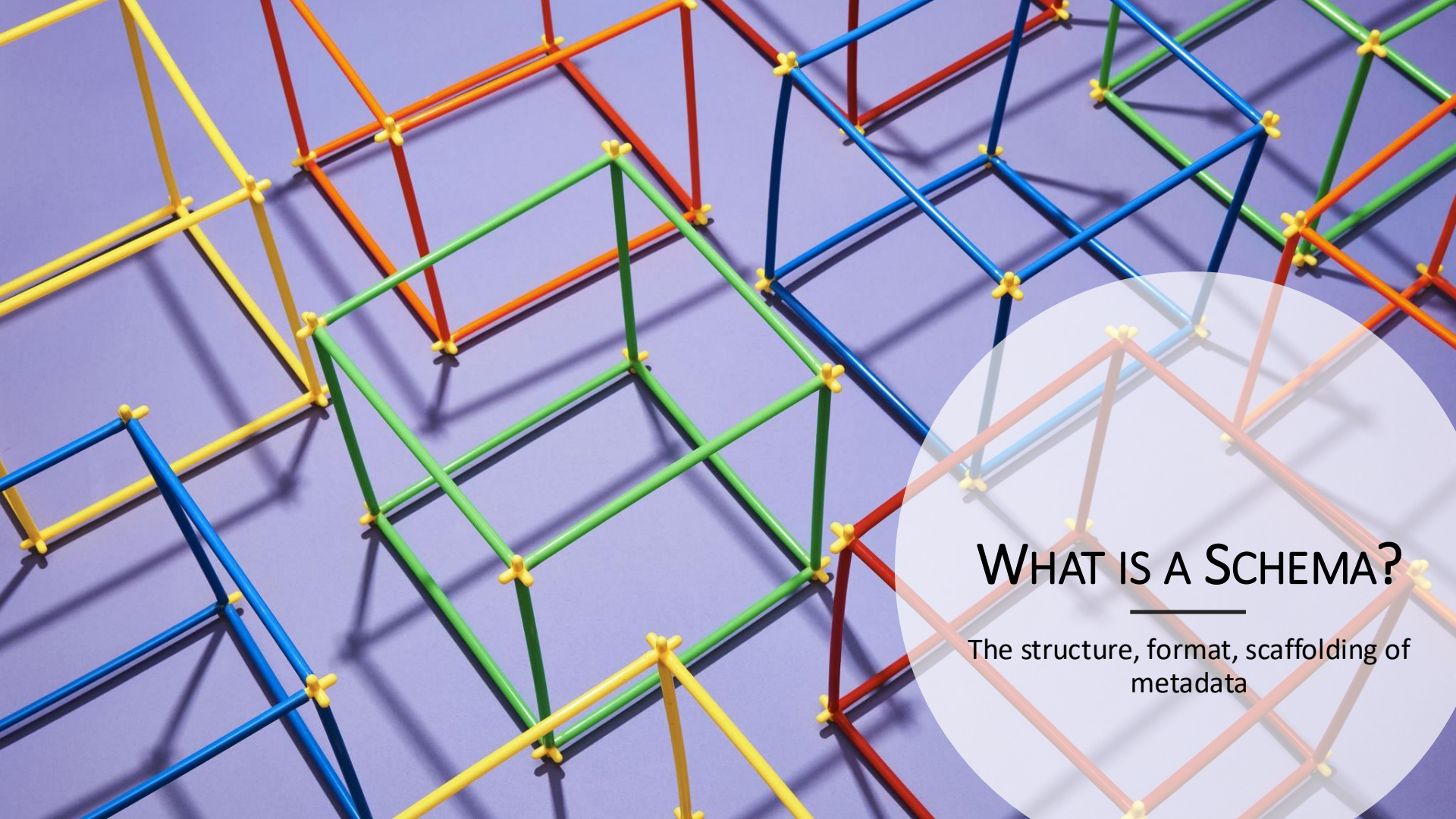
I 202: Information Organization & Retrieval Fall 2025

Class 11: Metadata Descriptor Languages

Today's Outline

- Brief Review
- Markup Languages (HTML, XML)
- Data Formats (JSON)
- Metadata Standards



What is a Schema?

The structure, format, scaffolding of metadata

The Relational Data Model

- A relation is a set of rows (also called tuples)
- Each row consists of a predefined set of attributes
- A database is a collection of relations
- These relations together define the data model

Relation / Table

Attributes / Columns

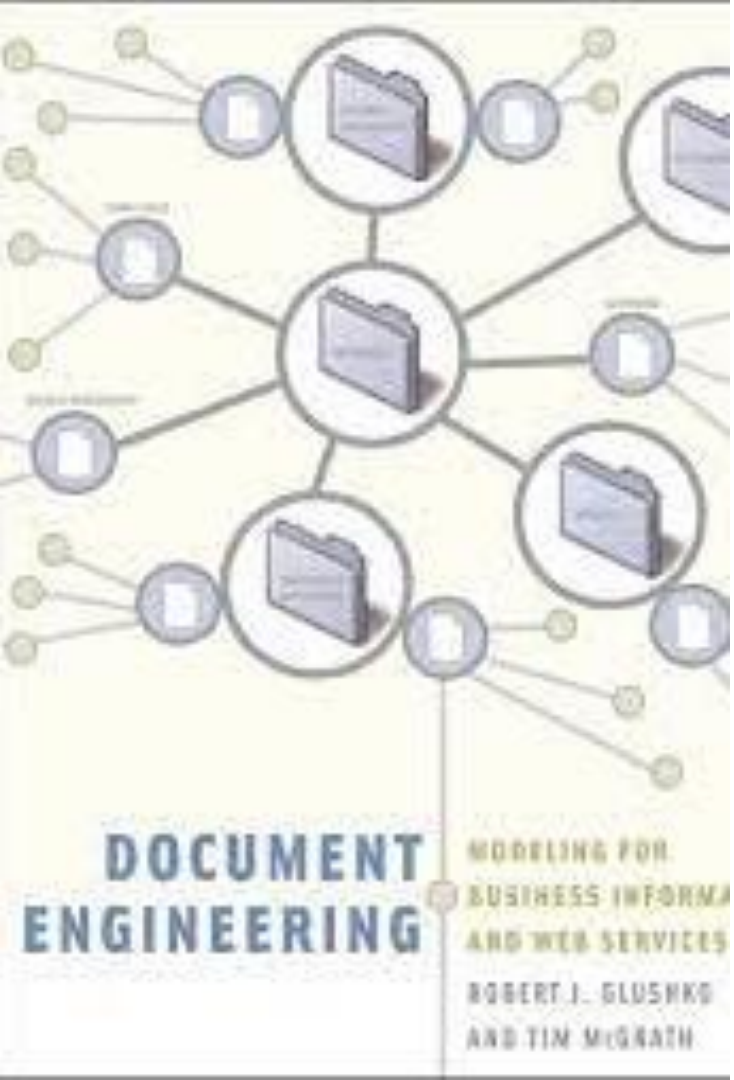
Name	Price	Category	Brand
Climber	$120	Boot	REI
Lita	$98	Flats	West
Arigato	$55	Sneaker	Keds

Rows / Tuples / Records

Slide adapted from Aditya Parameswaran

The Relational Model: What's Missing?

