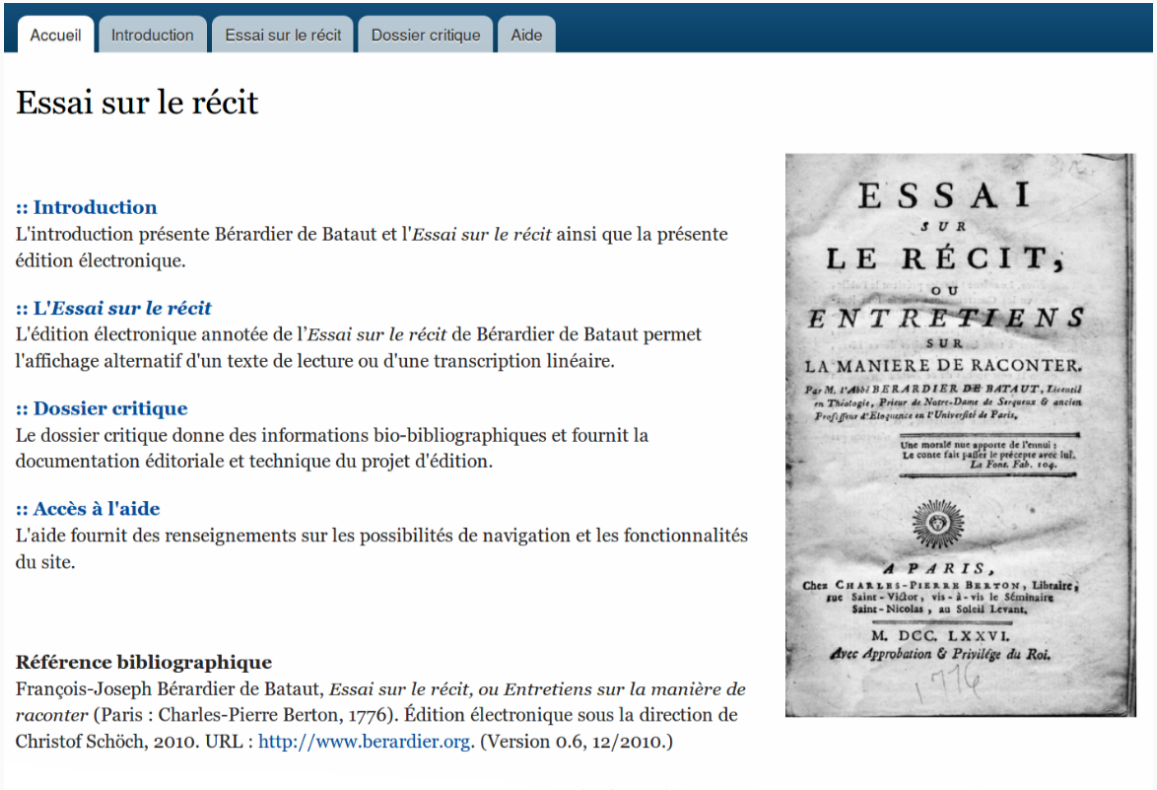
Edition Visualization Technology (EVT)

- *http://evt.labcd.unipi.it/* – *http://evt-project.sourceforge.net/*
- *https://visualizationtechnology.wordpress.com/*
- Roberto Rosselli Del Turco, Giancarlo Buomprisco, Chiara Di Pietro, Julia Kenny, Raffaele Masotti and Jacopo Pugliese: *Edition Visualization Technology: A Simple Tool to Visualize TEI-based Digital Editions*, Journal of the Text Encoding Initiative 8 (2014/15): Selected Papers from the 2013 TEI Conference "TEI Processing: Workflows and Tools"; *https://doi.org/10.4000/jtei.1077*
- Del Turco, Roberto Rosselli. "Designing an Advanced Software Tool for Digital Scholarly Editions: The Inception and Development of EVT (Edition Visualization Technology)." *Textual Cultures*, vol. 12, no. 2, 2019, pp. 91–111. JSTOR, *www.jstor.org/stable/26821538*.



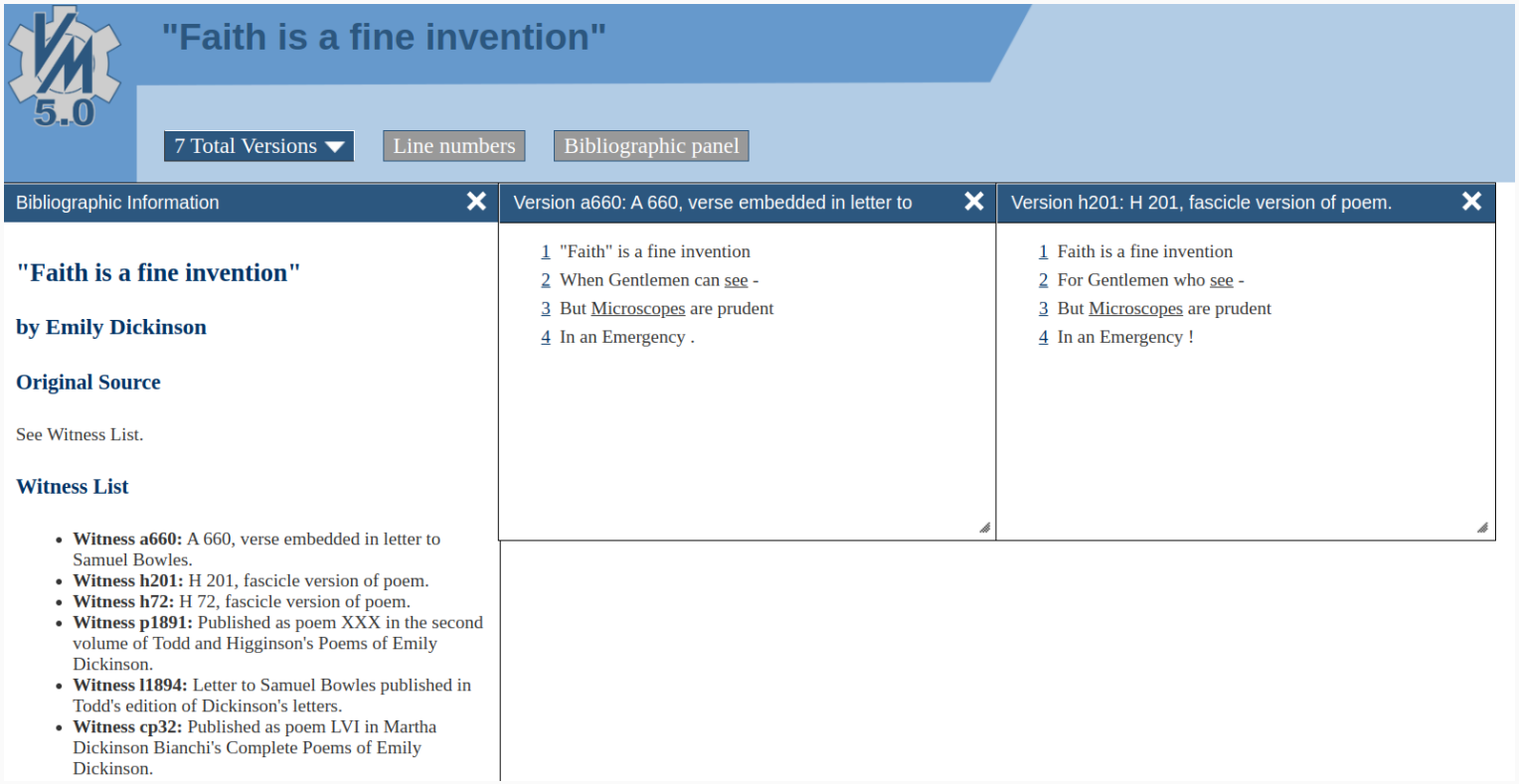
TeiCHI – Bringing TEI Lite to Drupal

- TEIChi: a TEI lite integration into Drupal (*http://teichi.org*)
- Sebastian Pape, Christof Schöch and Lutz Wegner: *TEICHI and the Tools Paradox. Developing a Publishing Framework for Digital Editions*, Journal of the Text Encoding Initiative 2 (2012): Selected Papers from the 2010 TEI Conference; *https://doi.org/10.4000/jtei.432*
- TEICHI wiki



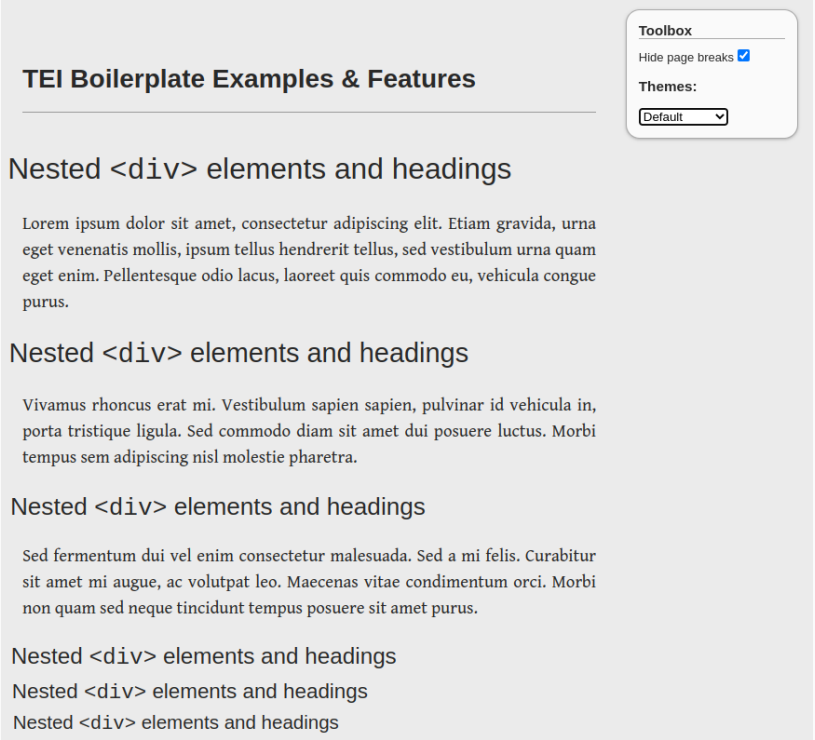
The Versioning Machine

- Versioning Machine (*http://v-machine.org/*)
- The TEI Versioning Machine Versioning Machine (Duke)



TEI Boilerplate

- *http://teiboilerplate.org/*
- Boilerplate (*http://dcl.slis.indiana.edu/teibp/*)
- Demo
- TEI Boilerplate Wiki
- Slides on TEI Boilerplate
- Exercise on TEI Boilerplate
- Poster on TEI Boilerplate



Oxgarage

- REST-based transformation web service for TEI documents (*https://oxgarage.tei-c.org/*)
- allows you to transform a number of (markup-based) formats to TEI or create them from TEI (*.html*, *.tex*, *.docx*, ...)
- Based on TEI base stylesheets (*http://www.tei-c.org/Tools/Stylesheets/*)

✔ very useful tool

✖ not all TEI elements taken into account, not customizable

Oxgarage

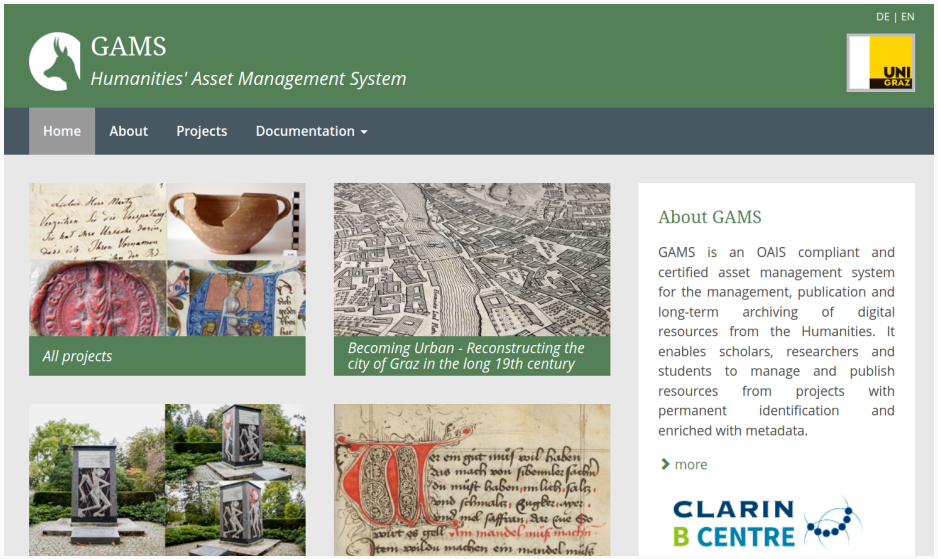
...offers a set of standard stylesheets to convert between TEI-encoded XML documents and other documents, mostly markup-based, such as *.docx*, *.html* or *.tex*.
However, not all elements are covered and you have no control over how exactly they are processed.

Tools make things easier superficially but can come at a cost: potential bugs, less than ideal usability, lack of control and customizability:

→ to customize we need to write our own transformation using XSL stylesheets

GAMS (Humanities Asset Management System) i

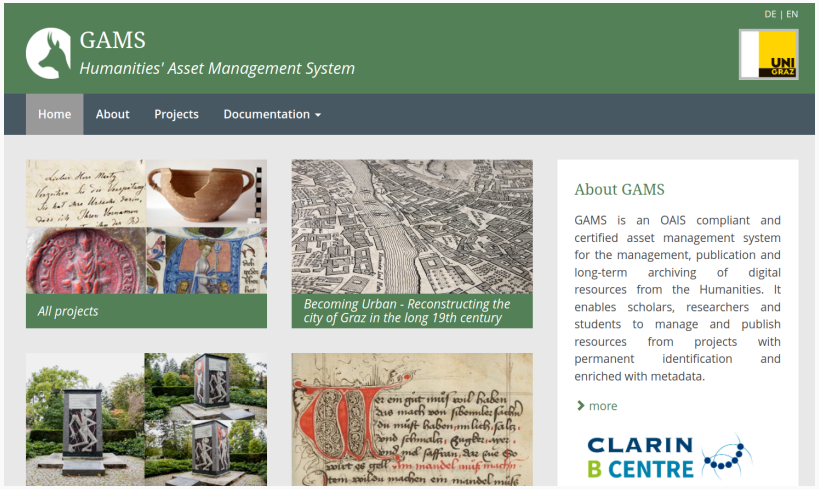
- GAMS (*Geisteswissenschaftliches Asset Management System* = AMS for the Humanities)
- based on **FEDORA** (*Flexible Extensible Digital Object Repository Architecture*) = infrastructure dedicated to the persistent archival and management of resources considered to be worthy of long-term preservation.



https://gams.uni-graz.at

GAMS (Humanities Asset Management System) ii

- user access provided through the **Cirilo client**
- **functionalities:** object creation and management, versioning, normalization & standards, choice of data formats.
- offers a plethora of **pre-defined content models** for data such as TEI, MEI, LIDO, SKOS, ontologies, R code and story lines
- → offers publication pipelines but is highly customizable
- more info: *http://gams.uni-graz.at/doku*



Long-term archiving heritage data i

Long-term preservation

denotes the process of maintaining, curating and keeping data usable over a long period of time (10+ years).

Key functionalities of long-term archiving architectures include

- persistent identification,
- versioning,
- support of different data formats,
- management of associated metadata,
- data export and retrieval,
- security and scalability.

Special emphasis is placed on

- sustainability,
- citability &
- guarantee of long-term access to the contained resources.

