Text Encoding and Semantic Representation

XSLT - XML to HTML

marilena.daquino2@unibo.it | https://github.com/marilenadaquino/tesr_dhdk

Outline

XSLT

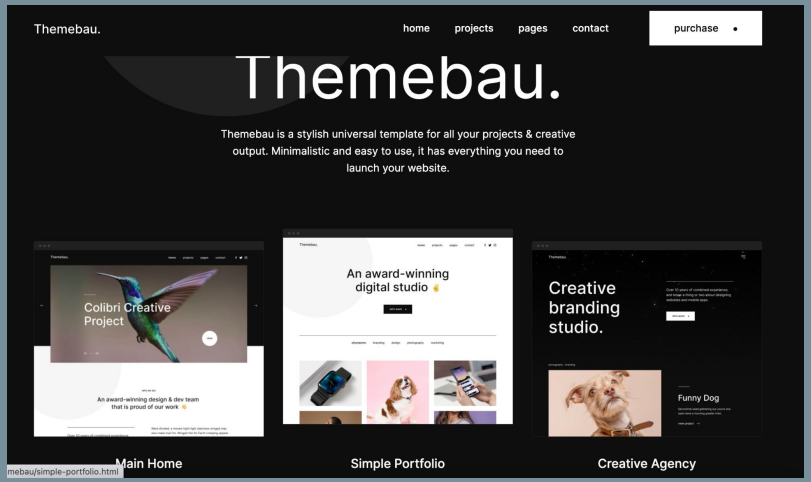
XML/TEI to HTML

Homework, all good?

Create an HTML page

In your editor, create an HTML document that reproduces the content of the webpage (only elements shown in the screenshot aside).

N.B. You cannot style your page if you do not have experience with CSS. For the sake of the exercise, we only focus on the identification of logical elements.



XML to HTML

Why?

The purpose of XML is to provide means to **describe and exchange metadata** according to a standard language (technological interoperability). It does not address **presentational aspects**, specifically, XML files are not meant to be visualised on a browser, which is instead the objective of HTML.

When recording information in XML/TEI we also addressed some presentational aspects of the original source of the edition. Such aspects, as well as others that are not explicitly encoded in XML/TEI (e.g. color and font size of the text), could be rendered on a web page, to facilitate the **dissemination of digital edition** with other scholars and lay people.

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XML to HTML

Transformation

To make a XML document browsable on the web it has to be **transformed** in a HTML document.

Notice that while you *should* have only one XML file (or collection of files) to represent the content of a digital edition, you *may* have as **many HTML** of the same content as you like, each showing different presentational aspects (e.g. a version for your personal website, another for the publisher).

To this extent, there are no expectations on the structure and appearance of the final HTML, which only depend on the developer that performs the transformation.
In other terms **there is not just one way to transform XML into HTML**.

XSLT

Transformation

eXtensible Stylesheet Language Transformations (XSLT) is the declarative language to convert XML documents into other XML documents, HTML and other formats (also PDF!)

A stylesheet is a file that includes the rules to identify XML nodes in a source document (using **XPath**), specifies how to manipulate them (e.g. find all <persName> elements in a text and add them to a list), and store them in a **new file**.

To function XSL needs a **processor**, which is included in browsers (e.g. Gecko in Mozilla Firefox), provided by programming languages libraries (e.g. lxml python library) or embedded in editors (e.g. Saxon in Oxygen editor).

XSLT

Example

Given a list of albums (<cd>) in a XML file, I want to create a web page which organises them in a **table**.

Each row of the table describes one album, and every column represents a piece of information about it (title, artist, label…).

```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Resistance</title>
    <artist>Muse</artist>
    <label>Warner Records</label>
    <country>UK</country>
    <year>2009</year>
  </cd>
  <cd>
    <title>Master of Puppets</title>
    <artist>Metallica</artist>
    <label>Elektra Records</label>
    <country>Denmark</country>
    <year>1986</year>
  </cd>
  <cd>
    <title>A night at the Opera</title>
    <artist>Queen</artist>
    <label>EMI</label>
    <country>UK</country>
    <year>1975</year>
  </cd>
</catalog>
```

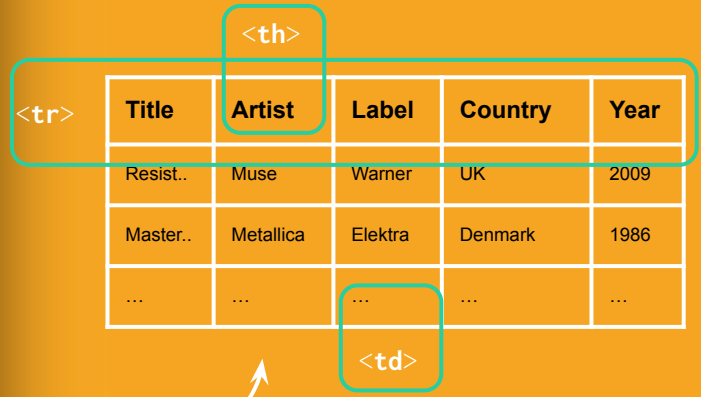
XSLT

Example

In HTML, the element <**table**> is used to visualize a table.