- Not good for semi-structured data
  - *Documents*
  - *Web pages*
  - *Hierarchically structured (nested) information*

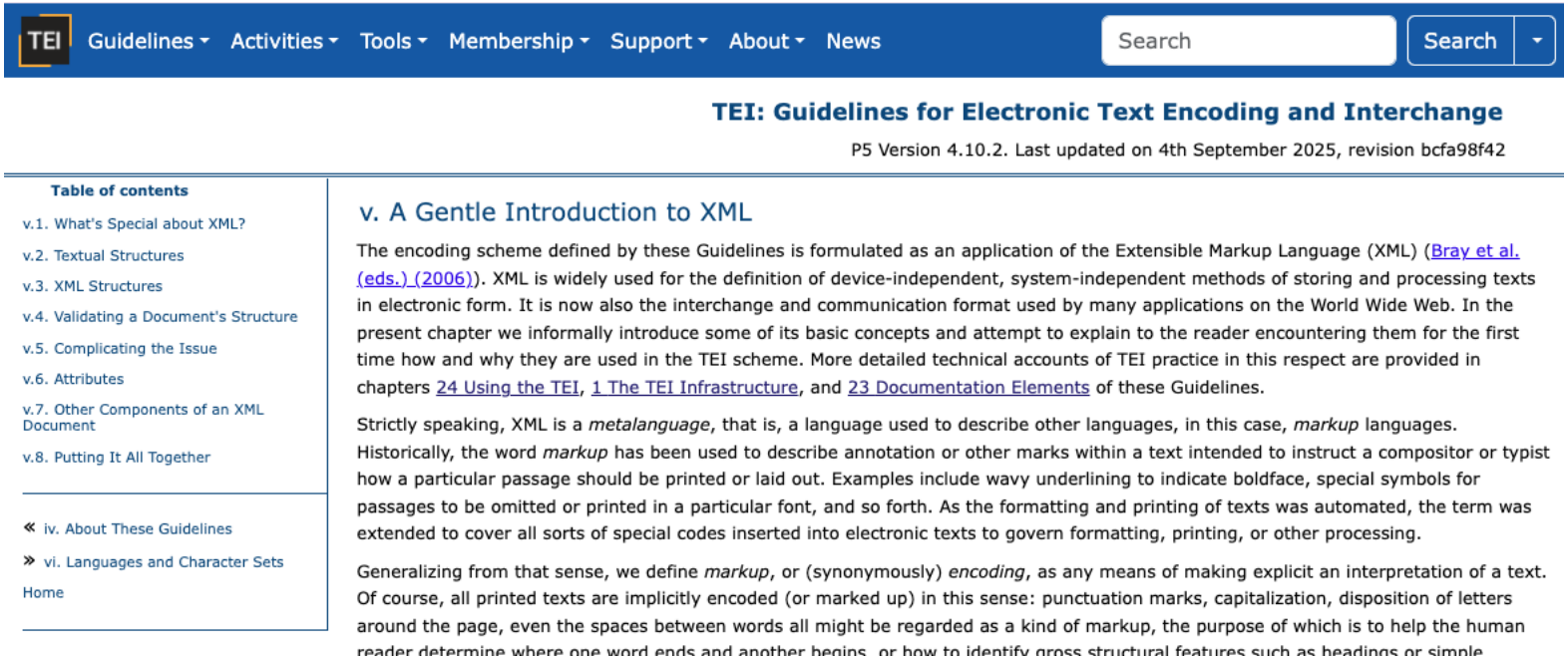


What To Take From This Reading (XML Foundations)

- What HTML and CSS are for – and not for
- HTML vs XML – the difference
  - *Separation of content and presentation*
- Markup Language Syntax basics
- The Main Goals of XML
- The Role of Schemas (like DTDs) in XML
- The ability to make a simple XML DTD

Another good XML Reading

https://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html



What is a Markup Language?

A set of instructions on a manuscript or tags in an electronic document to determine styles of type, makeup of pages, and the like.

https://www.thefreedictionary.com/markup

Content vs. Structure vs. Presentation

Content - "what does it mean"

Structure - "how it is organized or assembled"

Presentation - "how does it look" / "how is it displayed"

HTML & CSS

- HTML represents the **structure** of the document
  - *A small, restricted set of tags*
- CSS represents the **style** (formatting) rules
- **Neither** represent the **meaning** or semantics

HTML From Your Homework Assignment
(index.html)

```
<!DOCTYPE HTML>
<html>
    <header>
        <link rel="stylesheet" href="style.css">
        <link rel="stylesheet" href="table_style.css">
        <title>My Webpage Title</title>
    </header>
    <body>
```

HTML From Your Homework Assignment
(index.html)

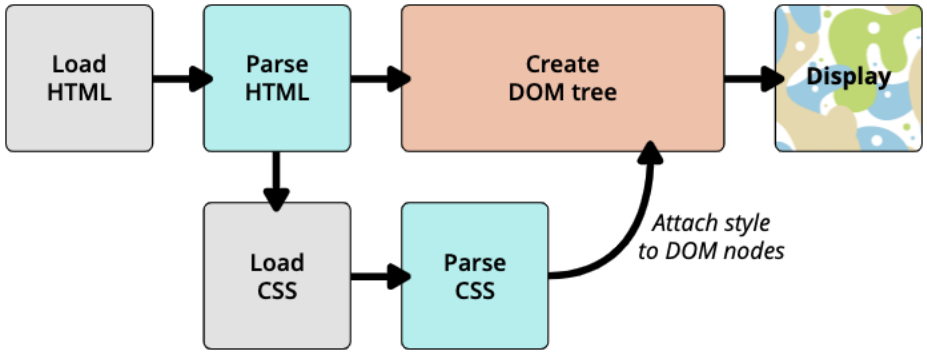
```
<div class="container d-flex align-items-center flex-column">

    <!-- This code is for the student name-->
    <h1 class="masthead-heading text-uppercase mb-0">Test Student</h1>
    <!--This code creates the two lines separating the student name from the subtitle-->
    <div class="divider-custom divider-light">
        <div class="divider-custom-line"></div>
        <div class="divider-custom-line"></div>
    </div>
    <!--This code creates the subheading underneath the line dividers-->
    <p class="masthead-subheading font-weight-light mb-0">School of Information Student</p>
</div>
```

CSS From Your Homework Assignment
(style.css, from Bootstrap)

```css
.divider-custom {
  margin: 1.25rem 0 1.5rem;
  width: 100%;
  display: flex;
  justify-content: center;
  align-items: center;
}
.divider-custom .divider-custom-line {
  width: 100%;
  max-width: 7rem;
  height: 0.25rem;
  background-color: #2c3e50;
  border-radius: 1rem;
  border-color: #2c3e50;
}

 .divider-custom.divider-light .divider-custom-line {
   background-color: #fff;
 }
```

HTML/CSS AND THE BROWSER

- HTML / CSS are visually rendered in a web browser (chrome, firefox, etc).
- The web browser has software to convert the tags into visual display
- The W3C sets the standards for how to render the tags



https://developer.mozilla.org/en-US/docs/Learn/CSS/First_steps/How_CSS_works

TEXT VS HTML

Viewed by human

Airline Schedule

Flight Information

United Airlines #200

- San Francisco
- 9:30AM
- Honolulu
- $368.50

Processed by web browser

```
<html>
<header>
<title>Airline Schedule</title>
</header>
<body>
    <h1>Airline Schedule</h1>
    <h2> Flight Information</h2>
    <h3>United Airlines #200</h3>
    <ul>
    <li>San Francisco</li>
    <li>9:30AM</li>
    <li>Honolulu </li>
    <li>$368.50</li>
    </ul>

</body>
</html>
```

WHAT IS XML?