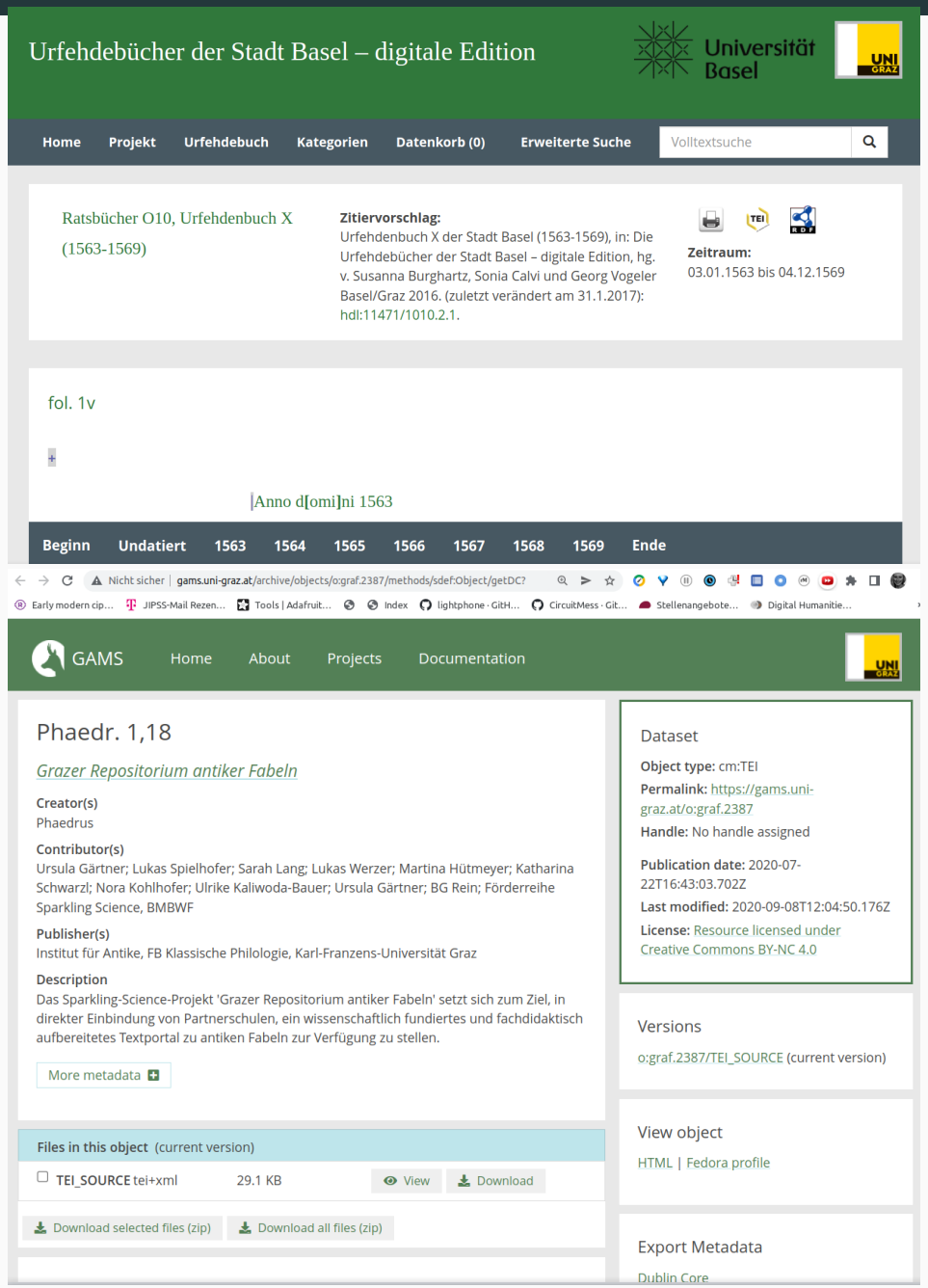
Long-term archiving heritage data ii

XML is great for long-term archiving!

Consequently, data formats and software used for preservation should follow **open source & non-proprietary standards**; data is ideally encoded in an **unicode XML format**:

- plain text files are small
- human- & machine-readable
- recognized standard stable since 1998
- more on XML later...

Examples of data quality in GAMS i



← *http://gams.uni-graz.at/o:ufbas.1563*

- **top image:** info on recommended citation, downloadable source data in XML & RDF
- **below:** 'data view' of the Dublin Core (DC) metadata for an XML object in GAMS (ensuring citability, etc.)

← *http://gams.uni-graz.at/archive/objects/o:graf.2387/methods/sdef:Object/getDC?*

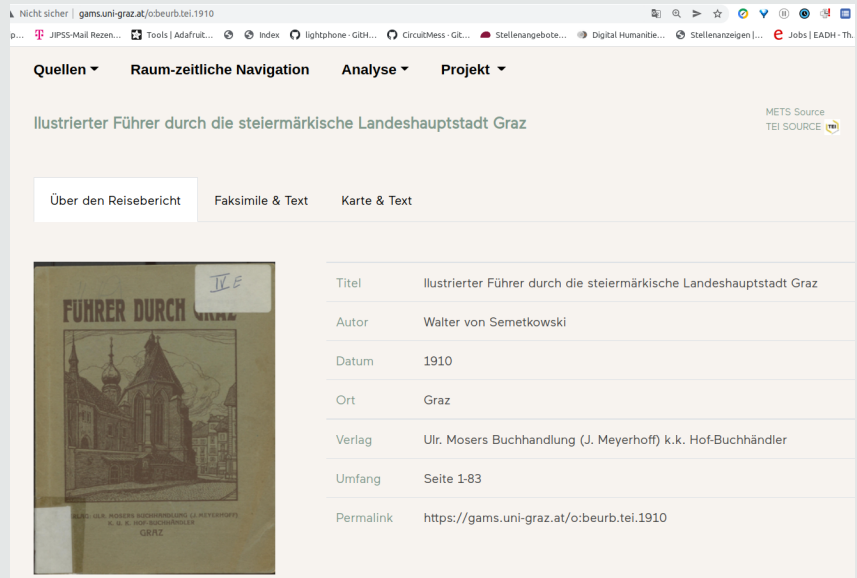
Examples of data quality in GAMS ii

- GAMS isn't an out-of-the-box tool – it's something between a **Content Management System (CMS)**, a **repository** (long-term archiving) and a **publication platform**.
- The XSLT transformations applied to files in GAMS are custom but there are **wippets** (standard javascript functionalities → *widget + snippet*, example here) and **standard templates** which can be reused.
- as a publication platform: presentation and long-term archiving, allowing for persistent access to the data, versioning, etc.

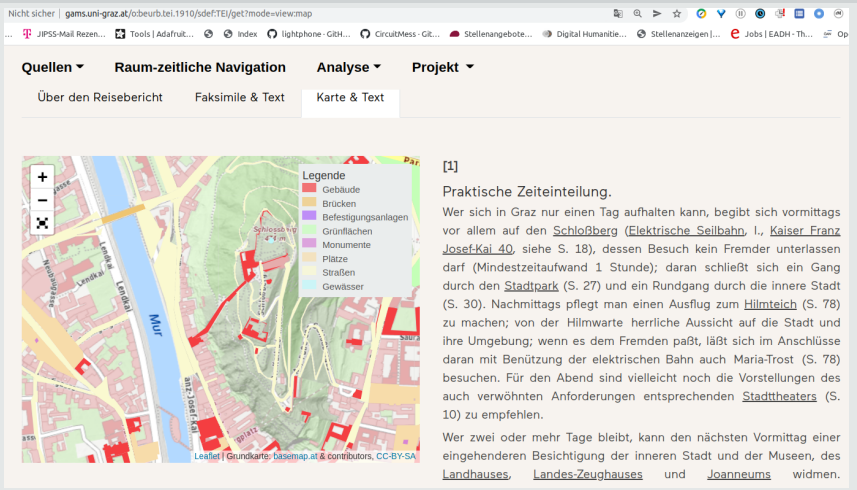
Examples of data quality in GAMS iii

One more example: *https://gams.uni-graz.at/beurb*

Metadata view on XML data



Map view linked to edited text



Both views are created ‘on-the-fly’ from the same XML file (=*single source principle*, more on that later…).

Practice!

Use `https://oxgarage.tei-c.org/` to transform your data to HTML. What do you notice? Do you like it or would you need to customize?

Digital Scholarly Editing

What is a Digital Scholarly Edition? i

Digital editions (as per Sahle 2016):

- discipline-independent & not exclusive to text
- overcome the limitations of print (financial, inter-/ multimediality, etc.)
- follow a digital paradigm like printed editions follow the print paradigm ("shaped by the technical limitations and cultural practices")
- "A digitised edition is not a digital edition." → it's not about the storage medium (just being on the internet doesn't make it a digital edition!)
- "A digital edition cannot be given in print without significant loss of content and functionality."
  - different views, interactivity, searchability
  - facsimiles alongside diplomatic transcription and reading versions
  - all generated from the same source data: **single source principle** → How? Data transformation!
  - ≠ Digital Archive, text corpus or facsimile → has critical engagement

What is a Digital Scholarly Edition? ii

We need to start at the beginning:

What is a scholarly edition?

- originally stems from ecdotics (*ecdotica, ecdotique, Editorik / Editionswissenschaft*) → tradition of Philology
- **goal = textual criticism**: reconstruct the original version of a text transmitted to us via textual witnesses (*Lachmannian paradigm*)

Sahle 2016

> A scholarly edition is the critical representation of historic documents.
> *Edition ist die erschließende Wiedergabe historischer Dokumente.*

What is a Digital Scholarly Edition? iii

Criteria from Sahle 2016:

1. **Representation:** recoding a document and its transformation (in the same or other media)
    - Visually: image reproduction, facsimile
    - Textually: transcription.
    - varying degrees of closeness/abstraction with regard to the original document
2. representation $\neq$ presentation
3. **Critical engagement** (based on scholarly agenda). "Critical engagement without representation is not an edition–but an examination, a catalogue or a description."
    - textual criticism
    - historic criticism
    - bibliographic criticism
    - material criticism
    - visual criticism
    - etc.

What is a Digital Scholarly Edition? iv

4. **Documents:** “every non-abstract object that is the subject of an edition can be called a document.”
5. **Historic:** editions “explain what is not evident to the present-day reader. In short, they bridge a distance in time, a historical difference. Texts that are created today do not need to be critically edited. They can speak for themselves.”

Sahle 2016

> A representation without [critical] treatment or the addition of information is not an edition–but a facsimile, a reproduction or–nowadays–a digital archive or library.
>
> **Critical representation** as a compound notion of editing aims at the reconstruction and reproduction of texts and as such addresses their material and visual dimension as well as their abstract and intentional dimension(Sahle 2016).

What is a Digital Scholarly Edition? v

Questions to ask if you want to know if it's a Digital Scholarly Edition:

1. **Is there a full representation** of the subject in question?
2. **Is it critical?** → processing rules stated and applied, scholarly knowledge included to make the document easier to understand, regarding material, document genesis/creation, context and reception?
3. **Is the edition of academic quality?** → transparent and rigorous edition process, responsibilities stated, enables future research on a reliable basis.
4. **Does the edition follow a digital paradigm?** → makes use of the possibilities of the digital, not printable without major loss of content or functionality

What is a Digital Scholarly Edition? vi

Ideally, a DSE should also implement the FAIR criteria

Findable. in library catalogs, discovery systems or repositories, with a persistent identifier

Accessible. free for any user, no access restrictions? (from open access to usability, language selection, etc.)

Interoperable. data in standardized and widely used format (for example TEI or similar standards), allowing for reuse and data exchange outside of the project.

Reusable. data accessible (individual download, aggregate download, repository, API)? Licenses allowing reuse. Data creation, modelling and processing documented adequately so that others can make sense of it?

→ RIDE (review journal for digital editions and resources)

→ RIDE Criteria for Reviewing Digital Editions and Resources

Edition workflows and schools i

The following slides are adapted from Georg Vogeler's slides on digital editing.

Traditional Workflow of Philological Editing

- **Heuristics:** Find your textual witnesses
- **Transcription:** transfer the text into your prefered alphabet (from the original/from a photo)
- **Collation:** Compare the textual witnesses
- **Recensio:** Evaluate the variants and create a stemma (commenting)
- Write your introduction
- Typesetting
- Create an index (refering to pages)
- Print and distribution

In the digital realm:

- edition
- beta version
- just TEI data publication
- hybrid (web and print) publication
- minimal edition (adaptation from minimal computing)

A digital edition is social, iterative, a process...

Edition workflows and schools ii

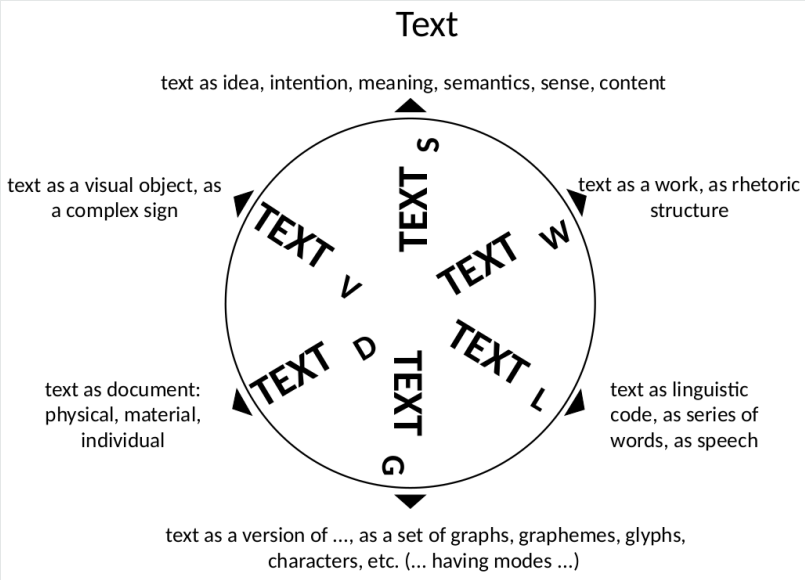
„Edition“

- a particular version of a book
- a particular version of a product
- all the copies of a book that are printed or published at one time

„Historical-critical“ Edition

- Documentation of the history of the text transmission in the “apparatus of variants”
- Critical evaluation of the textual transmission

Patrick Sahle’s text wheel



Karl Lachmann (1793–1851)

Editorial intervention: Philological knowledge (linguistics, style) allows to emend fragmentary textual transmission.

Edition workflows and schools iii

Editorial Schools

- Lachmann ('Historisch-Kritische Ausgabe', stemmatology)
- Reading text vs. critical edition = reader-oriented pragmatism / modernised edition / 'reading text'
- Last authorized edition
- First edition / *editio princeps*
- Main manuscript (Bédier 1937, 'Leithandschrift')
- 'diplomatic edition / transcription'
- Typographic/photographic facsimile
- 'documentary editing' (Tanselle)
- variorum edition
- genetic edition / *critique génétique* (Gabler)
- Copy-Text-Theory (Greg 1950/3, Bowers 1976, 1978)
- New Philology (Stackmann 1964, 1993, 1994, 1999; Ruh 1978, Cerquiglini 1989)

Digital Archives

Digital long-term archiving

- ensuring the authentic and sustainable availability of digital ressources on the level of the bitstream and on a semantic level
- integral principle to every form of sustainable data storage
- begins with the data production in a sustainable data format (and ideally, following a recognized data standard)
- requires standardization of data formats and archiving workflows
- serves both the dissemination as well as the preservation of digital content
- not just about technical solutions but also institutional stability and policies

A digital archive

- is more than a mere collection of scanned book pages or digitized images, etc.
- if offers metadata, norm data and controlled vocabularies

Digital Archives

Digital Archive

- organized collection of digital objects (text, images, audio, video and multimedia streams)
- digital objects are described by standards both in terms of contents (e.g. TEI) and bibliographically (e.g. Dublin Core)
- published sustainably using interfaces, services and APIs (e.g. OAI-PMH)
- digital objects have unique, persistent and citable identifiers (e.g. DOI, URN, PURL, PID)
- authenticity of objects is checked by means of digital signatures or checksums ('Is the number of bytes in the object still the same it used to be?')