Every row of the table is included in the element <**tr**>. Content of cells are represented by <**td**>. <**th**> is used for the table headers.

```
1  <table>
2    <!-- table header-->
3    <tr>
4      <th>Title</th>
5      <th>Artist</th>
6      <th>Label</th>
7      <th>Country</th>
8      <th>Year</th>
9    </tr>
10   <!-- row 1 -->
11   <tr>
12     <td>Resistance</td>
13     <td>Muse</td>
14     <td>Warner records</td>
15     <td>UK</td>
16     <td>2009</td>
17   </tr>
18   <!-- row 2 -->
19   <tr>
20     <td>Master of puppets</td>
21     <td>Metallica</td>
22     <td>Elektra Records</td>
23     <td>Denmark</td>
24     <td>1986</td>
25   </tr>
26   <!-- row n -->
27     ...
28 </table>
29
```

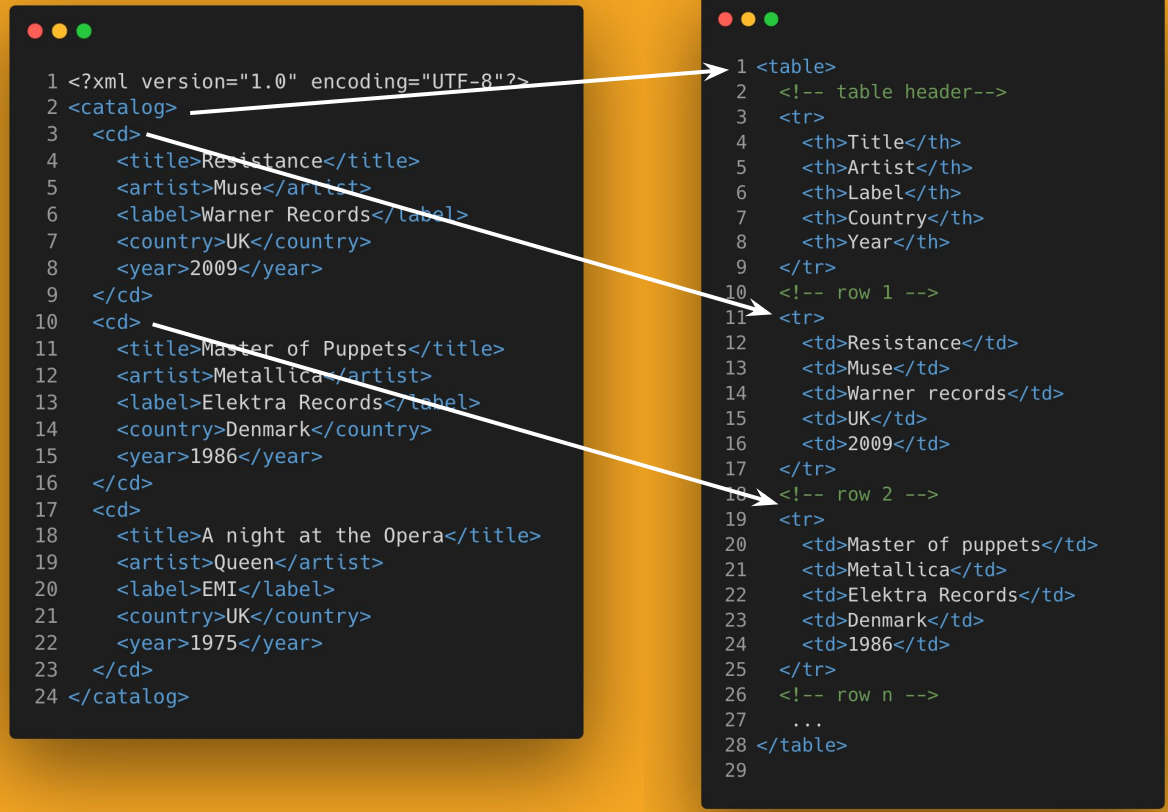


XSLT

Example

We define a preliminarily mapping between XML elements and HTML elements.