What is XML?

- XML stands for eXtended Markup Language
  - Descended from SGML (Standard Generalized Markup Language)
  - An international standard (SGML- ISO 8879:1986)
  - Adopted by the W3C in 1998
- A generic language for describing the content and structure of documents, and markup that can be used for those documents. Can be used for data too.
- XML enabled web applications to exchange data in an understandable text format
  - *Transformed the way information is exchanged online*
  - *JSON is perhaps more popular for data (vs documents)*
- What it is NOT:
  - *Not a visual document description*
  - *Not an application specific markup*
  - *Not proprietary*

The Extensible in XML

- **HTML**: a fixed set of format-oriented tags
  - *Meant to be processed "by eye"*
  - *The web browser makes it visual*

- **XML**: **you** decide on the tag set
  - *Meant to be processed by computer*
  - *Supports nesting / hierarchy*

XML vs HTML

```
<TransportSchedule  Type="Airline">
<Segment Id="United Airlines #200">
 <Origin>San Francisco</Origin>
<DepartTime>9:30 AM</DepartTime>
 <Destination>Honolulu</Destination>
<ArriveTime>12:30 PM</ArriveTime>
 <Price Currency="USD">368.50</Price>
</Segment>
</TransportSchedule>
```

```
<h1>Airline Schedule</h1>
<h2> Flight Information</h2>
<h3>United Airlines #200</h3>
<ul>
<li>San Francisco</li>
<li>9:30AM</li>
<li>Honolulu </li>
<li>$368.50</li>
</ul>
```

Notice: content vs presentation

XML can be nested (hierarchical)

```
▼<Class>
  ▼<Order Name="TINAMIFORMES">
    ▼<Family Name="TINAMIDAE">
       <Species Scientific_Name="Tinamus major"> Great Tinamou.</Species>
       <Species Scientific_Name="Nothocercus">Highland Tinamou.</Species>
       <Species Scientific_Name="Crypturellus soui">Little Tinamou.</Species>
       <Species Scientific_Name="Crypturellus cinnamomeus">Thicket Tinamou.</Species>
       <Species Scientific_Name="Crypturellus boucardi">Slaty-breasted Tinamou.</Species>
       <Species Scientific_Name="Crypturellus kerriae">Choco Tinamou.</Species>
     </Family>
   </Order>
  ▼<Order Name="GAVIIFORMES">
    ▼<Family Name="GAVIIDAE">
       <Species Scientific_Name="Gavia stellata">Red-throated Loon.</Species>
       <Species Scientific_Name="Gavia arctica">Arctic Loon.</Species>
       <Species Scientific_Name="Gavia pacifica">Pacific Loon.</Species>
       <Species Scientific_Name="Gavia immer">Common Loon.</Species>
       <Species Scientific_Name="Gavia adamsii">Yellow-billed Loon.</Species>
     </Family>
   </Order>
```

http://webdam.inria.fr/Jorge/prog/birds.xml

XML Schema and Validation

- An XML document *REQUIRES* ONLY the document instance
- But for real world usage, a DTD or Schema (XSD) is important.
  - *DTD is the older, simpler version of the XSD Schema*
  - *DTD: Document Type Definition*
  - *More recently: Relax NG is used for schema definition*
- The XML document is validated against the DTD/Schema
  - *Specialized validation programs are used for this.*
  - *A modern list can be found here: https://relaxng.org/*

Sample XML

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet type="text/xsl" href="calendar.xsl" ?>
<Calendar>
    <Organization>Center for Document Engineering</Organization>
    <TimePeriod>January 2004</TimePeriod>
    <Events>
    <Event type="Lecture">
            <Title>Delivering on the Promise of XML</Title>
            <Description>Eve Maler will introduce the <Keyword>Universal Business
Language (UBL)</Keyword> and the <Keyword>Security Assertion Markup Language
(SAML)</Keyword> and discuss their XML design features that maximize the sharing of
semantics and processing even when the core vocabularies are
customized.</Description>
            <Speaker>
                <Name>Eve Maler</Name>
                <Affiliation>Sun Microsystems</Affiliation>
            </Speaker>
            <DateTime>Monday, January 12 4:00-5:00 PM</DateTime>
            <Location>South Hall 202</Location>
    </Event>
    <Event type="Workshop">
```

From TDO

EXAMPLE DTD

The major components:

Entity Declarations

Element Declarations

Attribute Declarations

#PCDATA: text (character data)

#IMPLIED means optional attribute

(x | y) means OR
+ means non-empty list

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

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

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- DTD for simple calendar -->
<!-- calendar metadata -->

<!ELEMENT Calendar (Organization, TimePeriod, Events)>
<!ELEMENT Organization (#PCDATA)>
<!ELEMENT TimePeriod (#PCDATA)>

<!-- a calendar is a list of events -->
<!ELEMENT Events (Event+)>

<!-- definition of each event, optional Event Type attribute -->
<!ELEMENT Event (Title, Description?, Speaker?, DateTime, Location)>
<!ATTLIST Event
    type (Lecture | Workshop) #IMPLIED>

<!ELEMENT Title (#PCDATA)>

<!-- mixed content definition to allow for keywords in Description -->
<!ELEMENT Description (#PCDATA | Keyword)*>

<!ELEMENT Keyword (#PCDATA)>
<!ELEMENT Speaker (Name, Affiliation)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Affiliation (#PCDATA)>
<!ELEMENT DateTime (#PCDATA)>
<!ELEMENT Location (#PCDATA)>
```

ATTRIBUTES EXAMPLE

There are a variety of special defaults and data types that can be given in attribute definitions

```
<!ATTLIST element-name attribute-name attribute-type attribute-value>
```

<!ATTLIST memo status (PUBLIC | CONFIDENTIAL) PUBLIC>

This sets the **default** status for "memo" to public, but allows the status for "memo" be set to either public or confidential.

Let's Test Our Understanding!

https://pollev.com/i202

METADATA STANDARDS

There are Many Domain-Specific Metadata Standards

- Naming and ID systems (URIs, ISBN)
- Bibliographic / Document description (MARC, RDF, Dublin Core, TEI)
- Music (SMDL)
- Images and objects (CIMI, Getty Art Categories, VRA Core Categories)
- Numeric Data (SDSM, ICPSR)
- Geospatial Data (FGDC, ASTM)

Dublin Core

- Well-known, early standard
- Simple metadata for describing internet resources.
- For “Document-Like Objects”
- 15 Elements.