Trustworthy Digital Archives

as defined by the Research Libraries Group (RLG)

- secure organizational structure and legal status
- financial sustainability
- technological and procedural aptness
- ensuring data and system security
- documentation and transparency
- conformity with the OAIS standard

→ retro-digitizing objects but also standards for new (born digital) resources.

Open Archival Information System Reference Mode

OAIS

Generic model for the organization of a digital archive
→ developed 1995–2002 by the Consultative Committee for Space Data Systems (CCSDS)

Tasks for Digital Archives according to OAIS

1. data ingest
2. archival storage
3. data management
4. system administration
5. preservation planning
6. access

What's the difference between a Digital Archive and a Digital Scholarly Edition? (Sahle 2007)

Digital Archives vs. Digital Editions i

Digital documents

Material objects are the target of digitization but digitization doesn't reproduce them – it represents them in a digital format. A digital document is a view on the original material object.

Archive

Traditionally an archive is an ordered collection of documents with the goal of documenting them, preserving them in the long-term and making them accessible. → this function isn't only carried out by actual archives but also by museums, libraries and other cultural heritage institutions. Traditional archive material doesn't need to be represented because its physical objects can be accessed directly.

(we have already def'd editions)

Editions are often based in archival material. The edition isn't a storage device, it is a publication of a historical source and the editorial work done on it. While the edition contains lots of editorial work and enrichment, archival documents are usually original and largely unprocessed. A non-digital archive isn't a form of publication in and of itself – a digital archive *is* a form of publication, too, like a digital edition → blurs the lines a bit.
We could say that **the difference lies in the depth of data enrichment and editorial work.** → once a presentation form is provided, data from a digital archive becomes a digital edition.

Digital Archive

A main goal of a digital archive is to preserve and publish a specific choice of documents as a collection (ideally, representative and well-balanced). The digital objects aren't necessarily direct representations but may have undergone editorial intervention (e.g. normalized orthography). Its documents should be uniform in how they are encoded and processed. In the case of the single source principle, the edition is generated dynamically ('on the fly') from the archived source data. (If any edition is provided).

Annotating with XML markup

TEI, now what?

Why are we doing this workshop? The motivation from our abstract:

- [...] the Text Encoding Initiative (TEI) for XML has become the gold standard for scholarly editions of texts.
- ...

Goals for the next session

1. wait, what was...
   - ✖ XML?
   - ✖ TEI?
   - ✖ How do I use the TEI for digital editing?

XML: eXtensible Markup Language

- W3Schools Tutorial
- paradigm of the separation of form and content
- XML is a metalanguage

.XML

- RSS, SOAP, XAML
- MathML, GraphML
- XHTML
- RDF
- KML
- Scalable Vector Graphics (SVG)

> **Extensible Markup Language (XML)** is a **markup language** and file format for storing, transmitting, and reconstructing arbitrary data. It defines a **set of rules for encoding documents** in a format that is **both human-readable and machine-readable.** (Wikipedia)

XML rules

XML can be checked for **validity** (validation if it complies with a standard) and **well-formedness** (following the rules of XML) → will only be parsed if well-formed. Thus: Heed thy error messages!

There are rules on how elements can be named (you can look them up if relevant or will get informed by an error message).

`<key>value</key>` . XML as a key value notation

Rules

- Hierarchical nesting below the root
- exactly one root element, i.e. one out-most russion doll
- start and end tag
- tag names are case-sensitive (!)
- empty elements allowed (& can be shortened)

Minimal example

```
<?xml version="1.0" ?>
<root>
  <element attribute="value">
    content
  </element>
  <!-- comment -->
</root>
```

XML rules i

Prolog

`<xml version="1.0" encoding="utf-8">` ..... XML deklaration

`<?xsl-stylesheet type="text/xsl"`
`href="mein.xsl"?>` ..... processing instructions (optional)

you can include document models (optional)
DTD, XML Schema, RelaxNG, Schematron

entities ‘protected’ characters because they have a meta meaning in XML
like:

`<` ..... <

`>` ..... >

`&` ..... &

XML family and vocabularies

XML structured description of data
XPath navigating xml documents
XML Schema strict data model
XSL Extensible Stylesheet Language
XSLT XSL-Transformations, i.e. transforming XML documents
XSL-FO formatted output (e.g. print)
XQuery query langauge for XML databases
and more

- **(X)HTML** Hypertext Markup Language
- **EAD** Encoded Archival Description
- **TEI** Text Encoding Initiative
- **CEI** Charters Encoding Initiative
- **MEI** Music Encoding Initiative
- **LIDO** Lightweight Information Describing Objects (describing museum or collection objects)
- **SVG** Scalable Vector Graphics
- **KML** Keyhole Markup Language (Geographie)
- **MathML**
- **CML** Chemical Markup Language, ...

Text Encoding Initiative

TEI Primer

Text Encoding Initiative

.XML

XML-Standard, i.e. convention on how to use XML so that resulting data will be interoperable between different projects. (founded in 1987, consortium since 2000)

> The Text Encoding Initiative (TEI) is a text-centric community of practice in the academic field of digital humanities, operating continuously since the 1980s. The community currently runs a mailing list, meetings and conference series, and maintains the TEI technical standard, a journal, a wiki, a GitHub repository and a toolchain. (Wikipedia)

TEI minimal example

```
<TEI> <!-- root element -->
    <teiHeader>
      <!-- author, title, dating,
           sources, edition rules, etc.
    </teiHeader>
    <text> ... </text>
</TEI>
```

Resources

- Learn TEI
- Teach Yourself
- P5 = 5. Proposal
- MEI for music
- CEI for charters
- http://www.tei-c.org/

TEI Header

fileDesc = bibliographical description of the contents of the document

encodingDesc = connection of electronic document to source (i.e. transcription rules, etc.)

```
<TEI> <!-- root element -->
    <teiHeader>
        <fileDesc> ... </fileDesc> <!-- obligatory -->
        <encodingDesc> <!-- optional -->
        <profileDesc> <!-- optional -->
        <revisionDesc> <!-- optional -->
    </teiHeader>
    <text> ... </text>
</TEI>
```

profileDesc = decribes all non-bibliogaphical aspects of the text (i.e. creation, languages)

revisionDesc = tracks changes in the digital document

Using TEI i

Gentle Intro to XML

TEI Core

- **div** (division)
- **p** (paragraph)
- **head** (heading)
- **lb** (linebreak)
- **pb** (page break / beginning)
- **hi** (highlight)
- **l** (line)
- **lg** (line group)
- **list**
- **item**
- **listBibl**
- **bibl** (bibliographical information)

Attributes

- **@n** (label)
- **@type** (typing)
- **xml:id** (unique identifier)
- **xml:lang** (language)
- **@rend** (rendering)
- @ana (interpretation)

Using TEI ii

```
<foreign xml:lang="en">word</foreign>
<term type="homonym"/>
<date when="2009-04-27"/>
<time when="12:00:00"/>
<name type="person"/>
<persName n="Caesar" xml:id="#44BC">Caesaris</persName>
<!-- or -->
<persName key="ID.01.208"/>
<person/>
<emph/> <hi rend="italic">italic text</hi>
<seg/> <abbr type="acronym"/>
<placeName xml:id="#Whitby">Abbey</placeName>
```

Using TEI iii

Name spaces identified via URI

<präfix:name> e.g. *<tei:p>* ('I mean the *<p>* according to the TEI standard.')

declaration <element xmlns="URI"> ...
<prefix:element xmlns:prefix="URI"> ...
e.g.

```
<tei:p xmlns:tei="http://www.tei-c.org/ns/1.0">...
```

TEI is organized in modules

Acts of speech (reference) if speaker name is mentioned, otherwise TEIs ‘said’:

```
<sp who="#person">
    <speaker>1.</speaker> <p>Bla, bla, bla.</p>
</sp>

<said who="#Adolphe">- Alors, Albert, quoi de neuf?</said>
```

Letters in TEI (reference)

```
<div type="letter" n="14">
    <head>Letter XIV: Miss Clarissa Harlowe to Miss Howe</head>
        <opener>
            <dateline>Thursday evening, March 2.</dateline>
            <salute>Hallo,</salute>
        </opener>
    <p>On Hannah's depositing my long letter ...</p>
    <closer>
        <salute>Yours more than my own,</salute>
        <signed>Clarissa Harlowe</signed>
    </closer>
</div>
```

Names, Dates, Places

Named Entities & Indirekte reference

TEI 13: Names, Dates, People, Places

- **persName** for personal names, **<rs>** for *referring string* when mentioned indirectly (‘he’, ‘the woman’, etc.) →**@key** or **@ref** to specify who it is (reference).
- **forename**
- **surname**
- **roleName** (z.B. ‘king’)
- **genName** (‘the Younger’)
- **addName**
- **nameLink** (‘von’).

```
<name role="writer" type="person"
ref="http://d-nb.info/gnd/118540238">
Goethe</name>
<person>
  <addName type="Former">Murray</addName>
  <forename>Wilhelmina</forename>
  <addName type="nickname">Mina</addName>
</person>
```

Metadata in the TEI Header i

```
<teiHeader>
 <fileDesc>
  <titleStmt>
   <title>
<!-- title of the resource -->
   </title>
  </titleStmt>
  <publicationStmt>
   <p>
<!-- Information about distribution of the resource -->
   </p>
  </publicationStmt>
  <sourceDesc>
   <p>
<!-- Information about source from which the resource derives -->
   </p>
  </sourceDesc>
 </fileDesc>
</teiHeader>
```

Metadata in the TEI Header ii

The title and author in the *<titleStmt>* isn't the bibliographic data from the source! It describes the digital document and its authors or editors.

If you want to desribe your source documents, you need elements like *<sourceDesc>* or *<msDesc>*:

```
<sourceDesc>
 <bibl>
  <title level="a">The Interesting story of the Children in the Wood</title>. I
 <author>Victor E Neuberg</author>, <title>The Penny Histories</title>.
 <publisher>OUP</publisher>
  <date>1968</date>. </bibl>
</sourceDesc>
```

```
<sourceDesc>
 <p>Born digital: no previous source exists.</p>
</sourceDesc>
```

Metadata in the TEI Header iii

```
<teiHeader>
 <fileDesc>
  <titleStmt>
   <title>Thomas Paine: Common sense, a
       machine-readable transcript</title>
   <respStmt>
    <resp>compiled by</resp>
    <name>Jon K Adams</name>
   </respStmt>
  </titleStmt>
  <publicationStmt>
   <distributor>Oxford Text Archive</distributor>
  </publicationStmt>
  <sourceDesc>
   <bibl>The complete writings of Thomas Paine, collected and edited
       by Phillip S. Foner (New York, Citadel Press, 1945)</bibl>
  </sourceDesc>
 </fileDesc>
</teiHeader>
```

<msDesc>

```
<msDesc>
 <msIdentifier>
  <settlement>Oxford</settlement>
  <repository>Bodleian Library</repository>
  <idno type="Bod">MS Poet. Rawl. D. 169.</idno>
 </msIdentifier>
 <msContents>
  <msItem>
   <author>Geoffrey Chaucer</author>
   <title>The Canterbury Tales</title>
  </msItem>
 </msContents>
 <physDesc>
  <objectDesc>
   <p>A parchment codex of 136 folios, measuring approx
       28 by 19 inches, and containing 24 quires.</p>
   <p>The pages are margined and ruled throughout.</p>
   <p>Four hands have been identified in the manuscript: the first 44
       folios being written in two cursive anglicana scripts, while the
       remainder is for the most part in a mixed secretary hand.</p>
  </objectDesc>
 </physDesc>
</msDesc>
```

<titlePage>

To describe a title page (e.g. early modern print copperplates, etc.), use *<titlePage>*:

```
<titlePage>
 <docTitle>
  <titlePart type="main">THOMAS OF Reading.</titlePart>
  <titlePart type="alt">OR, The sixe worthy yeomen of the West.</titlePart>
 </docTitle>
 <docEdition>Now the fourth time corrected and enlarged</docEdition>
 <byline>By T.D.</byline>
 <figure>
  <head>TP</head>
  <p>Thou shalt labor till thou returne to duste</p>
  <figDesc>Printers Ornament used by TP</figDesc>
 </figure>
 <docImprint>Printed at <name type="place">London</name> for <name>T.P.</name>
  <date>1612.</date>
 </docImprint>
</titlePage>
```

<front>

You might also need *<front>* (front matter): contains any prefatory matter (headers, abstracts, title page, prefaces, dedications, etc.) found at the start of a document, before the main body.

```
<front>
 <epigraph>
  <quote>Nam Sibyllam quidem Cumis ego ipse oculis meis vidi in ampulla
     pendere, et cum illi pueri dicerent: <q xml:lang="grc">Σίβυλλα τί
       θέλεις</q>; respondebat illa: <q xml:lang="grc">ἀποθανεῖν θέλω.</q>
  </quote>
 </epigraph>
 <div type="dedication">
  <p>For Ezra Pound <q xml:lang="it">il miglior fabbro.</q>
  </p>
 </div>
</front>
```

Making the TEI your own

How to find information on TEI elements

...and teach yourself how to use new elements:

- General TEI guidelines (XML Primer, Learn the TEI page, etc.)
- web-search TEI + (element you want to know about), i.e. “tei teiHeader” and you will get:
  1. definition page
  2. list of all examples for that element → directly over websearch or click ‘show all’ in the examples on the ‘definitons page’
  3. sometimes even an module overview

all	All modules
5	Characters, Glyphs, and Writing Modes,
10	Manuscript Description,
11	Representation of Primary Sources,
12	Critical Apparatus,
13	Names, Dates, People, and Places.

Also: The TEI guidelines are documentation and reference,

TEI practice!

Fill out the `teiHeader` or `msDesc`.

Use websearch (‘tei msDesc’) to learn how to use new elements (overview plus examples view).

Markup & annotation

Annotating in the Humanities

→ Typical practice for the Humanities. Can be done on paper.
You can go about it in many ways:

- collect specific metadata to enrich your primary data with (research-driven modelling)
- provide metadata as general as possible to facilitate reuse (curation-driven modelling)
- we can provide administrative or technical metadata but also encode semantic information (implicit structures) for the machine.
- Often: describe logical structure of documents for computers ('This is a heading. This is a paragraph.')

Example:

```
<name type="person">J. W. v. Goethe</name>
```

In the digital realm

Adding additional data to source document in a machine-readable format.

Formal models

If you model is machine-processable, it's a formal model – i.e. markup creates a formal model of your data.

Many names for the same thing

markup, encoding, annotating, ...

Different levels of annotation

Base annotation Encode formal criteria of the text structure (i.e. headings, paragraphs), adding some metadata. Mostly *presentational*.

Data enrichment Going from formal aspects to semantics: Annotating personal names, places (*Named Entities*), linking those to norm data, thus creating *Linked (Open) Data* (LOD).