`catalog > table` *the container*
N.A. > **`tr[1]`** *the header*
`cd[1] > tr[2]` *the first row*
`cd[2] > tr[3]` *the second row*
`cd[3] > tr[4]` *the third row*



```xml
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Resistance</title>
    <artist>Muse</artist>
    <label>Warner Records</label>
    <country>UK</country>
    <year>2009</year>
  </cd>
  <cd>
    <title>Master of Puppets</title>
    <artist>Metallica</artist>
    <label>Elektra Records</label>
    <country>Denmark</country>
    <year>1986</year>
  </cd>
  <cd>
    <title>A night at the Opera</title>
    <artist>Queen</artist>
    <label>EMI</label>
    <country>UK</country>
    <year>1975</year>
  </cd>
</catalog>
```

```xml
<table>
  <!-- table header-->
  <tr>
    <th>Title</th>
    <th>Artist</th>
    <th>Label</th>
    <th>Country</th>
    <th>Year</th>
  </tr>
  <!-- row 1 -->
  <tr>
    <td>Resistance</td>
    <td>Muse</td>
    <td>Warner records</td>
    <td>UK</td>
    <td>2009</td>
  </tr>
  <!-- row 2 -->
  <tr>
    <td>Master of puppets</td>
    <td>Metallica</td>
    <td>Elektra Records</td>
    <td>Denmark</td>
    <td>1986</td>
  </tr>
  <!-- row n -->
   ...
</table>
```

XSLT

Example

An XSLT file to transform the prior XML file into HTML would look like this.

Notice that a XSLT file is **yet another XML file**. It has a prolog, a root element, and a namespace **xmlns:xsl.**
In this example, the elements that do not use any namespace are HTML elements.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
      version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>My music</title>
      </head>
      <body>
        <h2>My music collection</h2>
        <table>
          <tr>
            <th>Title</th>
            <th>Artist</th>
            <th>Label</th>
            <th>Country</th>
            <th>Year</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
              <td><xsl:value-of select="label"/></td>
              <td><xsl:value-of select="country"/></td>
              <td><xsl:value-of select="year"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XSLT

Templates

The idea behind XSLT is to **match XML elements with templates**, i.e. every time a certain element is found in the source, a certain behaviour/transformation is triggered.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
      version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>My music</title>
      </head>
      <body>
        <h2>My music collection</h2>
        <table>
          <tr>
            <th>Title</th>
            <th>Artist</th>
            <th>Label</th>
            <th>Country</th>
            <th>Year</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
              <td><xsl:value-of select="label"/></td>
              <td><xsl:value-of select="country"/></td>
              <td><xsl:value-of select="year"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XSLT

@match

The element **xsl:template** always comes with the attribute **@match**, including an XPath that matches one or more nodes in the source XML document.

“/” is an XPath that matches the whole XML document. Therefore the following instructions apply to the whole XML document.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
      version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>My music</title>
      </head>
      <body>
        <h2>My music collection</h2>
        <table>
          <tr>
            <th>Title</th>
            <th>Artist</th>
            <th>Label</th>
            <th>Country</th>
            <th>Year</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
              <td><xsl:value-of select="label"/></td>
              <td><xsl:value-of select="country"/></td>
              <td><xsl:value-of select="year"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Read: “When you match a document do…”

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XSLT

HTML

In detail:

when the XML document is matched (“/”) a full page HTML is created.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet
3       version="1.0"
4       xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
5   <xsl:template match="/">
6     <html>
7       <head>
8         <title>My music</title>
9       </head>
10      <body>
11        <h2>My music collection</h2>
12        <table>
13          <tr>
14            <th>Title</th>
15            <th>Artist</th>
16            <th>Label</th>
17            <th>Country</th>
18            <th>Year</th>
19          </tr>
20          <xsl:for-each select="catalog/cd">
21            <tr>
22              <td><xsl:value-of select="title"/></td>
23              <td><xsl:value-of select="artist"/></td>
24              <td><xsl:value-of select="label"/></td>
25              <td><xsl:value-of select="country"/></td>
26              <td><xsl:value-of select="year"/></td>
27            </tr>
28          </xsl:for-each>
29        </table>
30      </body>
31    </html>
32  </xsl:template>
33 </xsl:stylesheet>
```

Create this HTML structure in a new document

XInclude
XPointer
XLink
XPath
XQuery
XSLT

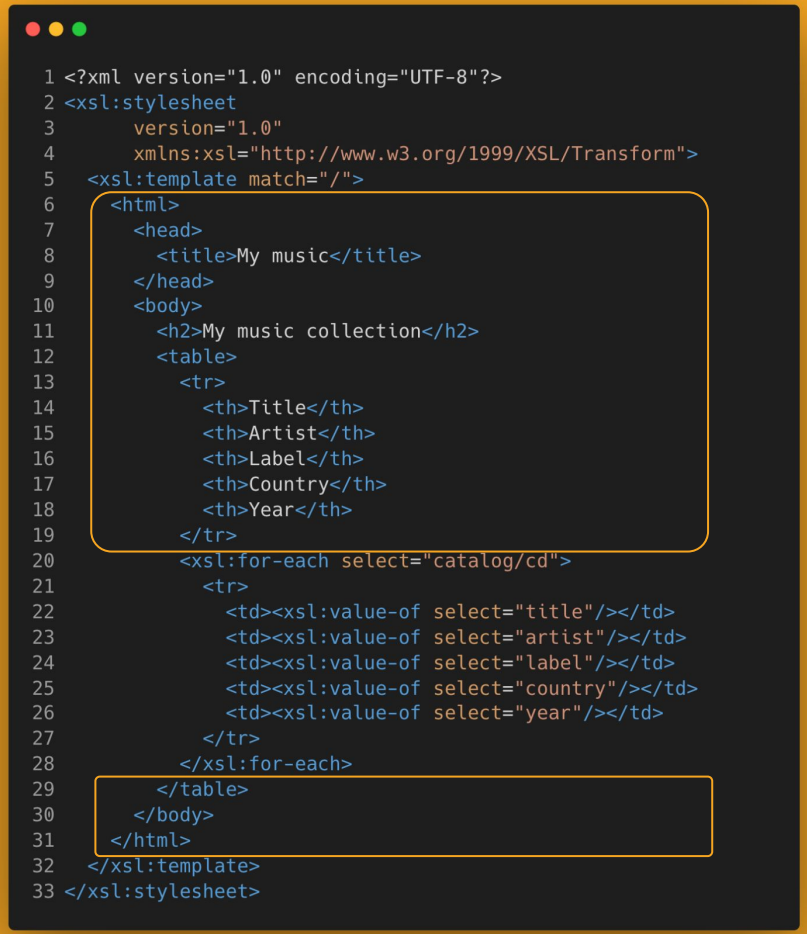
XSLT

HTML

The root <**html**> element is created, along with the <**head**> and <**body**> elements.

In <body> we decide to create a title <**h2**> and we create the container element of the table <**table**>.

In <table> we create the header, that is, the first <**tr**> which includes as many <**th**> elements as the columns we want.