Dublin Core Elements

- Title
- Creator
- Subject
- Description
- Publisher
- Other Contributors
- Date
- Resource Type
- Format
- Resource Identifier
- Source
- Language
- Relation
- Coverage
- Rights Management

TEI (Text Encoding Initiative) Standard (Used in the Digital Humanities, here Drama)

```
<sp>
  <speaker>COMP</speaker>
  <p>Hello.</p>
</sp>
<stage type="gesture">(user looks around and starts typing)</stage>
<sp>
  <speaker>USER</speaker>
  <p>What is on the table?</p>
</sp>
<sp>
  <speaker>COMP</speaker>
  <p>The table does not exist.</p>
</sp>
<stage type="gesture">(user frowns)</stage>
```

https://teibyexample.org/tutorials/TBED05v00.htm

Current Issue
» 2023: 17.2
Preview Issue
» 2023: 17.3
Previous Issues
» 2023: 17.1
» 2022: 16.4
» 2022: 16.3
» 2022: 16.2
» 2022: 16.1
» 2021: 15.4
» 2021: 15.3
» 2021: 15.2
» 2021: 15.1
» 2020: 14.4
» 2020: 14.3
» 2020: 14.2
» 2020: 14.1
» 2019: 13.4
» 2019: 13.3
» 2019: 13.2
» 2019: 13.1

2022
Volume 16 Number 3

2022 16.3 | XML | PDF | Print

Black Digital Humanities for the Rising Generation

Alanna Prince <prince_dot_a_at_northeastern_dot_edu>, Northeastern University https://orcid.org/0000-0002-3381-8154
Cara Marta Messina <cmessina_at_jsu_dot_edu>, Jacksonville State University https://orcid.org/0000-0001-7848-0815

Abstract

ditions that they
issue is not to
oth article and
aths using their

Formats of Submission and Publication

DHQ is an entirely digital journal, published in XML. We accept submissions in the following formats:

- XML files encoded in the DHQ markup language; a schema, authoring template, and encoding documentation are available.
- XML files encoded in TEI
- RTF, OpenOffice (and its variants), or MS Word

Framing Black Digital Humanities

We conceptualized and proposed this issue shortly after the 2018 *Intentionally Digital, Intentionally Black* conference held by University of Maryland's Center for African American History, Culture, and Digital Humanities (AADHum), co-organized by Catherine Knight Steele and Trevor Muñoz. One of the core challenges expressed by scholars was the limited legibility of their work in DH, a field that is predominantly

Simple TEI Example

```
<anthology>
 <poem>
  <heading>The SICK ROSE</heading>
  <stanza>
   <line>O Rose thou art sick.</line>
   <line>The invisible worm,</line>
   <line>That flies in the night</line>
   <line>In the howling storm:</line>
  </stanza>
  <stanza>
   <line>Has found out thy bed</line>
   <line>Of crimson joy:</line>
   <line>And his dark secret love</line>
   <line>Does thy life destroy.</line>
  </stanza>
 </poem>
<!-- more poems go here -->
</anthology>
```

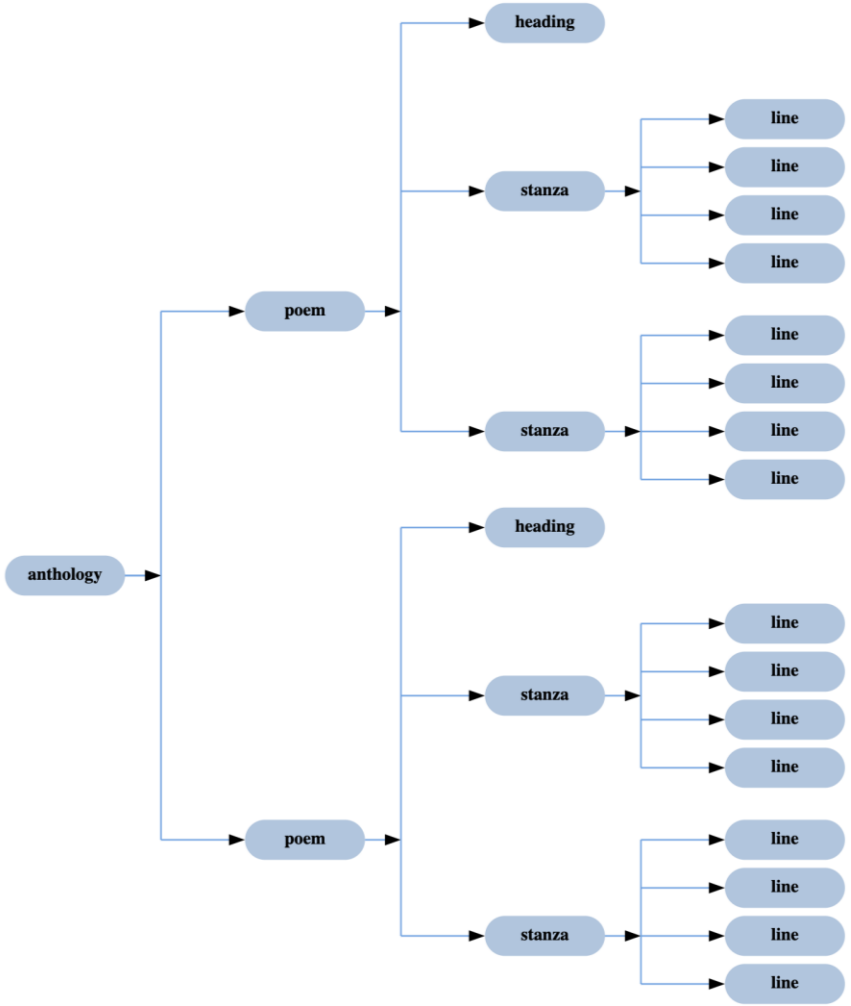
```
<anthology>

<!-- anthology markup elements here -->
 <svg:svg>
<!-- SVG markup elements here -->
 </svg:svg>
 <line>
  <gram:itj>O</gram:itj>
  <gram:nom>Rose</gram:nom>
  <gram:pron>thou</gram:pron>
  <gram:aux>art</gram:aux>
  <gram:adj>sick</gram:adj>
 </line>
</anthology>
```

```
... Rosalind's
 remarks <quote>This is the silliest stuff that ere I heard
 of!</quote> clearly indicate ...
```