Examples for norm data

- GND (*Gemeinsame Normdatei*, Integrated Authority File)
- GeoNames
- ...

Markup languages i

Annotation is also called mark-up.

> **Markup** refers to data included in an electronic document which is distinct from the document's content in that it is typically not included in representations of the document for end users, for example on paper or a computer screen, or in an audio stream. **Markup is often used to control the display of the document or to enrich its content to facilitate automated processing.**
>
> Older markup languages [...] typically **focus on typography and presentation**, [...] most modern markup languages, for example XML, **identify document components** (for example headings, paragraphs, and tables), with the expectation that technology such as stylesheets will be used to apply formatting or other processing.
>
> Some markup languages, such as the widely used HTML, have **pre-defined presentation semantics**, meaning that their specification prescribes some aspects of how to present the structured data on particular media. (Wikipedia)

Markup languages ii

machine readable

SMGL HTML XML ...

presentational vs. descriptive / semantic:

- e.g. ‘font size 14pt’ vs. ‘heading’ (=type). (explicit vs. implicit)
- text formatting vs. meaning of the text
- procedural, representative, descriptive / conceptual (semantic)
- WYSIWYG text processing vs. WYSIWYM
- Advantages of using macros in WS Word: change the settings once for the whole document
- this is achieved when we separate content from its presentation
- there are different ‘views’ on markup documents
- browser ‘renders’ HTML: I can see it in two different ways – rendered or as the HTML code

Markup languages iii

- There are tools to switch documents between different types of markup! (not all formats work equally well): **Pandoc** or **OxGarage** (for TEI mostly).
- Binary document formats, such as *.doc*, *.pdf*, *.dvi* (TEX output format) $\neq$ markup: You can tell by the fact that you cannot look at their 'code view'.
- A `.docx` file is a `.zip` archive (try unzipping it!) which contains XML files (but it's complicated).
- **Goal of markup:** Make the implicit (you know it's a heading) explicit for the computer (doesn't know otherwise).

Markup is all around you!

SGML

Standard Generalized Markup Language

.SGML

Introduces the principles of

the separation of form and content

Is a metalanguage (like XML)

HTML & XML are both derived from the older SGML (thus the similarity) – but HTML has a fixed tag set whereas XML is by definition *extensible*.

RTF

Rich Text Format Microsoft 1987 ● exchange format between text editors (different operating systems, such as Mac and Windows, didn't create interchangeable output formats).

Unlike plain text, RTF contains markup for formatting text which can then be 'retranslated' into the native editors.

.RTF

```
{\rtf1
Hello!
\line
{\i This} is \b{\i a
\i0 formattted \b0text}.
\par
\b THE \b0END.
}
```

JSON

.JSON JavaScript Object Notation pronounced like 'Jason' ● compact data format ● human readable ● set up in key-value pairs ● nested (like XML)

JSON vs. XML differences?

XML = describes structure, JSON = non-declarative syntax convention ● JSON: defines instances of structured data ● very flexible ● 'lightweight': little overhead, easier to read for data in key-value format ● valid javascript: can be instantiated into a javascript object via the *eval()* function.

Conclusion

JSON has advantages if you just need simple key-value pairs (**'simplicity'**). XML has more and allows for more **complexity**.

XML = mark up language

JSON = data exchange format

```
{
  "publisher": "Xema",
  "number": "1234-5678-9012-3456",
  "owner":
  {
    "Name": "Mustermann",
    "Vorname": "Max",
    "male": true,
    "hobbies": ["surfing", "chess"],
    "age": 42,
    "kids": [],
    "spouse": null
  }
}
```

PostScript

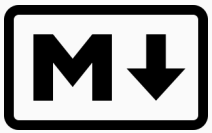
Page description language ● 1980s (Adobe Systems) ● vector graphic format for printers ● but also: Turing-complete, stack-oriented programming language ● used to be the standard in the printing industry ● today PDF (*Portable Document Format*) (also by Adobe, developed from PS) has become the standard ● could be generated via postscript printer drivers from all sorts of documents ● processed via 'Ghostscript' in UNIX ● decribes documents as scalable vector graphics which allows for loss-less zooming / scaling .ps // presentational

Example

This example program writes 'Hello World!' to position 50,50. By default, the PS coorindate system starts from the bottom left corner.

```
%!
/Courier findfont     % font type
20 scalefont          % font size 20
setfont               % set it
50 50 moveto          % (50, 50)
                      % = writing pos
(Hello World!) show   % print text
showpage              % show page
```

Markdown



→ Markdown in 60s

simple text formatting

.md // presentational

`*Italic*`	*Italic*
`**Bold**`	**Bold**
`## Heading 1`	Heading 1
`#### Heading 2`	etc.
`[Link]{http://a.com}`	'hidden' link
`![Image][http://url/a.png`	image
`> Blockquote`	quote
`- List`	list (can be nested, 2 spaces)
`* List`	alternative list
`1. enumerate`	numbered list
`---`	separator line
`` `Inline code` ``	Code ('backticks')
`` ```code block``` ````	code block

Practice!

Try Markdown in 60s or 10min exercise

Primer on data structures & metadata formats

TEI, now what?

Why are we doing this workshop? The motivation from our abstract:

- [...] the Text Encoding Initiative (TEI) for XML has become the gold standard for scholarly editions of texts.
- But what happens after an edition is encoded in TEI?

Goals for the next session

1. There are many different types of data. They get stored in different structures.
2. Why XML and not something else?
3. What else is there?
4. **Also:** some metadata literacy!

Digital data representation

→ i.e. machine processable

Digital representations

- Images
  - raster graphics (*.png*, *.jpeg*)
  - vector graphics (*.svg*)
- Text
  - plain text (*.txt*)
  - formatted text (*.docx*, *.rtf*, *.xml*, *.tex*)
- Lists & tables (*.csv*, *.xlsx*)
- Sound (*.wav*, *.midi*)
- **Objects:** (Simulated) 3D view, abstracted representation by description and images

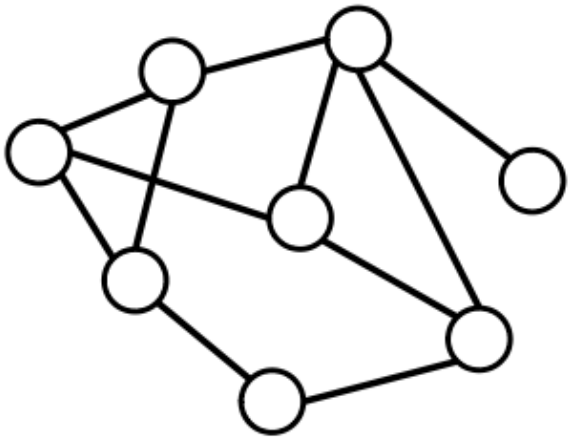
Further types

- markup languages (*.xml*, *.html*, etc.)
- data objects (*.json*, etc.)
- graphs / graph databases (*.rdf*, etc.)
- relational databases → SQL

Data structures: Graphs

Applications

- **Resource Description Framework (RDF):**
  - e.g. Blazegraph (Graph-DB)
  - Query: SPARQL
- Labeled Property Graphs
  - e.g. Neo4j (Graph-DB)
  - Query: Cypher



RDF/Turtle-Notation (`.ttl`)

```
@prefix ex: <http://example.com/#> .

ex:Graz a ex:city;
ex:name "Graz" ;
ex:inhabitants 288806 ;
ex:location [ ex:lat 47.4; ex:long 5.26 ] .

ex:Wien a ex:city;
ex:name "Wien" ;
ex:inhabitants 1897491 ;
ex:location [ ex:lat 47.12; ex:long 16.22 ] .
```

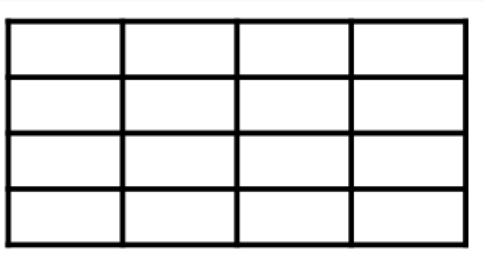
SPARQL query

```
@prefix ex: <http://example.com/#> .

SELECT ?name, ?population
WHERE {
    ?city ex:inhabitants ?population .
}
```

Data structures: tabular data / relational databases

Applications

- e.g. SQLite, MySQL (relational dbs)
- Query: SQL



SQL query

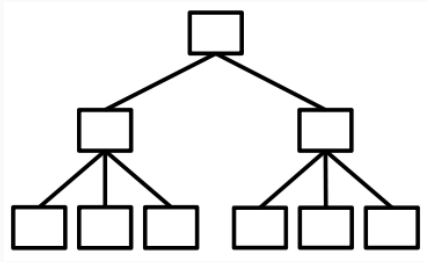
```sql
CREATE TABLE "places" (
 "name" TEXT,
 "population" INTEGER,
 "longitude" REAL,
 "latitude" REAL,
 PRIMARY KEY("name")
);

INSERT INTO places
VALUES
('Graz', 288806, 47.066667, 15.433333),
('Wien', 1897491, 48.208174, 16.373819);
```

(name, population, longitude, latitude)

Data structures: tree hierarchy 1 / JSON

Applications: **JavaScript Object Notation (JSON)**

- e.g. MongoDB
- no standardized query language, just JavaScript (*.js*)



JSON & JS (w3s)

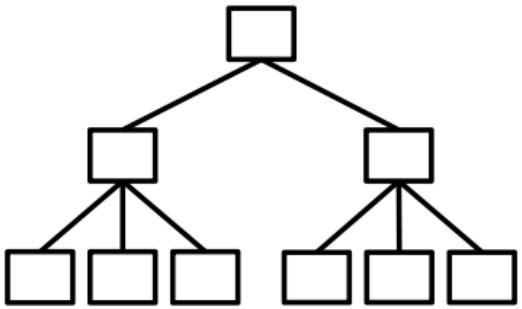
```
'{"name": "John", "age": 30, "car": null,
  "tree" : [
    "key": "value"
  ]
}'
const obj = JSON.parse('{"name":"John", "age":30, "city":"New York"}');
obj.age = obj.age.toString();
```

Data structures: tree hierarchy 2 / XML

Applications:

eXtensible Markup Language (XML):

- DBs: eXist, BaseX
- Query: XPath (w3s) and XQuery (w3s)

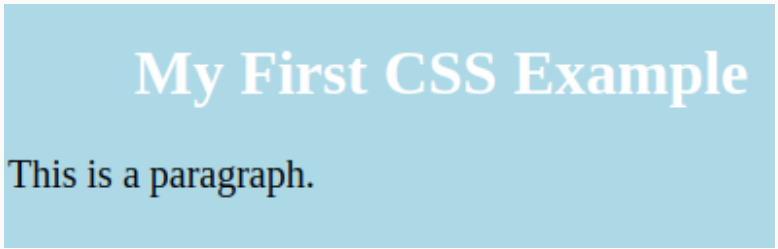


```
<!-- books.xml -->
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book>
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book>
    <title lang="en">Learning XML</title>
    <price>39.95</price>
  </book>
</bookstore>

<!-- XQuery -->
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title

<!-- XPath -->
//title[@lang='en']
/bookstore/book[price>35.00]
```

Data structures: tree hierarchy 3 / web pages (HTML)

HTML (w3s) – structure

```
<!DOCTYPE html>
<html>
 <head>
   <title>Page Title</title>
 </head>
 <body>
  <h1>This is a Heading</h1>
  <p>This is a paragraph.</p>
 </body>
</html>
```



CSS in HTML (w3s) – rendering

```
<!DOCTYPE html>
<html>
  <head>
    <style>
body {
  background-color: lightblue;
}
h1 {
  color: white;
  text-align: center;
}
p {
  font-family: verdana;
  font-size: 20px;
}
    </style>
  </head>
  <body>
    <h1>My First CSS Example</h1>
    <p>This is a paragraph.</p>
  </body>
</html>
```

Why so many data formats?

Different data formats (& standards) focus on different aspects & have different goals:

1. **text-based**
   - Text Encoding Initiative (**TEI**)
   - Extensible Hypertext Markup Language (**XHTML** = XML-compliant HTML)
   - Open Document Format for Office Applications (**ODF**)
2. **page-based**
   - TEX / LATEX
   - **XSL-FO** (XSL Formatting Objects, discontinued)
3. **ontology-based**
   - Resource Description Framework (**RDF**) & RDF Schema (
   - Machine-Readable Cataloging (MARC))
   - Web Ontologie Language (**OWL**)
   - Simple Knowledge Organisation System (**SKOS**)
   - Conceptual Reference Model (**CIDOC-CRM**)
4. **digital archiving / digital objects**
   - Dublin Core Metadata Initiative (**DCMI**), known as Dublin Core (**DC**)
   - Metadata Encoding and Transmission Standard (**METS**)
   - Metadata Object Description Schema (**MODS**)
   - Encoded Archival Description (**EAD**)
   - Charters Encoding Initiative (**CEI**)

A primer on metadata

What are metadata?

- „data about data“
  1. data about containers of data = **structural metadata**
  2. data about the content represented by data = **descriptive metadata**
- functions:
  1. descriptive
  2. administrative
  3. technical
  4. use

There are standards for the description of metadata (and many are XML-based), e.g.

- Machine-Readable Cataloging (**MARC**)
- Metadata Object Description Schema (**MODS**)
- Encoded Archival Description (**EAD**)
- Lightweight Information Describing Objects (**LIDO**)
- Collective Description of Works of Art (**CDWA**) / Visual Research Association (**VRA**)
- Europeana Metadata Model (**EDM**)
- Resource Description Framework (**RDF**)
- Metadata Encoding & Transmission Standard (**METS**)
- Dublin Core (**DC**)
- Functional Requirements for Bibliographic Records (**FRBR**)
- the *<teiHeader>* has metadata…

Slides on metadata in DH with more information

Dublin Core (DC) i

What is the DC?

- founded in Dublin (Ohio) in 1995
- **two levels:** simple (15 elements) & qualified (additional *`Audience`*, *`Provenance`* and *`RightsHolder`*)
- **classes of terms:** elements (nouns) & qualifiers (adjectives).
- can be expressed in RDF/XML
- each element is optional & can be repeated
- also: dc:terms

> The Dublin Core™ metadata **standard** is a simple yet effective **element set for describing a wide range of networked resources.** […] Another way to look at Dublin Core™ is as a "small language for making a particular class of statements about resources". In this language, there are two classes of terms – *elements* (nouns) and *qualifiers* (adjectives) – which can be arranged into a simple pattern of statements.
> (source)

Dublin Core (DC) ii

The Core

i.e. the elements:

1. title
2. subject
3. description
4. type
5. source
6. relation
7. coverage
8. creator
9. publisher
10. contributor
11. rights
12. date
13. format
14. identifier
15. language

Qualified

1. (audience)
2. (provenance)
3. (rights holder)

```
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:dc="http://purl.org/dc/elements/1.1/">

   <rdf:Description rdf:about="http://media.example.com
                               /audio/guide.ra">
      <dc:creator>Rose Bush</dc:creator>
      <dc:title>A Guide to Growing Roses</dc:title>
      <dc:description>Describes process for
        planting and nurturing different kinds
        of rose bushes.</dc:description>
      <dc:date>2001-01-20</dc:date>
   </rdf:Description>
</rdf:RDF>
```

Note the two namespaces *rdf:* and *dc:*.