```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet
3       version="1.0"
4       xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
5   <xsl:template match="/">
6     <html>
7       <head>
8         <title>My music</title>
9       </head>
10      <body>
11        <h2>My music collection</h2>
12        <table>
13          <tr>
14            <th>Title</th>
15            <th>Artist</th>
16            <th>Label</th>
17            <th>Country</th>
18            <th>Year</th>
19          </tr>
20          <xsl:for-each select="catalog/cd">
21            <tr>
22              <td><xsl:value-of select="title"/></td>
23              <td><xsl:value-of select="artist"/></td>
24              <td><xsl:value-of select="label"/></td>
25              <td><xsl:value-of select="country"/></td>
26              <td><xsl:value-of select="year"/></td>
27            </tr>
28          </xsl:for-each>
29        </table>
30      </body>
31    </html>
32  </xsl:template>
33 </xsl:stylesheet>
```

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XSLT

For loop

To dynamically populate the table with as many rows as the number of albums, we use a **for loop**, which in XSL is called **xsl:for-each**.

The for loop prevents us to write instructions for each and every <cd> elements, and allows us to establish a single rule to be applied every time an element <cd> is found.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet
3        version="1.0"
4        xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
5    <xsl:template match="/">
6      <html>
7        <head>
8          <title>My music</title>
9        </head>
10       <body>
11         <h2>My music collection</h2>
12         <table>
13           <tr>
14             <th>Title</th>
15             <th>Artist</th>
16             <th>Label</th>
17             <th>Country</th>
18             <th>Year</th>
19           </tr>
20           <xsl:for-each select="catalog/cd">
21             <tr>
22               <td><xsl:value-of select="title"/></td>
23               <td><xsl:value-of select="artist"/></td>
24               <td><xsl:value-of select="label"/></td>
25               <td><xsl:value-of select="country"/></td>
26               <td><xsl:value-of select="year"/></td>
27             </tr>
28           </xsl:for-each>
29         </table>
30       </body>
31     </html>
32   </xsl:template>
33 </xsl:stylesheet>
```

When you match the element cd, if child of catalog…

XSLT

For loop

Notice the Xpath “**catalog/cd**”

This means the document “**/**” has already been matched, and the root element <catalog> is the first to be accessed by the parser, followed by its children <cd>.

From this moment onwards, <cd> is the context node.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
      version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>My music</title>
      </head>
      <body>
        <h2>My music collection</h2>
        <table>
          <tr>
            <th>Title</th>
            <th>Artist</th>
            <th>Label</th>
            <th>Country</th>
            <th>Year</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
              <td><xsl:value-of select="label"/></td>
              <td><xsl:value-of select="country"/></td>
              <td><xsl:value-of select="year"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XSLT

For loop

In detail:

Whenever it matches the element <cd> - N.B. if this is a direct child of the element <catalog> - create an element <tr>. Also, create 5 <td> children elements, which correspond to the number of cells in the row.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
      version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>My music</title>
      </head>
      <body>
        <h2>My music collection</h2>
        <table>
          <tr>
            <th>Title</th>
            <th>Artist</th>
            <th>Label</th>
            <th>Country</th>
            <th>Year</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
              <td><xsl:value-of select="label"/></td>
              <td><xsl:value-of select="country"/></td>
              <td><xsl:value-of select="year"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Create a <tr> element and 5 <td> children

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XSLT

Value of

In each <td> element we include **the value of a XML element**, e.g. <title>, <artist>, etc. that are children of <cd>, using the construct **xsl:value-of**.