https://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html

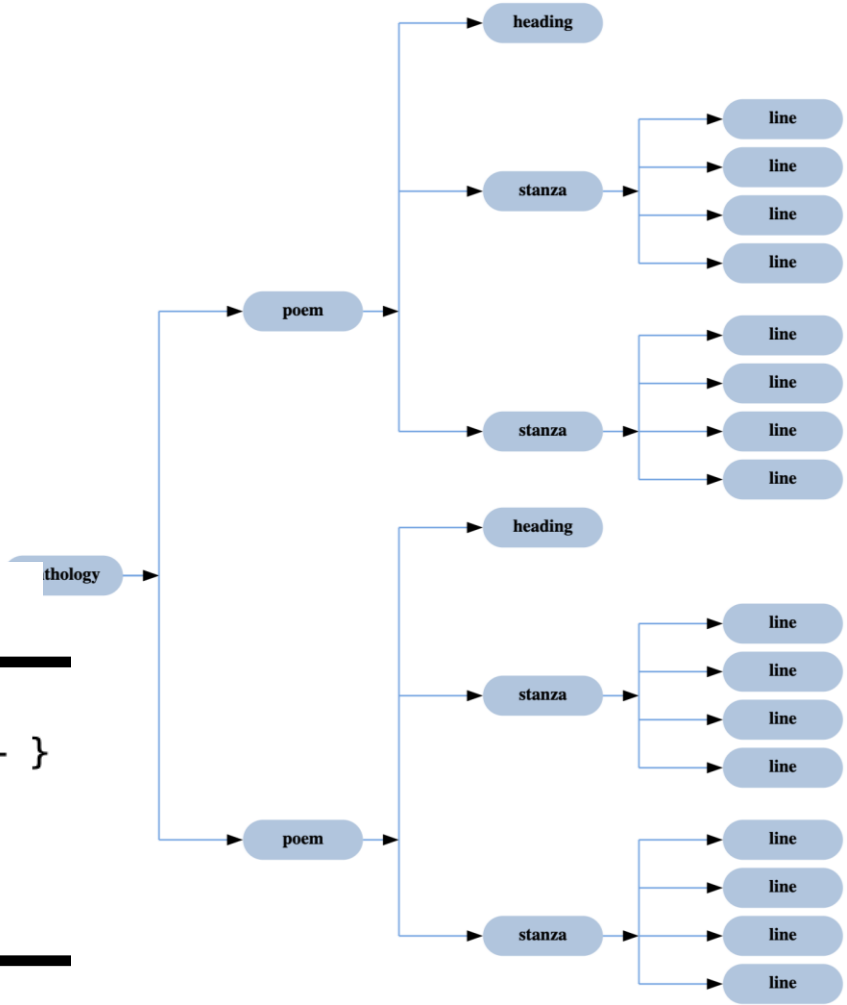
Simple TEI Example

```
<anthology>
 <poem>
  <heading>The SICK ROSE</heading>
  <stanza>
   <line>O Rose thou art sick.</line>
   <line>The invisible worm,</line>
   <line>That flies in the night</line>
   <line>In the howling storm:</line>
  </stanza>
  <stanza>
   <line>Has found out thy bed</line>
   <line>Of crimson joy:</line>
   <line>And his dark secret love</line>
   <line>Does thy life destroy.</line>
  </stanza>
 </poem>
<!-- more poems go here -->
</anthology>
```



https://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html

```
<anthology>
 <poem>
  <heading>The SICK ROSE</heading>
  <stanza>
   <line>O Rose thou art sick.</line>
   <line>The invisible worm,</line>
   <line>That flies in the night</line>
   <line>In the howling storm:</line>
  </stanza>
  <stanza>
   <line>Has found out thy bed</line>
   <line>Of crimson joy:</line>
   <line>And his dark secret love</line>
   <line>Does thy life destroy.</line>
  </stanza>
 </poem>
<!-- more poems go here -->
</anthology>
```



A simple schema written in Relax NG

```
anthology_p = element anthology { poem_p+ }
poem_p = element poem { heading_p?, stanza_p+ }
stanza_p = element stanza {line_p+}
heading_p = element heading { text }
line_p = element line { text }
start = anthology_p
```

https://www.tei-c.org/release/doc/tei-p5-doc/en/html/SG.html

Exercise: What is the Underlying Schema?

- Look at the example of song XML markup:
  - https://gist.github.com/jasonbaldridge/2597611
- Based on this, what do you think the schema is?
- Refer to this poetry example

```
anthology_p = element anthology { poem_p+ }
poem_p = element poem { heading_p?, stanza_p+ }
stanza_p = element stanza {line_p+}
heading_p = element heading { text }
line_p = element line { text }
start = anthology_p
```

```
att.status = attribute status {"draft" | "revised" | "published"}
```

```
poem_p = element poem { att.status?, heading_p?, stanza_p+ }
```

Metadata Query Languages: XQuery

- General purpose query language for XML
  - *W3C standard*
- Derived from earlier ones
  - *XPath*
  - *XML-QL*

```
<html><body>
{
  for $act in doc("hamlet.xml")//ACT
  let $speakers := distinct-values($act//SPEAKER)
  return
    <div>
      <h1>{ string($act/TITLE) }</h1>
      <ul>
      {
        for $speaker in $speakers
        return <li>{ $speaker }</li>
      }
      </ul>
    </div>
}
</body></html>
```

This query asks for a list of the unique speakers in each act of Shakespeare's play Hamlet:

https://en.wikipedia.org/wiki/XQuery

Metadata Query Languages

SPARQL (pronounced "sparkle", a recursive acronym[2] for **SPARQL Protocol and RDF Query Language**) is an RDF query language—that is, a semantic query language for databases—able to retrieve and manipulate data stored in Resource Description Framework (RDF) format.[3][4] It was made a standard by the *RDF Data Access Working Group* (DAWG) of the World Wide Web Consortium, and is recognized as one of the key technologies of the semantic web.[*citation needed*] On

This query returns names and emails of every person in the dataset:

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
SELECT ?name
       ?email
WHERE
  {
    ?person  a          foaf:Person .
    ?person  foaf:name  ?name .
    ?person  foaf:mbox  ?email .
  }
```

https://en.wikipedia.org/wiki/SPARQL

SQL vs XQuery

SQL (Relational Data)

- Flat; rows and columns
- Data is uniform
- Rows are unordered

XQuery (semi-structured)

- Nested
- Data is highly variable
- Elements are ordered

http://wwwlgis.informatik.uni-kl.de/cms/fileadmin/courses/SS2014/Neuere_Entwicklungen/Chapter_8_-_XQuery.pdf

Ontology vs HTML

Ontology

- For representing data
- Specific relation types
- Inference over relations

HTML

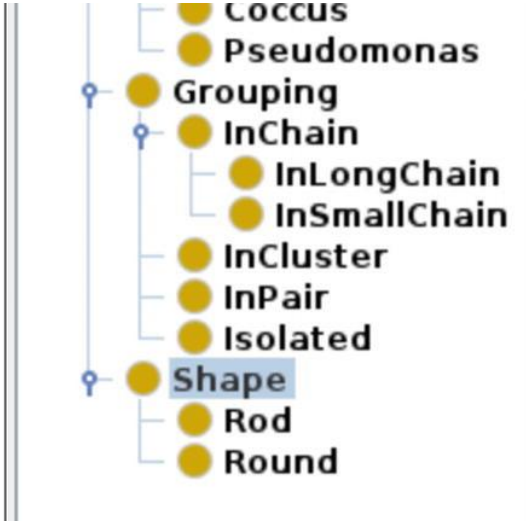
- For presenting info
- Links don't have types
- Links don't have semantics (except "get the linked page")

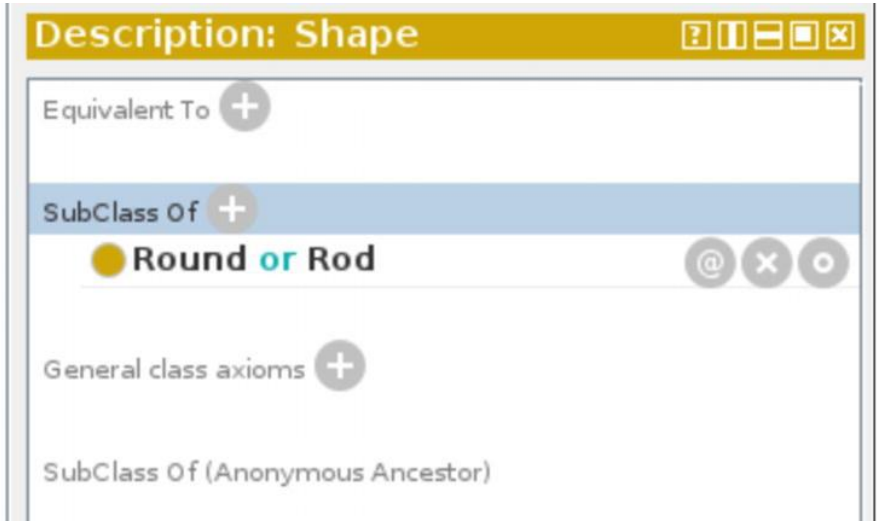
http://wwwlgis.informatik.uni-kl.de/cms/fileadmin/courses/SS2014/Neuere_Entwicklungen/Chapter_8_-_XQuery.pdf

Schema For Ontologies

- There has been a lot of development over decades
- Currently, OWL is the standard
  - *OWL: Web Ontology Language*
- Most commonly stored in RDF / XML format
  - *RDF: Resource Description Framework*
  - *Can use JSON*
- Represents information as Triples (also called Tuples)
- Protégé is a popular, free GUI for ontologies

Protégé Tool for Editing Ontologies





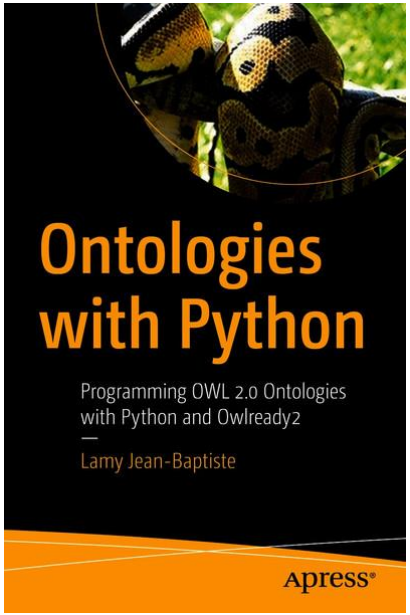
Jean-Baptiste, Ontologies with Python, O'Reilly, 2020

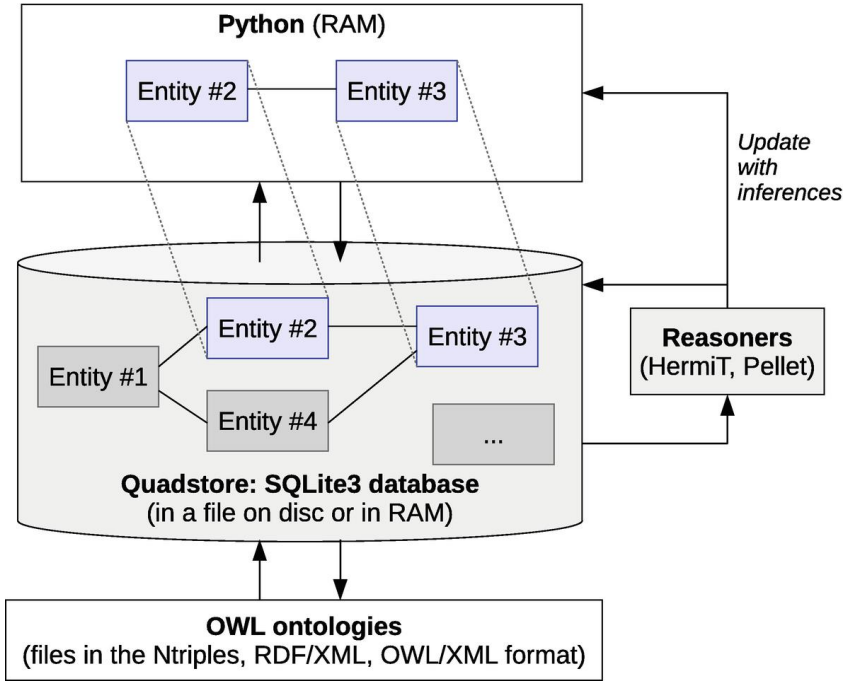
https://protege.stanford.edu/

Storing / Computing / Searching Ontologies

- Owlready
  - *Python interface to OWL*
- Neo4j
  - *Java interface for graphs*
- QuadStore
  - *RDF-backed database for Node.js*
  - *Quad: (subject, predicate, object, graph)*
- SPARQL
  - *Query language for searching in an RDF graph*

Architecture of the Owl-ready Python Interface for Ontologies





Jean-Baptiste, Ontologies with Python, O'Reilly, 2020

Data Formats

JSON

JSON

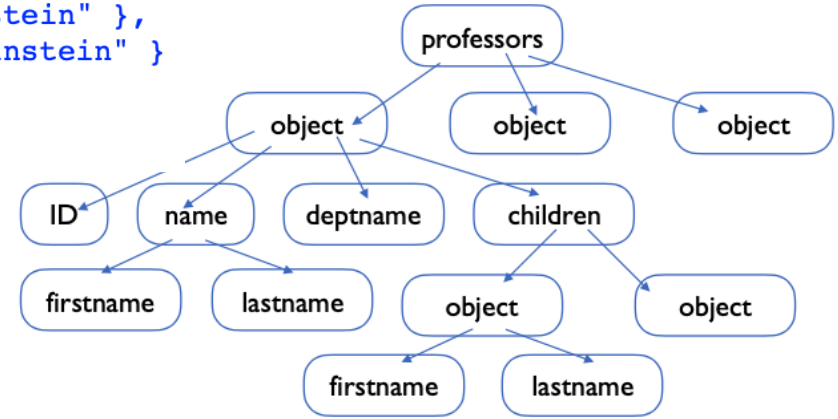
- Stands for Javascript Object Notation
- Textual representation for complex data types
  - *Java stores data as binary, JSON can be either*
- Now is widely used for transmitting data between applications and storing complex data
  - *Especially widely used in internet applications*
  - *Easily manipulated by javascript in the browser*
  - *More structure than a CSV file*

JSON Can Represent Hierarchy

```
{
  "ID": "22222",
  "name": {
  “firstname": "Albert",
  “lastname": "Einstein"
  },
  "deptname": "Physics",
  "children": [
    {"firstname": "Hans", "lastname": "Einstein" },
    {"firstname": "Eduard", "lastname": "Einstein" }
  ]
}
```

JSON Can Represent Hierarchy

```
{
  "ID": "22222",
  "name": {
  “firstname": "Albert",
  “lastname": "Einstein"
  },
  "deptname": "Physics",
  "children": [
   {"firstname": "Hans", "lastname": "Einstein" },
   {"firstname": "Eduard", "lastname": "Einstein" }
  ]
}
```



ADVANTAGES OF JSON

- **Flexibility** in set of attributes
  - Can add a new attribute in one tuple without changing others
- **Human-readable**, key-value pairs
- **Easily parsed** within JavaScript in the browser
- **Smaller** in space required than XML

DISADVANTAGES OF JSON

- JSON encourages **redundancy** in the representation
  - *This makes it take up more space than some other representations, but it is more human readable*
- Not efficient to flexibly search at scale
- Standard JSON does not have a schema.
  - *Can be seen as a pro or a con – makes it flexible but harder to know what you are getting with a*

JSON vs XML