Metadata Encoding and Transmission Standard (METS)

METS

- tool for encoding digital library objects
- container format for documents in which contents of different formats can be integrated
- also describes relationships between objects
- describes logical and physical structure of an object
- also contains descriptive (bibliographical) and administrative metadata
- relatively simple and straightforward
- supports a wide range of materials
- <website> & more info here

Only structural map is required.

```
<mets>
  <metsHdr/>
  <dmdSec/>
  <amdSec/>
  <fileSec/>
  <structMap/>
  <structLink/>
  <behaviorSec/>
</mets>
```

Metadata Object Description Schema (MODS)

MODS

- can represent the major elements from a MARC record

→ represents key bibliographic data in easily understandable names

→ easier to understand than MARC (for the uninitiated)

- bridges the gap between library application and bibliographic source that don't make use of cataloging metadata formats
- richer than the Dublin Core (DC) but less detailed than MARC
- partially backwards compatible with MARC

```
<mods:mods xmlns:mods="http://www.loc.gov/mods/v3">
  <mods:titleInfo>
    <mods:nonSort>The </mods:nonSort>
    <mods:title>
      1946 Library of Congress recital
    </mods:title>
  </mods:titleInfo>
  <mods:relatedItem type="constituent"
                    ID="DMD_disc01_tr001">
    <mods:titleInfo type="uniform">
      <mods:partName>Chaconne von Vitali
        </mods:partName>
    </mods:titleInfo>
  </mods:relatedItem>
  <mods:identifier type="lccn">99594334
    </mods:identifier>
</mods:mods>
```

Code example from here.

Encoded Archival Description (EAD)

EAD

is a standard for encoding descriptive information regarding archival records

example EAD & Wikipedia (source of the example)

```
<eadheader>
   <eadid countrycode="us" identifier="bachrach_lf">
     bachrach_lf</eadid>
   <filedesc>
      <titlestmt>
         <titleproper encodinganalog="Title">
          Louis Fabian Bachrach Papers</titleproper>
         <subtitle>An inventory of his papers at
           Blank University</subtitle>
         <author encodinganalog="Creator">Mary Smith</author>
      </titlestmt>
      <publicationstmt>
         <publisher encodinganalog="Publisher">
           Blank University</publisher>
         <date encodinganalog="Date" normal="1981">
           1981</date>
      </publicationstmt>
   </filedesc>
   <profiledesc>
      <creation>John Jones
         <date normal="2006-09-13">13 Sep 2006</date>
      </creation>
   </profiledesc>
</eadheader>
```

Charters Encoding Initiative (CEI)

CEI

considers the possibilities of a standard to encode medieval and early modern charters with XML.

- implements the *Vocabulaire Internationale de Diplomatique*
- founded in 2004
- MOM-CA, the collaborative charter archive of Monasterium.net works with CEI
- <website> & an example. Also: the example below.

```
<text type="charter">
  <idno>600202b</idno>
  <chDesc id="a">
    <head>Prag, 1360 Febr. 2.</head>
      <issued>
         <placeName>Prag</placeName>
         <date>1360-02-02</date>
      </issued>
      <abstract>
         <p> Karl verspricht Ludwig ...
         </p>
      </abstract>
      <witList>
         <witness sigil="B">
            Brandenburgisches LHA Potsdam
            “Rep. 37 Hohennauen Nr. 683,
            fol. 225” (18. Jh.)
         </witness>
      </witList>
      <diplomaticAnalysis>
         <bibl type="D">Fidicin, 42...</bibl>
      </diplomaticAnalysis>
   </chDesc> <tenor> </tenor>
</text>
```

Resource Description Framework (RDF)

RDF

- framework for describing resources in the World Wide Web
- can contain metadata
- language of the ‘Semantic Web’ (web 3.0) – makes things machine-processable
- RDF Schema (RDFS) offers Classes and Properties

RDF/ turtle notation (`.ttl`) example from before

```
@prefix ex: <http://example.com/#> .

ex:Graz a ex:city;
ex:name "Graz" ;
ex:inhabitants 288806 ;
ex:location [ ex:lat 47.4; ex:long 5.26 ] .
```

Simple Knowledge Organization System (SKOS)

SKOS

RDF vocabulary for representing semi-formal *knowledge organization systems* (KOSs), such as thesauri, taxonomies, classification schemes and subject heading lists.
$\rightarrow$ less rigorous than the logical formalism of ontology languages such as OWL

(SKOS primer)

```
@prefix skos:
  <http://www.w3.org/2004/02/skos/core#> .
@prefix rdf:
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs:
  <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix dct: <http://purl.org/dc/terms/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix ex: <http://www.example.com/> .
@prefix ex1: <http://www.example.com/1/> .
@prefix ex2: <http://www.example.com/2/> .

ex:animals rdf:type skos:Concept;
  skos:prefLabel "animals"@en;
  skos:narrower ex:mammals.

ex:mammals rdf:type skos:Concept;
  skos:prefLabel "mammals"@en;
  skos:broader ex:animals.
```

Europeana Data Model (EDM) i

EDM

Model to integrate data sources from different providers and thus improve interoperability:

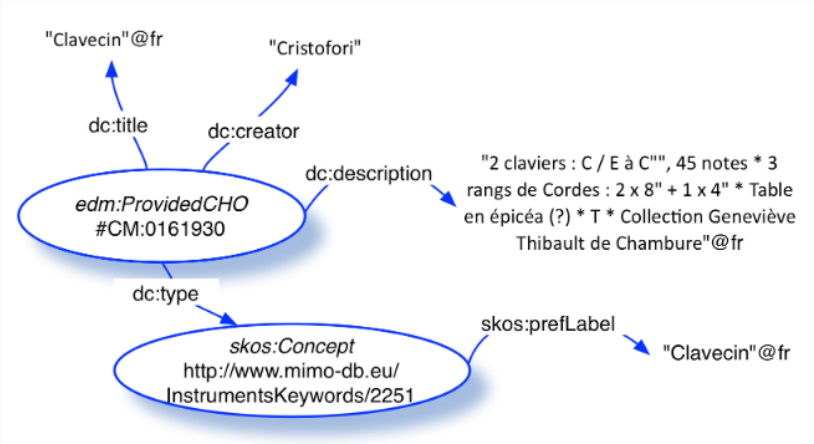
> "EDM transcends domain-specific metadata standards, yet accommodates the range and richness of community standards such as LIDO for museums, EAD for archives or METS for digital libraries." (EDM Factsheet)

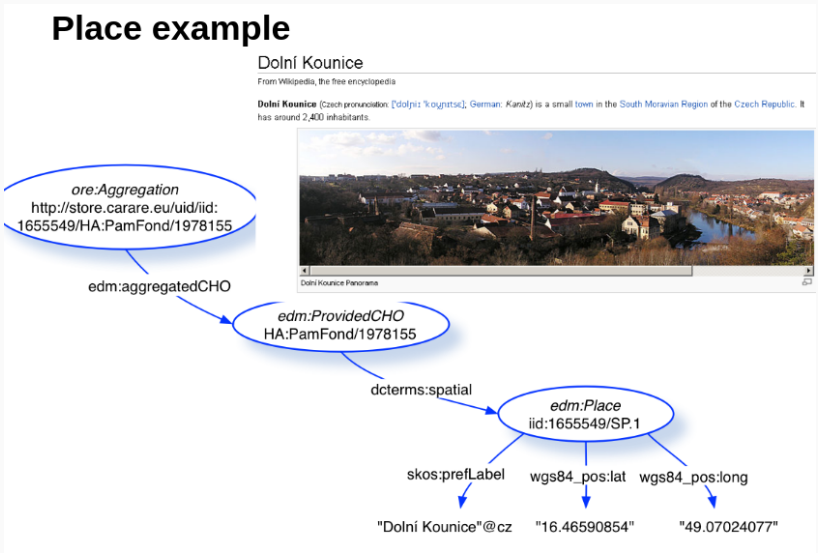
...integrates the following standards:

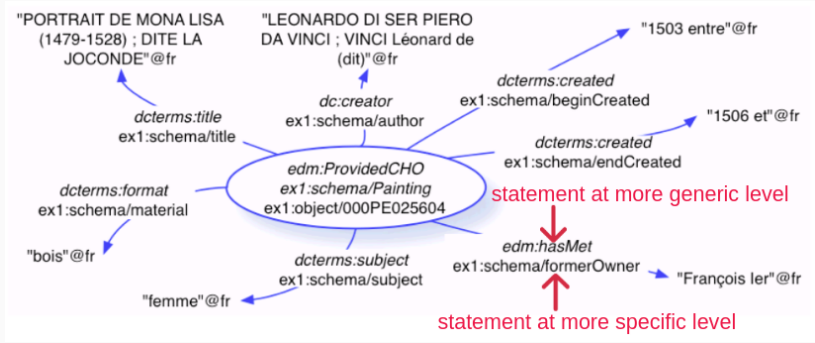
1. **OAI ORE** (Open Archives Initiative Object Reuse & Exchange) for organizing an object's metadata and digital representation(s)
2. **Dublin Core** for descriptive metadata
3. **SKOS** (Simple Knowledge Organization System) for conceptual vocabulary representation
4. **CIDOC-CRM** for event and relationships between objects

This is achieved in RDF (can be written as XML) → RDF uses the Semantic Web principles to integrate those data sources → Metadata standards aren't exclusive, they can be combined!

Europeana Data Model (EDM) ii







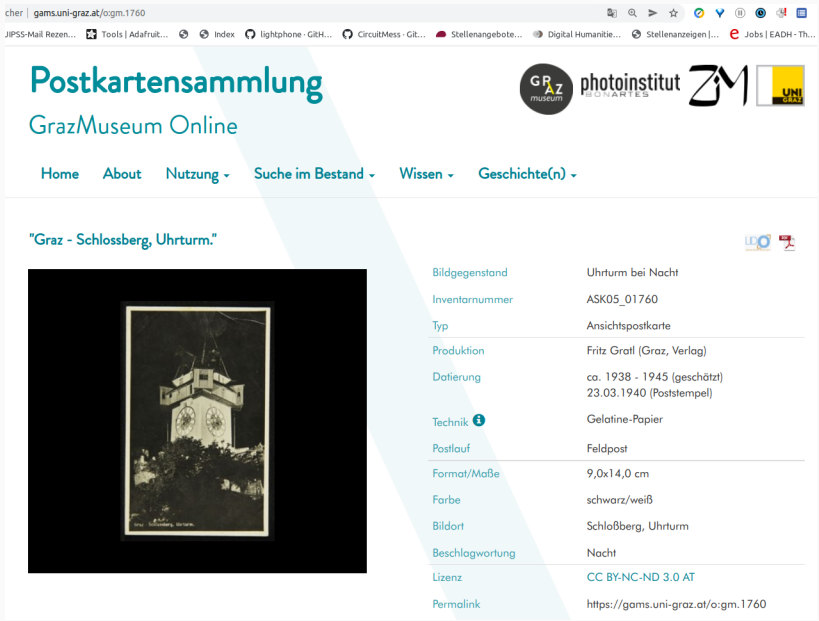
(More info.)

Lightweight Information Describing Objects (LIDO)

LIDO

> Lightweight Information Describing Objects (LIDO) is an **XML schema for describing museum or collection objects.** Memory institutions use LIDO for "exposing, sharing and connecting data on the web". It can be applied to all kind of disciplines in cultural heritage, e.g. art, natural history, technology, etc.
> LIDO is a specific application of CIDOC CRM. (Wikipedia)

A postcard (text-bearing object)




```
<lido:titleWrap>
    <lido:titleSet>
        <lido:appellationValue>
            Chickens and Ducks
        </lido:appellationValue>
    </lido:titleSet>
</lido:titleWrap>
```

LIDO example 1

Notice the two namespaces (*lido:* and *t:*)!

```
<lido:lido
  xmlns:lido="http://www.lido-schema.org"
  xmlns:t="http://www.tei-c.org/ns/1.0">
  <lido:lidoRecID lido:type="PID">o:gm.1760</lido:lidoRecID>
  <lido:category>
    <lido:conceptID lido:source="CIDOC"
                    lido:type="ID">E22</lido:conceptID>
    <lido:term xml:lang="eng">Man-Made Object</lido:term>
    <lido:term lido:label="info:fedora/context:gm">
      Postkartensammlung Online</lido:term>
  </lido:category>
  <lido:descriptiveMetadata xml:lang="deu">
    <lido:objectClassificationWrap>
      <lido:objectWorkTypeWrap>
        <lido:objectWorkType>
          <lido:conceptID lido:source="http://vocab.getty.edu/aat"
            lido:type="ID">300026819</lido:conceptID>
          <lido:term lido:label="info:fedora/context:gm-ansicht">
            Ansichtspostkarte</lido:term>
        </lido:objectWorkType>
      </lido:objectWorkTypeWrap>
    </lido:objectClassificationWrap>
    <!-- to be continued... LIDO is very verbose --->
```

LIDO example 2

```
<!-- continued... --->

    <lido:objectIdentificationWrap>
      <lido:titleWrap>
        <lido:titleSet>
          <lido:appellationValue>Graz - Schlossberg, Uhrturm.
             </lido:appellationValue>
        </lido:titleSet>
      </lido:titleWrap>
      <lido:repositoryWrap>
        <lido:repositorySet lido:type="orgname">
          <lido:repositoryName>
            <lido:legalBodyID lido:source="http://d-nb.info/gnd"
              lido:type="ID">2022740-1</lido:legalBodyID>
            <lido:legalBodyName>
              <lido:appellationValue>GrazMuseum</lido:appellationValue>
            </lido:legalBodyName>
          </lido:repositoryName>

    <!-- to be continued... LIDO is very verbose --->
```

Intro to HTML (& Bootstrap)

Understanding websites…

Why are we doing this workshop? The motivation from our abstract:

- ✔ But what happens after an edition is encoded in TEI?
- 👍 While it is an **ideal format for archiving digital data**, it is less than ideal for viewing and interacting with the edited text.
- 👍 The data transformation language XSLT allows editors to create multiple representations from their data encoded in XML, enabling the creation of both digital and print editions.

Goals for the next session

- ✔ learn some theory basics
- ✔ everybody on the same page on XML/TEI for digital editing
- ✖ creating websites in HTML (& Bootstrap)

Hyper Text Markup Language (HTML) i

.HTML

Defines the structure of websites. Due to their common origin in SGML: lots of similarity with XML but, unlike XML's extensibility, HTML has a fixed tag set (much less!).