Notice that the XPath. Again, it does not start with an absolute path “/”, but with a relative one, since the context node is the one matched in the for loop.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
      version="1.0"
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>My music</title>
      </head>
      <body>
        <h2>My music collection</h2>
        <table>
          <tr>
            <th>Title</th>
            <th>Artist</th>
            <th>Label</th>
            <th>Country</th>
            <th>Year</th>
          </tr>
          <xsl:for-each select="catalog/cd">
            <tr>
              <td><xsl:value-of select="title"/></td>
              <td><xsl:value-of select="artist"/></td>
              <td><xsl:value-of select="label"/></td>
              <td><xsl:value-of select="country"/></td>
              <td><xsl:value-of select="year"/></td>
            </tr>
          </xsl:for-each>
        </table>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

The value of <td> includes the value of the a child XML element (e.g. <title>)

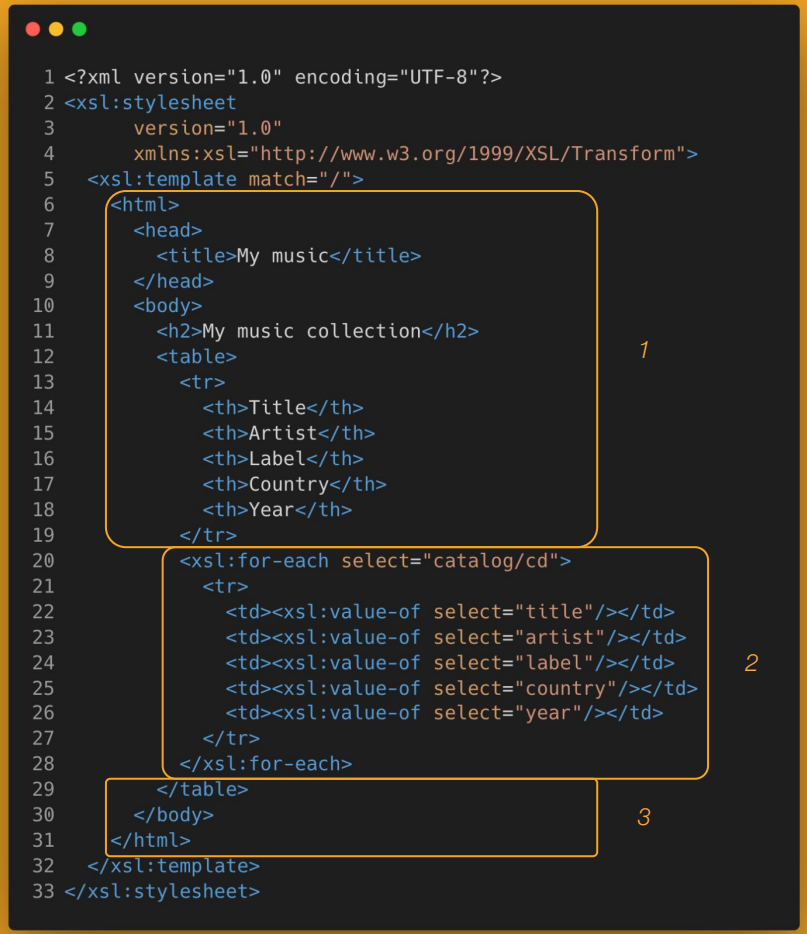
XInclude
XPointer
XLink
XPath
XQuery
XSLT

XSLT

Declarative

So in the end, XSLT allows you to **declare what should appear** in the final HTML.

Sometimes the declaration does not depend on any matched pattern (see blocks 1 and 3), while in other cases a pattern (block 2) needs to be matched in order to trigger a template rule.



XInclude
XPointer
XLink
XPath
XQuery
XSLT

XSLT

Try it out

This is the result.

You can try it out here using the XML example and XSLT example that you can find on the course repo /exercises/

Copy & paste XML and XSLT, save the resulting HTML and open it in a browser.

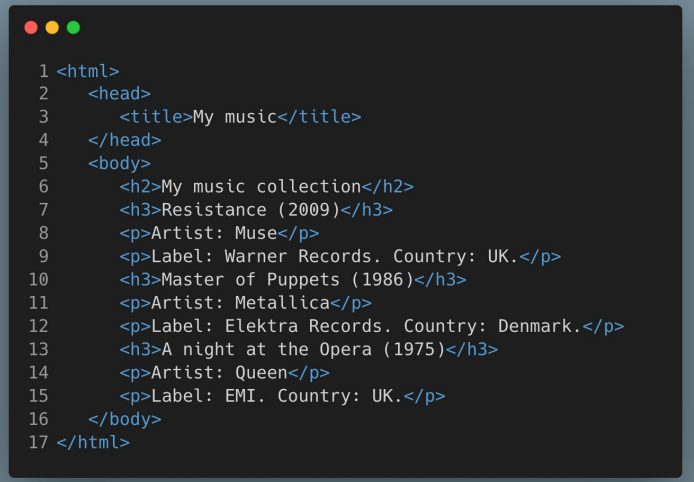
```
<html>
   <head>
      <title>My music</title>
   </head>
   <body>
      <h2>My music collection</h2>
      <table>
         <tr>
            <th>Title</th>
            <th>Artist</th>
            <th>Label</th>
            <th>Country</th>
            <th>Year</th>
         </tr>
         <tr>
            <td>Resistance</td>
            <td>Muse</td>
            <td>Warner Records</td>
            <td>UK</td>
            <td>2009</td>
         </tr>
         <tr>
            <td>Master of Puppets</td>
            <td>Metallica</td>
            <td>Elektra Records</td>
            <td>Denmark</td>
            <td>1986</td>
         </tr>
         <tr>
            <td>A night at the Opera</td>
            <td>Queen</td>
            <td>EMI</td>
            <td>UK</td>
            <td>1975</td>
         </tr>
      </table>
   </body>
</html>
```