```
{
  "student": [

     {
        "id":"01",
        "name": "Tom",
        "lastname": "Price"
     },

     {
        "id":"02",
        "name": "Nick",
        "lastname": "Thameson"
     }
  ]
}
```

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
        <student>
                <id>01</id>
                <name>Tom</name>
                <lastname>Price</lastname>
        </student>
        <student>
                <id>02</id>
                <name>Nick</name>
                <lastname>Thameson</lastname>
        </student>
</root>
```

https://www.guru99.com/json-vs-xml-difference.html

JSON vs XML

JSON	XML
Data format	Markup language
Smaller space	Larger space
Types: string, integer, array, Boolean	Typeless
Relatively easy to read	Relatively difficult to read
Objects easily accessible as objects by programs	Requires complex parsing
Does not support namespaces, fewer encodings	Supports name spaces, many encodings
No widely adopted schema language	Several schema languages (DTD, XSD, etc)

https://www.guru99.com/json-vs-xml-difference.html

JSON IN YOUR ASSIGNMENT
(THIS IS MY LIST; YOU WILL MAKE YOUR OWN)

List of publications

Each publication has a list of attributes

Notice the ”key: value” notation to separate key-value pairs

list_data.

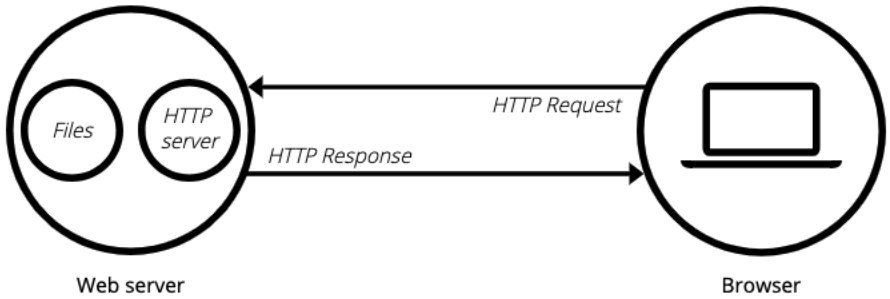
```
{
    "publications": [{
        "Title": "Automatically Generating Cause-and_Effect Questions from Passages",
                                    K., Rathod, M., Tu, T., Xiao, Y, and Hearst, M.A.",
                                    p",

    },
    {
        "Title": "Automatic Feedback Generation for Dialog-Based Language Tutors Using
            Transformer Models and Active Learning",
        "Authors": "Stasaski, K. and Ramanarayanan, V.",
        "Venue": "Human-in-the-Loop Dialogue Systems Workshop",
        "Year": "2020"
    },
    {
        "Title": "More Diverse Dialogue Datasets via Diversity-Informed Data Collection",
        "Authors": "Stasaski, K., Yang, G., and Hearst, M.A.",
        "Venue": "ACL",
        "Year": "2020"
```

JSON IN YOUR ASSIGNMENT

- The file *list.html* contains:
- HTML code that
  - *Creates the header*
  - *Creates the body, including a div called showDataJSON*
- Javascript code that
  - *Reads in the JSON data*
  - *Creates an HTML table*
  - *Fills in the table with the contents of the JSON data*
  - *Places the table at the div called showDataJSON*

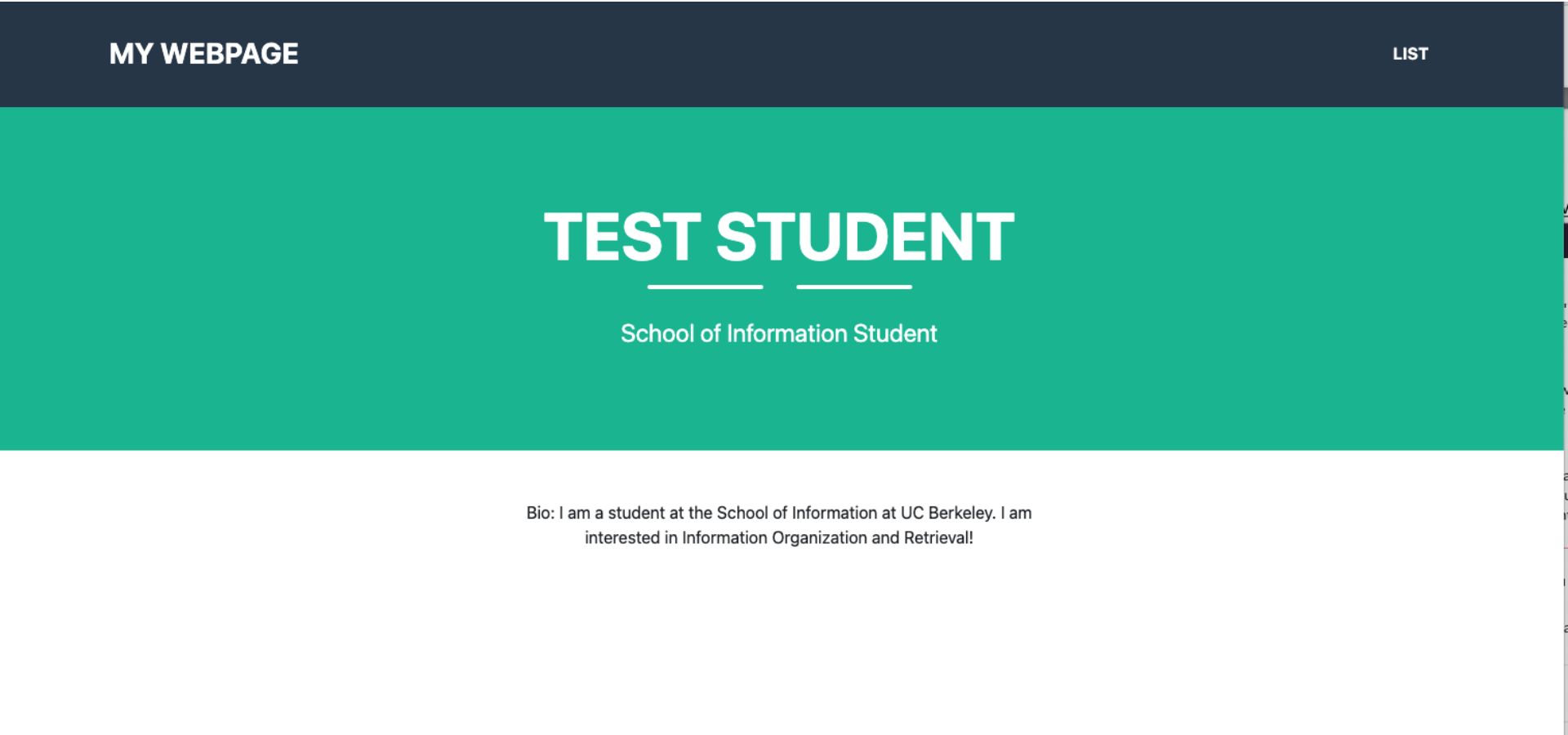
JAVASCRIPT WITH HTML IN YOUR ASSIGNMENT

- Javascript goes beyond the HTML standard
- It requires a web server to run
- In our assignment, we use github pages as our web server



More details:
https://developer.mozilla.org/en-US/docs/Learn/Common_questions/What_is_a_web_server

The Website


MY WEBPAGE
LIST
TEST STUDENT
School of Information Student
Bio: I am a student at the School of Information at UC Berkeley. I am interested in Information Organization and Retrieval!