The most important HTML elements to remember

html, head, body. div, p, h1-6. span, ul, ol, li. table, tr, td.

```
<!DOCTYPE html>
<html>
 <head>
  <title>Page Title</title>
  <style> /* preferably in extra file */
   p.important {
    color: green;
    }
  </style>
  <script>
    alert("Hello! I am an alert box!!");
  </script>
 </head>
 <body>
  <h1>This is a Heading</h1>
  <p style="color:red">I am a paragraph</p>
  <p>I like
   <span style="color:blue">blue</span>.</p>
  <p class="important">
   Note that this important!</p>
  </body>
</html>
```

(full example on the next slide)

Hyper Text Markup Language (HTML) ii

```
<!DOCTYPE html>
<html>
 <head>
  <title>Page Title</title>
  <style> /* better in extra file
  h1 {
   color: blue;
   font-family: verdana;
   font-size: 300%;
   }
   p  {
    color: red;
    font-family: courier;
    font-size: 160%;
    }
   p.important {
    color: green;
    }
  </style>
  <script>
    alert("Hello World!");
   </script>
 </head>
 <body>
  <h1>This is a Heading</h1>
  <p>This is a paragraph. <br />
    <a href="https://www.w3schools.com">
    This is a link</a>
  </p>
  <img src="img.jpg"
    width="500" height="600" />
  <p style="color:red">I am a paragraph</p>
  <p title="I'm a tooltip">
    This is a paragraph.
  </p>
  <p>My mother has
     <span style="color:blue">blue</span>
     eyes.</p>
  <p class="important">
    Note that this is an important
    paragraph. :)</p>
 </body>
</html>
```

CSS

CSS = Cascading Style Sheets

.CSS

- included via a stylesheet link in HTML or written inline
- describes the styling of websites
- separation of form & content:
  - **HTML:** content in structured form
  - **CSS:** the layout
  - (**JavaScript:** dynamic parts)
- the Bootstrap framework offers a ready-made, responsive design to reuse: box and grid model
- uses selectors to define layout

```
h1 {
font-family: Arial
}
```

```
p {
font-family: Arial ;
color: red
}
```

```
.person {
    font-weight:bold;
    font-size:smaller;
    font-style:italic;
}
```

Resources

- 15min to Bootstrap
- CSS3-Cheatsheet
- **W3 Schools:** W3schools CSS
- W3S Tables of properties
- CSS ZenGarden

modifying websites dynamically

.JS

- JS can change websites without having to reload them (‘dynamically’).
  remember when you had to manually refresh pages?
- When we want to highlight something using a check box, we use js.
- Unlike HTMl (which is a markup language, think of it like a file format), js is a programming language (like XSLT).
- → we won’t learn this today! But there is a mini example and you will load js components when using Bootstrap and the GAMS wippets later.

A ‘Hello World!’ example

```
1 document.getElementById("demo").innerHTML = "Hello JavaScript";
```

Bootstrap Framework i

> Bootstrap is a free and **open-source CSS framework** directed at **responsive, mobile-first front-end web development.** It contains HTML, CSS and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. (Wikipedia)

How to use Bootstrap?

1. load the scripts in the HTML head
2. figure out new Bootstrap elements using W3Schools Tutorials.
   - *https://www.w3schools.com/howto/*
   - *https://www.w3schools.com/bootstrap/*

Bootstrap Framework ii

We need to include the bootstrap stylesheet from this <link> in the HTML *<head>* but also the javascript plugins.

Careful, *<head>* in HTML is equivalent to the *<teiHeader>*. Headings in HTML are called: *<h1>-<h6>*.

Under <examples>, Bootstrap offers different types of ready-to-use websites. I have included one in our practice XSL templates. → Link to the source-Code of the starter template → append it in the address or click right and 'inspect element' or sth)

```
<html>
    <head>
    <meta charset="utf-8">
    <link rel="stylesheet" href="https://...bootstrap.min.css">
    <script src="https://code.jquery.com/jquery-3.2.1.slim.min.js"></script>
    </head>
    <body> [...] </body>
</html>
```

Bootstrap Framework iii

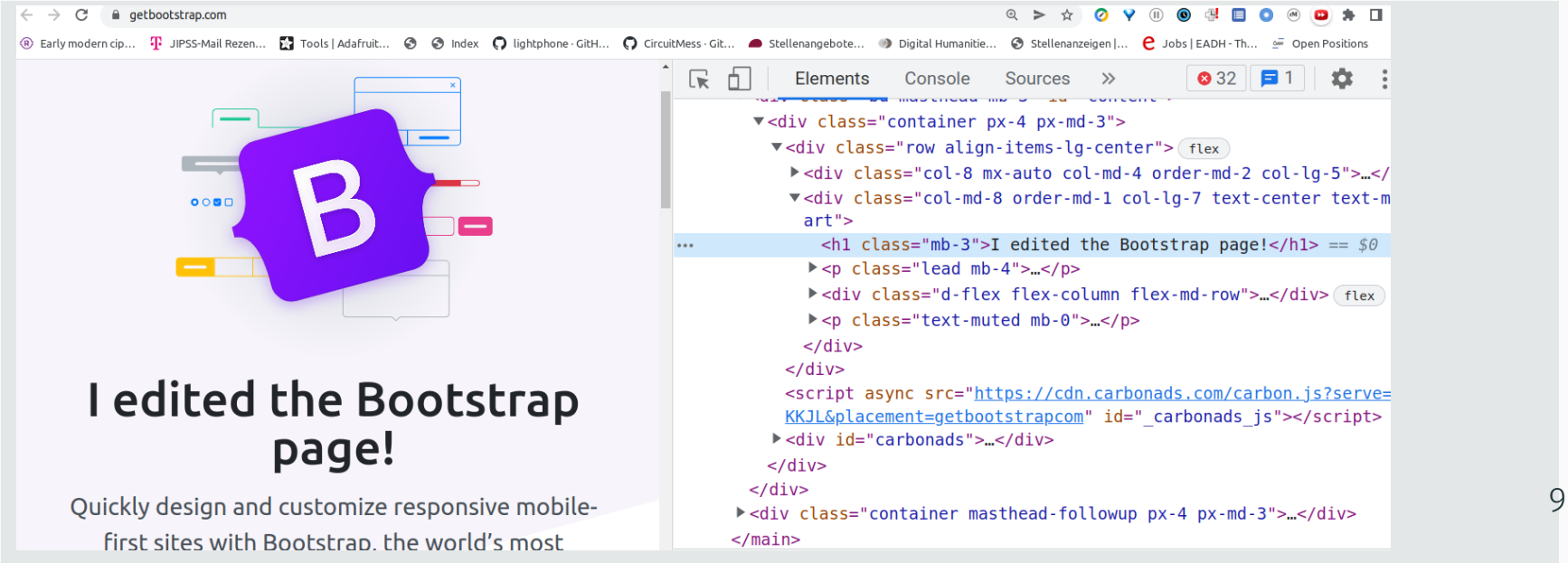
```
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <!-- Bootstrap CSS -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/
    bootstrap.min.css" rel="stylesheet"
    integrity="sha384-1BmE4kWBq78iYhFldvKuhfTAU6auU8tT94WrHftjDbr
    CEXSU1oBoqyl2QvZ6jIW3" crossorigin="anonymous">

    <title>Hello, world!</title>
  </head>
  <body>
    <h1>Hello, world!</h1>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/
    js/bootstrap.bundle.min.js"
    integrity="sha384-ka7Sk0Gln4gmtz2MlQnikT1wXgYsOg+
    OMhuP+IlRH9sENBO0LRn5q+8nbTov4+1p"
    crossorigin="anonymous"></script>

  </body>
</html>
```

Resources

1. Lessons 1 and the start of 2 on Dash (signup required).
2. Codecademy HTML (good but verbos)
3. Interneting is hard:
    - Intro
    - Chapter 2: HTML Basics
4. Mozilla Learn HTML & in German
5. W3Schools interactive HTML or respectively the more text based HTML tutorial.
6. Learn HTML.org
7. Web Design in 4 minutes (CSS)
8. W3Schools tutorials

`.html` files and your browser

- clicking on a ***`.html`*** file will open up a browser (which parses and interprets the code). This doesn't mean it's on the internet! (check the address line, it links to your local computer)
- ergo: the webbrowser's job is downloading a website's code, i.e. getting it for you, and also reading & displaying that code. The second part is what happens when you open a local ***`.html`*** file.
- To inspect the code of a local file, open as text (for example in Oxygen XML).
- To inspect a web page's code online, right-click and chose 'inspect' (or sth like that). In the Dev Tools, you can edit this code (only on your local browser, of course), try it out!



HTML Practice!

To understand the basics of HTML and the ‘web triad’ (HTML, CSS, JS) better, do one of the following exercises:

‘Dash’ HTML / CSS tutorial (great but signup required)

W3Schools tutorial

Start working on the XPath/XSLT exercise sheet (resources/materials folder), XSLT 2.1 part. Feel free to use the slides and cheatsheet for help.

Intro to LaTeX (& `reledmac`)

Producing PDFs…

Why are we doing this workshop? The motivation from our abstract:

- ✔ But what happens after an edition is encoded in TEI?
- 👍 While it is an **ideal format for archiving digital data**, it is less than ideal for viewing and interacting with the edited text.
- 👍 The data transformation language XSLT allows editors to create multiple representations from their data encoded in XML, enabling the creation of both digital and print editions.

Goals for the next session

- ✔ learn some theory basics
- ✔ everybody on the same page on XML/TEI for digital editing
- ✔ creating websites in HTML (& Bootstrap)
- ✖ creating PDFs & print(able) editions using LaTeX (& `reledmac`)

- `.tex` Typesetting with TeX using Lamport macros (i.e. shortcuts to make complicated code easy)
- LaTeX reads in those macros (with telling names like `\emph{}` for 'emphasis', an 'intelligent' command which can be redefined for the whole document)
- feels like markup for those producing the code (illusion of descriptive markup)
- placeholder for complex procedural language
- **WYIWYG**-Editors (*what you see is what you get*, i.e. MS Word) vs. **WYSIWYM** (*what you see is what you mean* → LaTeX)

Example commands

Commands below without the spaces

`\textit {Italic}` ..... *Italic* (presentational.)

`\emph {Italic}` ..... *Italic* (semantic/'intelligent')

`\textbf {Bold}` ..... **bold face**

`\section {Title}` ..... Heading 1

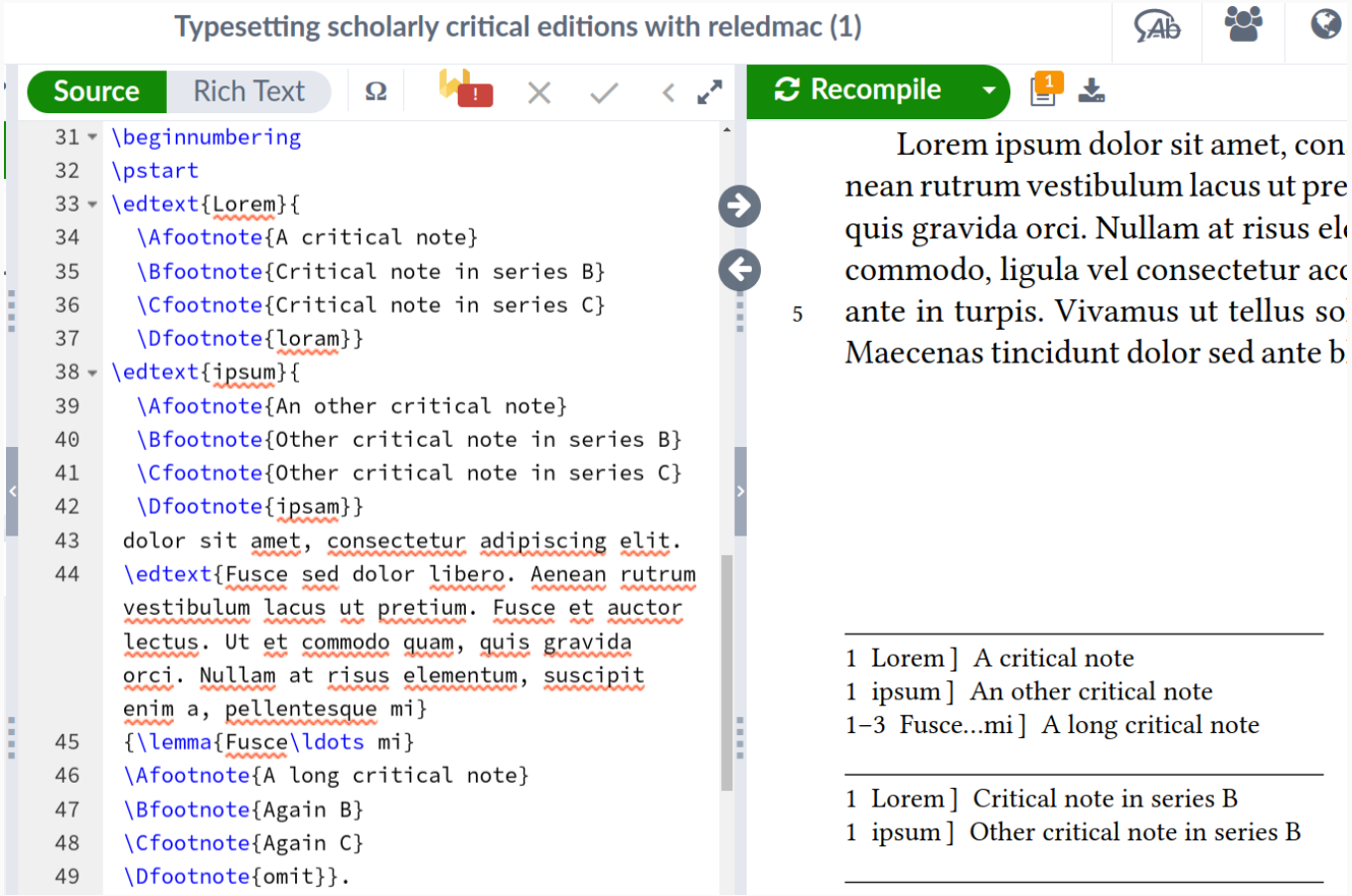
`\subsection {Subtitle}` .... etc.

`\href {http://a.com}{Link}` .. 'hidden' link

`\includegraphics {bla.png}` ..... image

Typesetting critical editions with the `reledmac` package i

→ *`reledmac`* example



Typesetting critical editions with the `reledmac` package ii

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Fusce sed dolor libero. Aenean rutrum vestibulum lacus ut pretium. Fusce et auctor lectus. Ut et commodo quam, quis gravida orci. Nullam at risus elementum, suscipit enim a, pellentesque mi. Morbi commodo, ligula vel consectetur accumsan, massa metus egestas velit, eu fringilla leo
5 ante in turpis. Vivamus ut tellus sollicitudin, facilisis ipsum sit amet, tincidunt odio. Maecenas tincidunt dolor sed ante blandit tincidunt. Etiam vulputate ultricies facilisis.

---

1 Lorem] A critical note
1 ipsum] An other critical note
1–3 Fusce...mi] A long critical note

---

1 Lorem] Critical note in series B
1 ipsum] Other critical note in series B

1–3 Fusce...mi] Again B

---

1 Lorem] Critical note in series C

1 ipsum] Other critical note in series C

1–3 Fusce...mi] Again C

---

1 Lorem] loram 1 ipsum] ipsam 1–3 Fusce...mi] omit

Typesetting critical editions with the `reledmac` package iii

```
\documentclass{article}
\usepackage{polyglossia,fontspec,xunicode}
\usepackage{libertine}
\setmainlanguage{latin}
\setotherlanguage{english}

\usepackage[series={A,B,C,D},noend,noeledsec,
            nofamiliar,noledgroup]{reledmac}
\Xarrangement[B]{twocol}
\Xarrangement[C]{threecol}
\Xarrangement[D]{paragraph}
\begin{document}

\title{Critical notes}
\maketitle

\beginnumbering
\pstart
\edtext{Lorem}{
  \Afootnote{A critical note}
  \Bfootnote{Critical note in series B}
  \Cfootnote{Critical note in series C}
  \Dfootnote{loram}}
\edtext{ipsum}{
  \Afootnote{An other critical note}
```

Explantation of the `reledmac` example i

The following loads the *reledmac* package with the option to have four different sets of critical notes.

```
\usepackage[series={A,B,C,D},noend,noeledsec,
            nofamiliar,noledgroup]{reledmac}
\Xarrangement[B]{twocol}
\Xarrangement[C]{threecol}
\Xarrangement[D]{paragraph}
```

Explanation

This defines the following arrangement:

Each note of type A gets its own paragraph.

Each note of type B gets its own paragraph but notes arranged in 2 columns.

Each note of type C gets its own paragraph but notes arranged in 3 columns.

Each note of type D is in the same paragraph.