Exercise

Practice with XSLT

- Copy & paste the XML document on the tool
- Write a XSL file that transforms the XML file into a HTML file like the following

Notice that multiple XML elements are merged in one HTML element (e.g. the elements <h3> and <p>). It is possible to concatenate multiple <xsl:value-of/> and text in the same HTML element.



```
<html>
   <head>
      <title>My music</title>
   </head>
   <body>
      <h2>My music collection</h2>
      <h3>Resistance (2009)</h3>
      <p>Artist: Muse</p>
      <p>Label: Warner Records. Country: UK.</p>
      <h3>Master of Puppets (1986)</h3>
      <p>Artist: Metallica</p>
      <p>Label: Elektra Records. Country: Denmark.</p>
      <h3>A night at the Opera (1975)</h3>
      <p>Artist: Queen</p>
      <p>Label: EMI. Country: UK.</p>
   </body>
</html>
```

XSLT

Filter

Like in Python, XSLT allows you to filter out elements, e.g. in a for loop, when a certain condition is met.

The construct **<xsl:if>** is used to specify the condition in the attribute **@test**.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3   <xsl:template match="/">
4     <html>
5       <head>
6         <title>My music</title>
7       </head>
8       <body>
9         <h2>My music collection</h2>
10          <xsl:for-each select="catalog/cd">
11            <xsl:if test='country/text() = "UK"'>
12              <h3><xsl:value-of select="title"/> (<xsl:value-of select="year"/>)</h3>
13            </xsl:if>
14          </xsl:for-each>
15      </body>
16    </html>
17  </xsl:template>
18 </xsl:stylesheet>
```

XSLT

Filter

The value of @test is either a **XPath** if you want to check the existence of a node, or an **expression** including an operator. Common operators include + , - , * , = , != , > , < , or , and .

Also functions can be used, e.g. string-length.

```
1 country/text() # the text node exists
2 string-length(country/text()) > 0 # the length is greater than zero
3 country/text() = "UK" # the text exists and is equal to "UK"
4
5 year < 2000 # year is less than 2000
6 year != 2009 # year is anything but 2009
7 year = 2009 or year = 1986 # year is either 2009 or 1986
8 year > 2000 and year < 2010 # year is greater than 2000 and less than 2009
```

XSLT

Choose

Notably, only a construct for describing the **if** condition exists, while there is no construct for describing the **else** condition.

To manage several scenarios, <**xsl:choose**> is used. It includes one or more <**xsl:when**> conditions and one <**xsl:otherwise**> to define a default behaviour in case none of the prior conditions is met.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>My music</title>
      </head>
      <body>
        <h2>My music collection</h2>
          <xsl:for-each select="catalog/cd">
            <xsl:choose>
              <xsl:when test='year &gt; 2000'>
                <h3><xsl:value-of select="title"/> (from the '00)</h3>
              </xsl:when>
              <xsl:when test='year &gt; 1980 and year &lt; 1990'>
                <h3><xsl:value-of select="title"/> (from the '80)</h3>
              </xsl:when>
              <xsl:otherwise>
                <h3><xsl:value-of select="title"/> (older than you think!)</h3>
              </xsl:otherwise>
            </xsl:choose>
          </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

Notice the way > and < have been replaced with the escaped value

XSLT

Sort

You can sort elements in a sequence (e.g. in a for loop) according to one or more parameter, thanks to <**xsl:sort**>.

Notice the element xsl:sort affects results of the parent instruction

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>My music</title>
      </head>
      <body>
        <h2>My music collection</h2>
          <xsl:for-each select="catalog/cd">
            <xsl:sort select="year"/>
            <h3><xsl:value-of select="title"/></h3>
          </xsl:for-each>
      </body>
    </html>
  </xsl:template>
</xsl:stylesheet>
```

```xml
<html>
   <head>
      <title>My music</title>
   </head>
   <body>
      <h2>My music collection</h2>
      <h3>A night at the Opera</h3>
      <h3>Master of Puppets</h3>
      <h3>Resistance</h3>
   </body>
</html>
```

Exercise

Practice with XSLT

Use the same XML file and write a **<xsl:if>** condition for these scenarios (no need to return the whole HTML structure, return just strings):

1. The title of songs that are not produced by Warner records
2. The name of artists that wrote a song after 2000
3. The name of labels that are active in Denmark

Write a **<xsl:choose>** condition for this scenario:

- If produced by a UK-based label return “British!”, otherwise “Somewhere else”

XSLT

Apply templates

Templating rules can be **externalised**.