The Website

MY WEBPAGE LIST

Publications

Title	Authors	Venue	Year
Automatically Generating Cause-and_Effect Questions from Passages	Stasaski, K., Rathod, M., Tu, T., Xiao, Y, and Hearst, M.A.	BEA Workshop	2021
Automatic Feedback Generation for Dialog-Based Language Tutors Using Transformer Models and Active Learning	Stasaski, K. and Ramanarayanan, V.	Human-in-the-Loop Dialogue Systems Workshop	2020
More Diverse Dialogue Datasets via Diversity-Informed Data Collection	Stasaski, K., Yang, G., and Hearst, M.A.	ACL	2020
Construction of a Large Open Access Dialogue Dataset for Tutoring	Stasaski, K., Kao, K., and Hearst, M.A.	BEA Workshop	2020

```
<!DOCTYPE HTML>
<html>
    <header>
        <link rel="stylesheet" href="style.css">
        <link rel="stylesheet" href="table_style.css">
        <title>My Webpage Title</title>
    </header>
    <body>
        <!---Navigation--->
        <!--This section of code defines the navigation bar at the top of the page.
        The classes used here are defined in the style.css file.-->
        <nav class="navbar navbar-expand-lg bg-secondary text-uppercase fixed-top"
        id="mainNav">
            <div class="container">
                <!--This code defines the "My Webpage" link in the top right corner
                of the navigation pane-->
                <a class="navbar-brand" href="">My Webpage</a>
                <!--This code defines the "List" link in the top left corner of the
                navigation pane-->
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0
                    px-lg-3 rounded" href="./list.html">List</a></li>
                </ul>
            </div>
        </nav>
        <!--This code creates the middle section of the webpage-->
        <header class="masthead bg-primary text-white text-center">
            <h1>My Movies</h1>
        </header>
        <!--Here is where the table will be loaded in.  The actual table will be
        loaded in JavaScript.-->
        <section class="mt-5 text-center">
            <div id="showDataJSON"></div>
        </section>
    </body>
```

Page header and nav bar

```
<!DOCTYPE HTML>
<html>
    <header>
        <link rel="stylesheet" href="style.css">
        <link rel="stylesheet" href="table_style.css">
        <title>My Webpage Title</title>
    </header>
    <body>
        <!---Navigation--->
        <!--This section of code defines the navigation bar at the top of the page.
        The classes used here are defined in the style.css file.-->
        <nav class="navbar navbar-expand-lg bg-secondary text-uppercase fixed-top"
        id="mainNav">
            <div class="container">
                <!--This code defines the "My Webpage" link in the top right corner
                of the navigation pane-->
                <a class="navbar-brand" href="">My Webpage</a>
                <!--This code defines the "List" link in the top left corner of the
                navigation pane-->
                <ul class="navbar-nav ms-auto">
                    <li class="nav-item mx-0 mx-lg-1"><a class="nav-link py-3 px-0
                    px-lg-3 rounded" href="./list.html">List</a></li>
                </ul>
            </div>
        </nav>
```

Section for table
Table will be filled in with json data

```
<!--This code creates the middle section of the webpage-->
<header class="masthead bg-primary text-white text-center">
    <h1>My Movies</h1>
</header>
<!--Here is where the table will be loaded in.  The actual table will be
loaded in JavaScript.-->
<section class="mt-5 text-center">
    <div id="showDataJSON"></div>
</section>
```

```
<script>
    /*
    * RENDERING THE JSON TABLE
    * Get JSON data from file
    * Get headers from retrieved data
    * Set headers for table
    * Add data to table
    * Add table to DOM
    */
    fetch('./list_data.json')
        .then(response => response.json())
        .then(data => {

            /*
            This section creates the base table and finds the root json node.
            */
            var jsonTable = document.createElement("table");
            var tr = jsonTable.insertRow(-1);
            jsonTable.style.width = "75%";
            let root;
            for (let prop in data) {
                root = prop;
            }

            /*
            This section selects the headers for the table
            */
            let headers = Object.keys(data[root][0]);
            headers.forEach(header => {
                var th = document.createElement("th");
                th.innerHTML = header;
                tr.appendChild(th);
            })
            /*
            This section adds the data into each row of the list
            */
            let items = Object.keys(data[root]);
            items.forEach(item => {
                tr = jsonTable.insertRow(-1);
                for (let key in data[root][item]) {
                   var tabCell = tr.insertCell(-1);
                   tabCell.innerHTML = data[root][item][key];
                }
            })
            /*
            This section adds the table to the HTML
            */
            var divContainer = document.getElementById("showDataJSON");
            divContainer.innerHTML = "";
            divContainer.appendChild(jsonTable);
        });
</script>
```

```
<script>
    /*
    * RENDERING THE JSON TABLE
    * Get JSON data from file
    * Get headers from retrieved data
    * Set headers for table
    * Add data to table
    * Add table to DOM
    */
    fetch('./list_data.json')
        .then(response => response.json())
        .then(data => {

            /*
            This section creates the base table and finds the root json node.
            */
            var jsonTable = document.createElement("table");
            var tr = jsonTable.insertRow(-1);
            jsonTable.style.width = "75%";
            let root;
            for (let prop in data) {
                root = prop;
            }
```

Javascript code
Reads in the json
Creates the table object
Sets the table style

```
        /*
        This section selects the headers for the table
        */
        let headers = Object.keys(data[root][0]);
        headers.forEach(header => {
            var th = document.createElement("th");
            th.innerHTML = header;
            tr.appendChild(th);
        })
        /*
        This section adds the data into each row of the list
        */
        let items = Object.keys(data[root]);
        items.forEach(item => {
            tr = jsonTable.insertRow(-1);
            for (let key in data[root][item]) {
               var tabCell = tr.insertCell(-1);
               tabCell.innerHTML = data[root][item][key];
            }
        })
        /*
        This section adds the table to the HTML
        */
        var divContainer = document.getElementById("showDataJSON");
        divContainer.innerHTML = "";
        divContainer.appendChild(jsonTable);
    });
</script>
```

Creates objects for the table headers

Creates objects for the table rows

Inserts the table objects into the HTML

SUMMARY

- **Metadata** is data about data
- **Schemas** are the structure of metadata for a collection
- **Relational databases** are a particular type of data organization that allow fast, reliable processing
- **Markup languages** provide a standardized, computerizable representation of metadata and data, representing structure, presentation, and/or semantics.

Next week's focus

Data / Information Collections

Categories
- Types of categories
- Cognitive / language aspects
  - **Naming / Lexical similarity**
- Structure
  - Hierarchical / Taxonomy
  - Faceted
  - Overlapping / Clustering
  - Network / Ontology
- Use in Navigation & Search
  - Information Architecture
  - Faceted Navigation

Technology Support for Info Org
- Identifiers
- Metadata
- Markup
- Schema / Databases
- Search Ranking / Evaluation
- Automated category creation
- **Automated similarity**

Social / Ethical Aspects
- Cultural Bias
- Intellectual Property
- Standards Process