Explantation of the reledmac example ii

This is one note – ‘lorem’ in the \edtext-command is the lemma on which the notes ‘hang’, i.e. we open a nested list. With \Afootnote we specify that the note should be added to the A apparatus (and so forth).

```
  \edtext{Lorem}{
\Afootnote{A critical note}
\Bfootnote{Critical note in series B}
\Cfootnote{Critical note in series C}
\Dfootnote{loram}}
```

The superstructure of all this is

```
\beginnumbering
\pstart
[one paragraph of edition stuff]
\pend
\endnumbering
```

Explantation of the `reledmac` example iii

Special usecases, like a note spanning multiple lines, goes like this:

```
\edtext{Fusce sed dolor libero. Aenean rutrum vestibulum lacus ut
pretium. Fusce et auctor lectus. Ut et commodo quam, quis gravida
orci. Nullam at risus elementum, suscipit enim a, pellentesque mi}
{\lemma{Fusce\ldots mi}
\Afootnote{A long critical note}
[...]
}
```

With `\lemma` we define what the abbreviated lemma should look like in the apparatus.

The overleaf example also contains three more documents you might want to look at: *`sidenotes.tex`*, *`tabular.tex`* and *`verses.tex`*. To get the result of their code, click into the document and then click 'Recompile' from there.

Explantation of the `reledmac` example iv

Further reading

- **A review on the RIDE journal:** *Reledmac. Typesetting technology-independent critical editions with LaTeX* = Reledmac, Maïeul Rouquette (ed.), 1987–2019. *https://ctan.org/pkg/reledmac* (Last Accessed: 21.07.2019). Reviewed by Andrew N. J. Dunning. In *RIDE – A review journal for digital editions and resources* 11 (Tools and Environments), 2020. *https://ride.i-d-e.de/issues/issue-11/reledmac/*.
- **Package documentation:** For more info, see the *reledmac* package documentation (496 pages). You can make indices, glossaries – all sorts of options are documented here. Also: read up on the history of the package.
- **Marjorie Burghart's TEI Critical Apparatus Toolbox** contains a tool to turn TEI into *reledmac* PDF edition. → XSLT is here.
  - The web service gives you a *.zip* with the *.tex* code and the PDF generated from it. Or you can just transform it yourself, using the XSL from Github.
  - Our template (*mini-latex-reledmac.xsl*) is a simplified version of it – but with your new XSLT skills, you can make it your own (take what you need, leave what you don't).

LaTeX Practice!

To understand the basics of LaTeX better, open the Humanities' seminar paper with LaTeX in 10min Overleaf template. Read the code, comments and understand the resulting PDF.

Open the `reledmac` template on Overleaf and play around with it.

Start working on the XPath/XSLT exercise sheet (resources/materials folder), XSLT 2.1 part and answer the same questions for LaTeX (for example, taking the above `reledmac` template into account). Feel free to use the slides and cheatsheet for help.

Navigating XML using XPath

Slowly moving beyond TEI…

Why are we doing this workshop? The motivation from our abstract:

- ✔ But what happens after an edition is encoded in TEI?
- 👍 While it is an **ideal format for archiving digital data**, it is less than ideal for viewing and interacting with the edited text.
- 👍 The data transformation language XSLT allows editors to create multiple representations from their data encoded in XML, enabling the creation of both digital and print editions.

Goals for the next session

- ✔ understand the terms from the abstract
- ✔ everybody on the same page on XML/TEI for digital editing
- ✔ creating websites in HTML (& Bootstrap)
- ✔ creating PDFs & print(able) editions using LaTeX (& `reledmac`)
- ✖ ~~creating different representations from our data~~ → first: navigating XML documents automatically using XPath

Introduction to XPath i

XPath can be used to navigate in XML documents.
//persName = 'Give me all personal names / persons in the whole document.' → all findings will be highlighted in Oxygen. I have thus 'found' or 'addressed' them. I can 'select' them, for example to process them in XSLT.

Every occurence of Mina would result in the following once 'translated' to XPath:

1. All *<persName>* which have *@ref='#Mina'*, *in the whole document*
2. all *<persName>* in the whole document: *//persName*.
3. … having *@ref='Mina'*:

```
//persName[@ref='#Mina']
```

4. Try it out! (Use the XPath field in Oxygen – if you don't have any, look in the settings/preferences: Window → Tools: check 'XPath'.)

Introduction to XPath ii

How to find something using XPath:

1. Formulate a normal English sentence expressing what you wish to achieve, e.g. ‘I want all occurrences of the person Mina.’
2. ‘Translate’ this to the technical structure of the document: All those occurrences are encoded as `persName ref="#Mina"` – this gives me the means to address them.
3. Do I want to get the element nodes themselves (for further processing) or just their text contents? Or the result of a function? (‘Count all occurrences and give me their number.’)?
4. Are there conditions? Do I want just some specific elements (a subset of the above) or everything that is found (‘matched’) by my request?

paths & scope i

Depending on the query, I can search *in the whole document* or *in a certain scope*. To do that, I need to distinguish:

relative paths

Not every step in traversing the tree hierarchy of XML is listed but the query is run depending on the cursor's or programme's current location. Thus I need to know my current position.
This is useful because I only need to know about what's below that point in the tree (a limited scope). It's bad if I don't know where I am and get other results than expected. If you're not sure, prefer absolute paths!
In XSLT, we will need relative paths.

absolute paths

Every step in the hierarchy is stated explicitly. I only get the desired path (not accidentally matching something else).

```
//titleStmt/title
```

→ returns only the document title from the *<teiHeader>*, not other titles from bibliographical references.
Gets more complicated if the result comes from different places (use 'or' statements here: ||).

paths & scope ii

scope

It's possible that I continue working with a subset of all nodes in XSLT, for example following a `<xsl:for-each>` loop. Here I have to use relative paths because I can't know at the time of writing the code where such elements will be found.

A relative path can lead you to find unwanted things: maybe you get more or less than expected. Maybe you're not in the right place and the desired query doesn't yield any results in the current scope.

So how does XPath work? i

query result = set of nodes

A query returns a set of nodes. If you put a dot (`.`), you get just the text content of the current node.

`matches(.,'([0-9]{4})')` Does 'everything' (current node text content) contain a four letter number (i.e. data)? Answer: yes/no. → no direct access to the contents found, just a yes/no answer is the result!

`//*[matches(.,'([0-9]{4})')]` = 'Give me any node in the whole document which contains (nothing but) a four digit numer/date' = returns a set of nodes to which the answer to the question in the brackets (XPath condition) is 'yes' = 'true'.

Difference between strings & nodes

Once XPath is in the substring, it only has characters (strings) left to access, not nodes/elements. Same is true for dot (`.`). Some functions return an answer, some make changes (like *`translate()`*). You might get the error *`XPath failed due to: A sequence of more than one item is not allowed as the first argument of`*. This is because some functions only allow one element or one char-litteral as input, example:

```
contains(//t:origPlace/@ref, 'pleiades')
```

Tests results in error if there is more than one *`<origPlace>`* element in a document because many string operations only allow a char-literal or one element, not a set of nodes as input. The error can appear even when there actually is only one element where the node test is true (only one *`@ref='pleiades'`*). Solution:

```
//t:origPlace/@ref[contains(., 'pleiades')]
```

Also, you can only 'click into' (jump to) the results if it's a set of notes, not a string or yes/no answer.

So how does XPath work? ii

predicates [] = conditions

Predicates are conditions and noted [in brackets]. You can query the result of a function there (yes/no) and thus, make queries like: Give me all personal names if/where XY is true.

functions

Functions can go into predicates but also around the whole expression → in the latter case, you usually get an 'answer', not a set of nodes (which you probably wanted). Can result in the error explained on this slide. (Kind of) Like a function in math, you get just value as an answer – not a set of results.

```
function(//location, parameter)
```

Unlike in normal search an replace, we can now query conditions: "Give me all occurrences of the person Mina but only in chapters where Dracula doesn't appear."

So how does XPath work? iii

Absolute and relative paths

Try playing around with absolute and relative paths: To use a relative path, click into the *<titleStmt>* and search for just:

```
title
```

Try all these options. Are there differences?

```
title
/TEI/teiHeader//title
//titleStmt/title
//title
```

The last option will give you all titles, so if there's more than just the document title in the header, you will get all of them. *//* always give you the global scope.

```
//person//surname
//persName[@ref]
//persName/@ref
//persName[@ref="bla"]
//persName[contains(.,'bla')]
contains(.,'bla') = yes/no, no nodes
```

Negating queries

Negate the query with *!=* OR *not*. Example: A *<persName>* which doesn't have an *@ref* attribute:

```
//persName[@ref]
//persName[not(@ref)]
```

All persons except Mina:

```
//persname[@ref !='Mina']
```

Examples to get started

See exercise sheet.

```
1. //div[@type="chapter"]   -- count(//head)
2. //head -- //div[@type='chapter']/head
3. substring-after(//div[type=....],'Chapter')
4. distinct-values(//placeName/@xml:id)
  OR distinct-values(//placeName)
```

→ What's the difference?

```
5. distinct-values(//persName[@ref="#Mina"]
7. //p[contains(.,'Christmas')]
  OR //p[matches(.,[A-Z][a-z])]
8. //div[@type="chapter"][@n="4"]/head
  OR  //div[@type="chapter"][4]/head
9. //persName[@ref != "#Mina"]
  OR \\persname[not(@ref='#Mina')]
10. //div[@type='chapter'][1]/persName | //div[@type='chapter'][1]/placeName
OR //person | //place
```

XPath i

XML document = tree structure → navigate via paths

- root node (/) ≠ root element = child of the root node.
- Additionally: attribute node, text node, element node, (+ namespace + comment + processing instructions)
- Query = path, result = node set / value.
- part of the XSL language family (XPath, XSLT, XSL-FO, ...XQuery).

absolute (*/person/name*) OR
relative (*../../title*)

Syntax

- *axis name::nodetest[predicate]*
- axis = direction of movement
- node test: which node?
- predicate: condition.

XPath ii

/child::person[attribute::gender]/child::name ..... long
/person[@gender]/name ..... shorthand

axes

::self ..... current node itself (shorthand: .)
::ancestor ..... all ancestors of current node
attribute:: ..... attributes of current node (**shorthand: @**)
child:: ..... direct children of current node
descendant:: ..... ancestors of current node (**shorthand: //**)
parent:: ..... parent of current node (**shorthand: ../**)
preceding-sibling:: ..... preceding siblings of current node
following-sibling:: ..... following siblings of current node

Navigating XML

XPath expressions are evaluated in their context ● absolute & relative paths are possible (*dynamic & static context* ● namespaces ● functions, variables

XPATH basics

`/` ..... root node / one step further down the tree

`::*` ..... ::* = all types of elements irrespective of name

`@` ..... attribute

`[number]` ..... rank in result set

`text()` ..... text content of element

`::elementtype` ..... condition for element type (node test)

`//` ..... at any depth

`[text() = 'text content']` ..... condition (predicate)

`.` ..... self

`(/../` ..... jump one hierarchical level regardless of name

`path//subpath` ..... arbitrary depth

XPath predicates

Conditions which define subsets = predicates

```
//person[@gender="female"]
//person[firstname = "Stefan"]
//person[firstname != "Tanja"]
//person[1]
//person[last()]/firstname
//person[position() = 2]
/participants/person[position()=5]/firstname
/participants/person[last()]/firstname
count(child::*)
```

XPath functions

functions – set of nodes

`position()` ..... position of current node, return: number

`count()` ..... number of nodes in given node set

`last()` ..... last node in selected node set

string functions

`substring-before(value, substring)` .... substring of text node (*value*) coming before the indicated *substring*

`substring-before(., 'mina')` ..... e.g.

`substring-after(value, substring)` ..... like before

`substring(value, start, length)` ..... with start pos & length

`string-length(.)` ..... length of current text node

`concat(value1, value2, ...)` ..... concatenate

`concat(surname, ', ', forename)` ..... e.g.

`translate(target, string, replacement)` ..... replace

`noramlize-space(target)` ..... removes trailing whitespace

XPath functions

Boolean functions

`starts-with(value, substring)` ..... return: true/false

`starts-with(., 'D')` ..... true/false

`contains(value, substring)` ... if the current text node contains the string, returns yes

`not(comparison)` ..... returns yes if not XY

Functions with regex support

`matches(target, 'RegEx')` .. like `contains()` but only exact matches

`replace()` ..... like `translate()` but supports regex

`tokenize()` ..... tokenize text into single words

Troubleshooting i

1. **[Nothing works]** In case of doubt it's always the namespace! Or you're somewhere other than you expected (using relative paths) → try an absolute path!
2. **[Nothing found]** XML is **hierarchical**! Am I in the right spot? Can I reach the degree of **depth** that I wanted? In doubt, include `/../` to be more independent of hierarchy.
3. **[Not found enough?] Jumps in path depth:** A path (without `path//subpath`) in between will not jump any levels, only go one step deeper! Are there in-between layers you forgot? Like another nested `<div>`? Try putting `//` but then, depth is completely arbitrary (maybe also not the wanted outcome) – if you want a particular depth, like 3, try `/../../../`.
4. **[Path works but doesn't do what I wanted]** Formulate your query as precise as possible. Do I find all I want? Does my query match more than I wanted?
5. XSL's standard processing behaviour causes text content to be printed as is unless it's explicitly changed or suppressed.

Troubleshooting ii

Preparing XSL transformations

First query all elements used in your file (once each using the `distinct-values()` function) to be sure that you have everything covered or learn what you need to write template rules for.

`distinct-values(//@*/name())` ..... What types of attributes are there?
`distinct-values(//@*)` ..... What attribute values are in my document?
`distinct-values(//@rend)` ..... What values does `@rend` take?
`distinct-values(//name())` ..... What elements are there?

XPath practice!

Start working on the XPath/XSLT exercise sheet (resources/materials folder); XPath 1.1 & 1.2. It has lots of explanations. Feel free to use the slides and cheatsheet for help. Read the info on the worksheet!

XML transformations using XSLT

Finally: Beyond TEI!

Why are we doing this workshop? The motivation from our abstract:

- 👍 While it is an **ideal format for archiving digital data**, it is less than ideal for viewing and interacting with the edited text.
- 👍 The data transformation language XSLT allows editors to create multiple representations from their data encoded in XML, enabling the creation of both digital and print editions.

Goals for the next session

- 👍 Navigating XML using XPath
- ✖ What is XSLT?
- ✖ Setting up a transformation scenario
- ✖ Creating different representations from our data using XSLT
  - HTML (& Bootstrap)
  - LaTeX (& *reledmac*)

What is XSLT? i

XSL (*eXtensible Stylesheet Language*) is a programming language for transforming XML data. This allows for the storage of presentation-independent base data from which different representations can be generated (i.e. structure information for HTML).

This is called the *single source principle.*

XSLT (=XSL-Transformations) is often used as a synonym but actually is only one part like **XSL-FO** (*Formatting Objects* for describing print layouts in PDF; discontinued since 2012).

XSLT is made up of a set of processing rules (*templates*) according to which an XML document is traversed and processed. The rules are being applied whenever one matches the current node/content.

What is XSLT? ii

The parsing begins at the root node wich is why the first template is always this:

```
<xsl:template match="/">...</xsl:template>
```

Here, you can build your output document structure before the rest of the processing begins.

If no matching template is encountered, built-in rules are used (printing the text content of elements and their children. You will encounter this as content just being dumped into your output unformatted (usually, you only want the *<body>* printed and not all metadata of the *<teiHeader>* in unstructured form. Thus you need to explicitly suppress this built-in behaviour.