Instead of declaring the HTML output and inject the rule we declare rules somewhere else and call them in the point of the HTML we need it with <**xsl:apply-templates**>. The order is important!

In python this mechanism is similar to function declaration.

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
3    <xsl:template match="/">
4      <html>
5        <head>
6          <title>My music</title>
7        </head>
8        <body>
9          <h2>My music collection</h2>
10           <xsl:for-each select="catalog/cd">
11             <section>
12                 <xsl:apply-templates/>
13             </section>
14           </xsl:for-each>
15       </body>
16     </html>
17   </xsl:template>
18
19   <xsl:template match="title">
20     <h3><xsl:value-of select="."/></h3>
21   </xsl:template>
22
23   <xsl:template match="artist">
24     <p><xsl:value-of select="."/></p>
25   </xsl:template>
26
27   <xsl:template match="year | label | country"/>
28 </xsl:stylesheet>
```

Three templates are called by the above one.

Notice the XPaths in @match: these imply the context node is "catalog/cd", otherwise they won't match any node.

XSLT

Apply templates

This is the result of the prior transformation.

Notice that elements can be **ignored** by applying an empty template. If not specified, the text content of those nodes is included in the final output.

Try it! Remove l.27 fro the xslt file and see the result.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
  <xsl:template match="/">
    <html>
      <head>
        <title>My music</title>
      </head>
      <body>
        <h2>My music collection</h2>
          <xsl:for-each select="catalog/cd">
            <section>
                <xsl:apply-templates/>
            </section>
          </xsl:for-each>
      </body>
    </html>
  </xsl:template>

  <xsl:template match="title">
    <h3><xsl:value-of select="."/></h3>
  </xsl:template>

  <xsl:template match="artist">
    <p><xsl:value-of select="."/></p>
  </xsl:template>

  <xsl:template match="year | label | country"/>
</xsl:stylesheet>
```

```xml
<html>
   <head>
      <title>My music</title>
   </head>
   <body>
      <h2>My music collection</h2>
      <section>
         <h3>Resistance</h3>
         <p>Muse</p>
      </section>
      <section>
         <h3>Master of Puppets</h3>
         <p>Metallica</p>
      </section>
      <section>
         <h3>A night at the Opera</h3>
         <p>Queen</p>
      </section>
   </body>
</html>
```

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XSLT

How to use it?

To transform XML documents into HTML using XSLT there are a few technical solutions:

- **Stand-alone (or server-side)**: a software + a XSL processor to produce a static HTML file.
- **Client-side**: the browser (Javascript) dynamically produces a HTML file on demand

XSLT

Stand-alone transformations

You can use:

- **online tools** like Free Formatter
- **editors** with an integrated XSLT Processor (e.g. Oxygen, which requires an Academic license) or with XSLT plugins (VS Code + XSLT/XPath extension + Saxon processor)
- Command-line programs
- Programming languages

XSLT

Command-line transformation

Requirements

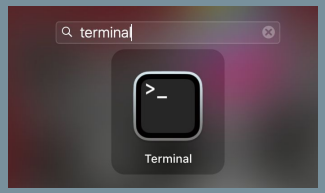
- Create a folder **`Desktop/tesr_dhdk/5_exercise`**
- **Download** XML and XSL files in the folder
- Check or install Java: go to the terminal, type **`java -version`** and press enter. If it returns an error, you must install it - click on the link above, choose your OS, use the installer
- Download Saxon (for practical reasons, unzip it in the folder with XML and XSL files)

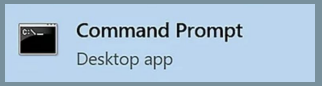
XSLT

Command-line transformation

Go to terminal

- Mac (go to launchpad, type Terminal and go)
- Windows (next to the menu, search Command Prompt)





Navigate to your folder

- Mac: type **`cd Desktop/tesr_dhdk/5_exercise`**
- Windows: type **`cd c:\Users\YOURUSERNAME\Desktop\tesr_dhdk\5_exercise`**

Press enter

XSLT

Command-line transformation

Type this command and press enter:

On Mac

```
java -jar SaxonHE11-4J/saxon-he-11.4.jar -s:5_catalogue.xml -xsl:5_catalogue.xsl
-o:index.html
```

On Windows

```
java -jar SaxonHE11-4J\saxon-he-11.4.jar -s:5_catalogue.xml -xsl:5_catalogue.xsl
-o:index.html
```

XSLT

Command-line transformation

You should now have a index.html file in your folder. Open it with the browser and check the HTML file in an editor.

To learn more on how to use Saxonica command-line programs, look here

XSLT

Server-side transformations

Python, among other languages, offers APIs (i.e. libraries) to manipulate XML documents, e.g. **lxml.**

Lxml allows you to parse and query XML documents, use XPath to identify and retrieve nodes, as well as perform XSL transformations.

See more information here.

XSLT

Server-side transformations

- Go to the shell/terminal and type **`pip install lxml`** to install the library.
- See an example of XSL transformation with Python here
- You can download the file and run it in your laptop: open the shell (or VS Code), navigate to the folder where you downloaded the python file and run it **`python3 5_catalogue.py`**
- See the resulting html here

XSLT

Client-side transformations

Some browsers (e.g. Safari) allow you to perform a XSL transformation on the fly.

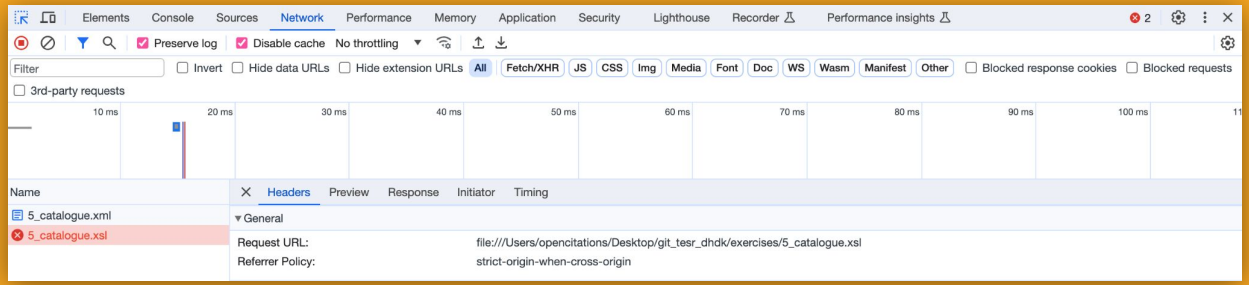
You must specify a XSL file in the XML document and open the XML document in the browser.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <?xml-stylesheet type="text/xsl" href="cdcatalog.xsl"?>
3 <catalog>
4   ...
5 </catalog>
```

XSLT

Client-side transformations

In some browsers (e.g. Chrome, Firefox) you will see an error is returned, due to **CORS cross origin access restrictions** (see below the inspector web of Chrome). This is because the browser interprets the XSL file as in a different server (what..?!) than the one hosting the XML file, then it prevents you to do it for security reasons.



XSLT

Client-side transformations

To overcome CORS barriers, we use **Javascript** to instruct the browser on how to apply a XSL stylesheet on a XML file. To actually overcome the restrictions, we publish our files on an online (or local) web server (in our case github pages), so that the origin of the two files (XML and XSLT) is the same.

Javascript?
Javascript is a programming language widely used for web development along with HTML and CSS. A Javascript script is usually included in a **.js** file and referenced in a html file with the tag <script> in order to be imported. Alternatively, js code can be injected in HTML files.

XSLT

Client-side transformations

We create a skeleton HTML file (e.g. index.html) where we include a JS script that **renders the XML as HTML snippet** on demand.

See a demo: https://marilenadaquino.github.io/tesr_dhdk/exercises/catalogue_js.html

Source code: https://github.com/marilenadaquino/tesr_dhdk/blob/main/exercises/catalogue_js.html

PS. You can use this code in your final project to show the XML to HTML transformation of your XML/TEI file

Read more

Learn XSLT

To keep learning XSLT, use the following materials and look at the extended bibliography:

- Mozilla documentation
- W3School tutorial
- Python etree + XSLT tutorial

XML/TEI to HTML

Tools

Several works exist trying to generalise the transformation of XML/TEI documents into HTML, mainly using XSLT developed by community members, attempting to foresee any potential combination of TEI elements in a given XML document.

- Tei2html, a collection of XSL files developed for the project Gutenberg
- TEI community XSLT stylesheets
- TEIGarage conversion tool
- TEI Boilerplate

NB. For the sake of the exam, you can either use of these tools, or you can create your own XSLT stylesheet. In case you reuse something existing, you will have to document the process you used on the project website.

Homework

Transform XML/TEI to HTML

Given the XML file, create a XSLT file to transform it in a valid HTML file.
Use the command-line program or the python script or the online tool to validate the transformation.

```
1 <xsl:stylesheet version="1.0"
2    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
3    xmlns:tei="http://www.tei-c.org/ns/1.0">
4  <xsl:template match="/">
5    <html>
6      <head>
7        <title><xsl:value-of select="tei:TEI/tei:text/tei:body/tei:div/tei:head/text()"/></title>
8      </head>
9      ...
10   </html>
11 </xsl:template>
12 </xsl:stylesheet>
```

Pay attention to the use of namespaces and prefixes!