→ **In German:** IDE Kurzreferenzen ● **In English:** W3Schools tutorial

XSLT output formats

XSL(T): eXtensible Stylesheet Language (Transformations)

.XSL

An XSL stylesheet

is an XML document itself and describes rules for the transformation process of an input XML file (into one or multiple output formats).

Output formats are, for example, ...

- (x)HTML
- **XML** → e.g. Word, TEI and other XML standards, RDF, SVG
- **Text** → e.g. LaTeX (→ PDF), RTF, MARC.

Be careful! LaTeX is a lot like markup, too, but has different protected entities than XML and other conventions for 'escaping' them: LaTeX ampersand = `\&` versus XML = `&`
Use `xsl:output` and check for errors!

Getting started with XSL Transformations

Open an XML you want to transform & a new XSL stylesheet in Oxygen.

Notice that...

1. It's empty (the *<stylesheet>* has only attributes).
2. *<stylesheet>* has a namespace (*<xsl:stylesheet>*)

The auto-generated new *.xsl* document will look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Tran
  xmlns:xs="http://www.w3.org/2001/XMLSchema
  exclude-result-prefixes="xs"
  version="2.0">

</xsl:stylesheet>
```

Namespace

Namespaces are defined in *xmlns* attributes:

- the link is the reference for the schema
- after *xmlns:* is the shorthand for the namespace (*xsl*)
- later we will address TEI as *t:TEI*
- this avoids mixups between the sometimes similar elements of HTML and TEI (*html:p* and *t:p*).
- what you call the namespace doesn't matter (*tei:* or *t:* both work) → shorter is better as long as it's not confusing

Configuring the transformation scenario

Click into the XML file & configure a transformation scenario (tool symbol):

1. Select ‘new’ (XSLT transformation).
2. **XSLT tab:**
   - 2.1 select XML file (probably already has the shorthand *`${currentFileURL}`*)
   - 2.2 select our new XSL file (needs to be saved)
   - 2.3 (you can do mass-transformations using the project options)
   - 2.4 select transformer: Saxon (any 9 version).
3. **XSL-FO tab:** Doesn’t concern us → to create PDFs in a manner alternative to LaTeX but not maintained since 2012
4. **Output tab:**
   - 4.1 ‘Save as’ → click green arrow, select *`${cfn}`* (‘current file name’). Then add *`-transform.xml`*, so our original file doesn’t get overwritten. (Whenever you transform, the transformation result file will be overwritten – avoid that by adding an ID / renaming it each time).
   - 4.2 Pick ‘Open in editor’ & show as XML (if XML or HTML if HTML). Also say ‘open as XML’ for LaTeX: ignore the complaints that ensue ☺
5. **Ok, then run it.** Now our – as of yet empty – new stylesheet (transformation) will be applied to the original file.

XSLT practice!

Set up your transformation scenario and run it.

What do you notice?

This is the standard behaviour of XSLT in the absence of matching processing rules (templates).

standard transformation scenarios i

With the last template, we indicated that we wanted the standard processing to be applied. This is just printing the text contents. → the resulting document isn't actually XML anymore! (it has no elements and no root)

Empty 'match root' template (<xsl:template match="/">)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="/">
        <!-- What happens now? -->
    </xsl:template>

</xsl:stylesheet>
```

standard transformation scenarios ii

pushing the XML contents through the XSL tree using `apply-templates`

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="/">
        <xsl:apply-templates/>
    </xsl:template>

</xsl:stylesheet>
```

standard transformation scenarios iii

‘Hard code’ structure into that root template

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="/">
        <xml>
            <xsl:apply-templates/>
        </xml>
    </xsl:template>

</xsl:stylesheet>
```

(i.e. type it in there)

Matching templates

Instead of laboriously hardcoding everything, we can also define templates which always become active whenever a certain element (that they match as indicated by the *@match* attribute) is encountered.

Read: Whenever you find *<p/>*, do XY. We declare a 'namespace' (*xmlns*) called *t:* so our TEI data gets recognized as such:

```
xmlns:t="http://www.tei-c.org/ns/1.0"
```

When this line is present in the *<xsl:stylesheet>* root element, paths from our TEI file always need to have the prefix *t:*, e.g. *t:p*.

Read: *<p>* from the namespace with the shorthand *t:* as per the definition in the *<xsl:stylesheet>* root element.

processing/creating <p> elements in the output

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3
    xmlns:xs="http://www.w3.org/2001/XML
    xmlns:t="http://www.tei-c.org/ns/1.0
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="/">
        <xml>
            <xsl:apply-templates/>
        </xml>
    </xsl:template>

    <xsl:template match="t:p">
        <new>
            <xsl:apply-templates/>
        </new>
    </xsl:template>

</xsl:stylesheet>
```

XSLT processing paradigms i

All *p*, including their contents are preserved (thanks to *apply-templates*) but instead of *<p>* they are now called *<new>* (just for demonstration purposes).

This is referred to as the **push paradigm**.

You can see the **pull paradigm** where the *<head>* is added. We explicitly get the value of *<title>* to put in this element (careful about finding more titles than you wanted!

value-of select="" ≠ apply-templates

apply-templates will check for further processing rules deeper down the XSL hierarchy to match child elements of the current element.

value-of select="" only copies/prints the current element content (not the node!), further nested elements are lost!

value-of select="." gives you the current node's text content.

attribute={@rend} is a shorthand for attributes.

XSLT processing paradigms ii

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:t="http://www.tei-c.org/ns/1.0"
    exclude-result-prefixes="xs"
    version="2.0">

    <xsl:template match="/">
        <xml>
            <head><xsl:value-of select="//t:title"/></head>
            <xsl:apply-templates/>
        </xml>
    </xsl:template>

    <xsl:template match="t:p">
        <new attribute={@rend}>
            <xsl:apply-templates/>
        </new>
    </xsl:template>

</xsl:stylesheet>
```

Deleting contents

XSLT automatically prints all element values (contents) but we can delete them. That way, we delete all elements, including their contents and child elements (!), from our output.
→ Don't accidentally write/save this on your original source data!
(Careful when setting up the transformation scenario to not name the output the same as the input XML.)

Delete consciously by creating an empty rule

```
<xsl:template match="t:hi">
    <!-- delete -->
</xsl:template>
```

This is mostly all you need to know.

Of course, there are more complicated functions for advanced usage. A few example are shown in the following.

XSLT example: poem to HTML page

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns="http://www.w3.org/1999/xhtml" version="2.0">

    <xsl:template match="/">
        <html>
            <head>
                <title><xsl:value-of select="div/head" /></title>
            </head>
            <body><xsl:apply-templates select="div" /></body>
        </html>
    </xsl:template>

    <xsl:template match="div">
        <h1><xsl:value-of select="head" /></h1>
        <div><xsl:apply-templates select="lg" /></div>
    </xsl:template>

    <xsl:template match="lg">
        <p><xsl:apply-templates /></p>
    </xsl:template>

    <xsl:template match="l">
        <xsl:value-of select="." /><br />
    </xsl:template>
```

More functionalities i

variables

```
<xsl:variable name="substitute" select="content">
```

XSLT handles variable with a so-called *pass by value*, not *by reference*. Ergo only the value is passed that the element has at the moment the variable is created. If I define it too early, I might get the wrong value!

attributes

`<xsl:attribute>` is used to set attributes in the output document. It's not always necessary: Often you can use the shorthand `type={@rend}`.

More functionalities ii

plain text

`<xsl:text>` puts the text as it is, i.e. won't put in unwanted spaces (like XSL tends to do which can be a problem in LaTeX where space is significant. XSL will be very liberal with spacing, unless you explicitly suppress this using `preserve-space`.

sorting

There is `<xsl:sort>` for sorting values.

conditions

`<xsl:choose>` lets you choose a different way of processing according to a test using `<xsl:when>`. `<xsl:if>` only does something when a condition is met (i.e. if an element doesn't exist or has content XY). You should use this to check if optional elements exist to avoid errors!

More functionalities iii

Person list in HTML using `for-each`

```
<ul> <!-- unordered in HTML -->
<xsl:for-each select="t:persName">
    <li> <!-- list element -->
    <xsl:value-of select="." />
    </li>
</xsl:for-each>
</ul>
```

More functionalities iv

Setting attributes

```
<xsl:for-each select="t:persName">
    <span class={@rend}><xsl:value-of select="."></span>
</xsl:for-each>
<!-- OR -->

<span>
    <xsl:attribute name="id">
        <xsl:value-of select="@rend"><xsl:text>-person</xsl:text>
    <xsl:attribute>
    <xsl:attribute name="interpretation">
        <xsl:value-of select="concat(@rend,@ana)">
    <xsl:attribute>
    <xsl:apply-templates /> <!-- for the content -->
<span>
<!-- result e.g.
    <span id="monster-person" interpretation="monster-evil">
    Weird Sisters </span> -->
```

For-each-group

```
<!-- could be sorted alphabetically.
     use sparingly, can produce mysterious errors. -->

<xsl:for-each-group select="bla"
  group-starting-with="dings[@rend="startbla]]"
    <xsl:apply-templates select="current-group()">
    <xsl:value-of select="current()/bla">
</xsl:for-each-group>
```

Merging multiple documents with (document( ))

```
<xsl:apply-templates
  select="document('Letter1_TEI.xml')/tei:TEI//tei:body/tei:div"/>
```

Loops: `for-each`

Example: Get each person (*//persName*) and generate a listing (*<ul>*)of the last names (*lastname*):

```
<ul>
<xsl:for-each select="//persName">
    <xsl:sort select="lastname" order="ascending" />
    <li> <xsl:value-of select="lastname"/> </li>
</xsl:for-each>
</ul>
```

Conditions I: if

xsl:if only runs if condition in *@test* evaluates as 'true':

```
<xsl:if test=" xpath-ausdruck "> ... </xsl:if>

<xsl:for-each select="//book">
    <xsl:if test=" author = 'Cicero' ">
        <li><xsl:value-of select="title"/></li>
    </xsl:if>
</xsl:for-each>
```

Conditions II: choose

You can also differentiate a number of cases:

```
<xsl:choose>
    <xsl:when test="some xpath"> ... </xsl:when>
    <xsl:otherwise> ... </xsl:otherwise>
</xsl:choose>
```

You can also sort (*xsl:sort*), copy (*xsl:copy - xsl:copy-of*) and use variables.

XSL paradigms: push

push paradigm

```
<xsl:apply-templates select="path"/>
```

Suitable for processing the text body (mostly).

Elements are processed whereever they are encountered, you don't need to know their exact structure or order. The input document's structure is preserved (or not if defined otherwise in the template rules). Useful if keeping a similar document structure is the intended goal.

```
<xsl:template match="/">
    <xsl:apply-templates/>
</xsl:template>
```

push processing

Call Template / push method

```
<xsl:template match="/">
    <xsl:call-template name="etc">
</xsl:template>

<xsl:template name="etc">
    ... do sth ...
</xsl:template>
```

XSL paradigms: pull i

pull paradigm

```
<xsl:call-templates name="etc"/>
```

Useful mainly for extracting metadata from the *<teiHeader>* to display them somehow. Also allows you to go over the whole document and create a list of persons mentioned in it, for example at the beginning of the output, using *<xsl:for-each>*.

<xsl:for-each> and *<xsl:value-of select="">* are typical commands in the pull paradigm.

pull processing

Pick some specific nodes with full control. Useful if the output document's structure should differ from the input quite a bit or you just want to selectively keep certain elements (like just make a list of dates mentioned in a document without printing the text of the original document).

```
<xsl:value-of select=''pattern''/>
<xsl:apply-templates select=''pattern''/
<xsl:for-each select=''pattern''/>
```

The paradigms in practice

In practice, you will use a combination of both, maybe even for the same elements:

- **pull**, e.g.
  - get the metadata from the header to create a recommended citation
  - create a table of contents
- **push**, e.g.
  - get (select) and process the document body automatically
  - process chapter headings inside the document body

```
<xsl:apply templates
  select="//t:body" />

<xsl:apply-templates
  select="head" mode="toc"/>
```

`<xsl:template match="">`

process the element where it appears.

`<xsl:value-of select="">`

give me the element content as plain text/string (child elements and structure are lost).

`<xsl:apply-templates>`

traverses the rules and it applies the rules for anything found in a *@match*. You always have to push the *apply-templates*) in each single template rule.

Facts

How to create your stylesheet

top down, starting with the root of the source document. Regelwerk nach Top-down-Prinzip, ausgehend vom Wurzelelement des Quelldokuments.

root element `xsl:stylesheet`

XSLT is XML istself which is why it has an all-encompassing root element where lots of parameters can be declared (namespaces, XSL processor, etc.)

XSLT practice!

Start with setting up your transformation scenario (see slides, cheatsheet and video).

Do the XSLT (prep) exercises on the exercise sheet if you haven't already done so (2.1). Do exercises 2.2 and 2.3.

Set up the transformation to run `mini-bootstrap.xsl` and/or `mini-latex.xsl` (using the `dracula.xml`) (see materials/resources folder)

Try to run it on your own data (won't transfer properly), then adapt for your own data.

XSLT to HTML practice!

First, try to configure a transformation scenario and get this to work at all (ideally on the `dracula.xml` first). Then try the following:

Easy: Write simple template rules for a few elements (where do they belong in the hierarchy?), copy how it's done from the existing rules. Read the info on the worksheet!

Harder: Try to adapt the 'Show persons' toggle and template match for TEI `placeName` to 'Show places'.

If you're feeling adventurous (HTML):

First, try to configure a transformation scenario and get this to work at all (ideally on the `dracula.xml` first). Then try the following:

Have a look at the `mini-bootstrap-popover.xsl`. Try to understand what's going on.

The root template (`match="/"`) builds an HTML document and loads the Bootstrap framework. You can mostly ignore this for now. Start from the bottom where the `t:p` template is.

The template is quite a bit longer than the other mini example but not too complicated. It implements the popover wippet by Roman Bleier which allows users to show expanded/abbreviated versions for `<expan>` and `<abbr>`.

Try to add the infobox wippet (read the comments in the code) or look up other Bootstrap elements.

Can you make the template work on your own data?

XSLT to LaTeX practice!

First, try to configure a transformation scenario and get this to work at all (ideally on the `dracula.xml` first). Then try the following:

Easy: Write simple template rules for a few elements (where do they belong in the hierarchy?), copy how it's done from the existing rules.

Harder: Take a look at the next slide…

If you're feeling adventurous (LaTeX):

First, try to configure a transformation scenario and get this to work at all (ideally on the `dracula.xml` first). Then try the following:

Have a look at the `mini-latex-reledmac.xsl`. Try to understand what's going on & configure a transformation scenario to use the template on your own data.

The template is mostly a simplified version of the TEI Critical Apparatus Toolbox template but still complicated. Some templates are very long – leave them alone!

Remember: This is your first day using XSLT. The template might be too difficult for now. If so, use it for practicing after the workshop.

References

[1] Patrick Sahle. “Digitales Archiv und Digitale Edition. Anmerkungen zur Begriffsklärung”. In: *Literatur und Literaturwissenschaft auf dem Weg zu den neuen Medien*. Ed. by Michael Stolz. Zürich, 2007, pp. 64–84.

[2] Patrick Sahle. “What is a Scholarly Digital Edition?” In: *Digital Scholarly Editing: Theories and Practices*. Ed. by Matthew J. Driscoll and Elena Pierazzo. Cambridge: Open Book Publisher, 2016, pp. 19–39. URL: *https://books.openedition.org/obp/3381*.