

Text Encoding and Semantic Representation

XML family | HTML

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

XML-family

Markup overlap

Markup overlap is a well-known problem in the XML (and TEI) community.

- Different (logical, metrical, linguistic, content) structures may apply to the same portion of text (e.g. a <persName> split across multiple verse lines <|>).
- Texts with a complex tradition require to annotate small chunks of text for which not only multiple interpretations exist, but interpretations tend to encompass overlapping chunks. (e.g. “textA textB textC” has 2 rdgs for the chunks “textA textB” and “ textB textC”)

XML-family

Some solutions

Several solutions have been proposed, among which some that have already been described:

- The usage of milestone elements (e.g. <anchor>) to frame text chunks
- The usage of dedicated elements (e.g. listPerson, **standOff**) to record other information
- The usage of external files to record annotations (stand-off markup)

Some solutions include non-XML languages to record and link annotations. See [Wikipedia](#) for a summary and [TEI guidelines](#) for detail.

XML-family

XInclude
XPointer
XLink
XPath
XQuery
XSLT

Identify and link XML docs

Notably, when annotations are recorded in an external document, the latter does not include both text and annotations, but only references to the original text. Therefore, stand-off markup requires **mechanisms to identify XML snippets** and link them into another document.

Such mechanisms are used in a plethora of scenarios that go beyond standoff markup, e.g. **query** XML documents to retrieve pieces of information, **transform** XML-like documents into other XML-like documents, etc.

XML-family

XML standards

Some standards part of the XML-family contribute to these tasks:

- **XInclude, XLink**: standards to link documents
- **XPointer, XPath**: syntaxes to identify XML snippets
- **XSLT**: transform XML-like documents into other XML-like documents
- **XQuery**: query and transform XML documents

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XML-family

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XML standards

Some standards part of the XML-family contribute to these tasks:

- **XInclude**, **XLink**: standards to link documents
- **XPointer**, **XPath**: syntaxes to identify XML snippets
- **XSLT**: transform XML-like documents into other XML-like documents
- **XQuery**: query and transform XML documents

We present all the standards, but we focus on these three

Xinclude

Definition

XInclude is a standard that specifies a syntax for the inclusion within an XML document of data fragments placed in different resources (e.g. other XML files or text files). It defines a namespace (`xi:http://www.w3.org/2001/XInclude`), and two elements, `<xi:include>` and `<xi:fallback>`.

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0"?>
2 <book xmlns:xi="http://www.w3.org/2001/XInclude">
3   <xi:include href="frontmatter.xml"/>
4   <xi:include href="part1.xml"/>
5   <xi:include href="part2.xml"/>
6   <xi:include href="part3.xml"/>
7   <xi:include href="backmatter.xml"/>
8 </book>
```


Xinclude

Example

XInclude

XPointer

XLink

XPath

XQuery

XSLT

The element `<xi:include>` uses the attribute `@href` to reference a XML document (or part of it) to be included in a second XML file

NB. in the example we assume this file and the referenced ones are in the same folder.

```
1 <?xml version="1.0"?>
2 <book xmlns:xi="http://www.w3.org/2001/XInclude">
3   <xi:include href="frontmatter.xml"/>
4   <xi:include href="part1.xml"/>
5   <xi:include href="part2.xml"/>
6   <xi:include href="part3.xml"/>
7   <xi:include href="backmatter.xml"/>
8 </book>
```

XInclude

XPointer

The attribute @xpointer may reference fragments of the XML document, e.g. a node.

To do so, it uses the **XPointer** mechanism, which specifies rules (syntax and functions) to identify XML nodes (elements), e.g. via their identifier

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version='1.0'?>
2 <!DOCTYPE price-list SYSTEM "price-list.dtd">
3 <price-list xml:lang="en-us">
4   <item id="w001">
5     <description id="w001-description">
6       <p>Normal Widget</p>
7     </description>
8     <prices id="w001-prices">
9       <price currency="USD" volume="1+">39.95</price>
10      <price currency="USD" volume="10+">34.95</price>
11      <price currency="USD" volume="100+">29.95</price>
12    </prices>
13  </item>
14  <item id="w002">
15    <description id="w002-description">
16      <p>Super-sized widget with bells <i>and</i> whistles.</p>
17    </description>
18    <prices id="w002-prices">
19      <price currency="USD" volume="1+">59.95</price>
20      <price currency="USD" volume="10+">54.95</price>
21      <price currency="USD" volume="100+">49.95</price>
22    </prices>
23  </item>
24 </price-list>
```

```
1 <?xml version='1.0'?>
2 <price-quote xmlns:xi="http://www.w3.org/2001/XInclude">
3   <prepared-for>Joe Smith</prepared-for>
4   <xi:include href="price-list.xml" xpointer="w002-description"/>
5   <xi:include href="price-list.xml" xpointer="element(w002-prices/2)"/>
6 </price-quote>
```

XInclude

XPointer

or via their position in the hierarchy

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version='1.0'?>
2 <!DOCTYPE price-list SYSTEM "price-list.dtd">
3 <price-list xml:lang="en-us">
4   <item id="w001">
5     <description id="w001-description">
6       <p>Normal Widget</p>
7     </description>
8     <prices id="w001-prices">
9       <price currency="USD" volume="1+">39.95</price>
10      <price currency="USD" volume="10+">34.95</price>
11      <price currency="USD" volume="100+">29.95</price>
12    </prices>
13  </item>
14  <item id="w002">
15    <description id="w002-description">
16      <p>Super-sized widget with bells <i>and</i> whistles.</p>
17    </description>
18    <prices id="w002-prices">
19      <price currency="USD" volume="1+">59.95</price>
20      <price currency="USD" volume="10+">54.95</price>
21      <price currency="USD" volume="100+">49.95</price>
22    </prices>
23  </item>
24 </price-list>
```

```
1 <?xml version='1.0'?>
2 <price-quote xmlns:xi="http://www.w3.org/2001/XInclude">
3   <prepared-for>Joe Smith</prepared-for>
4   <xi:include href="price-list.xml" xpointer="w002-description"/>
5   <xi:include href="price-list.xml" xpointer="element(w002-prices/2)"/>
6 </price-quote>
```

XPointer

XPointer and XPath

XPointer was born as an **extension** of another language, i.e. XML Path language, or XPath, that we will discuss more in depth. It includes a syntax and a number of functions to identify XML nodes (e.g. elements, text, groupings).

XPointer is **more powerful** than XPath, although it is less flexible and can be used in less contexts. In particular, XPointer can be used with XInclude and XLink, while XPath can be used with any of the previous one, XQuery and XSLT.

XLink

Definition

XLink is a mechanism used to create hyperlinks between XML documents. However, such links do not work in a browser like hypertext links in HTML.

The XLink namespace is <http://www.w3.org/1999/xlink> and has three main attributes: @type, @href, and @show

XInclude
XPointer
XLink
XPath
XQuery
XSLT

XLink

Example

@xlink:type="simple" creates a "HTML-like" link (click on a word and get redirected to another page or multimedia)

@xlink:href attribute specifies the URL to link to (e.g. an image or another XML file)

@xlink:show="new" specifies whether the link should open in a new window.

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <bookstore xmlns:xlink="http://www.w3.org/1999/xlink">
2
3   <book title="Harry Potter">
4     <description
5       xlink:type="simple"
6       xlink:href="/images/HPotter.gif"
7       xlink:show="new">
8       As his fifth year at Hogwarts School of Witchcraft and
9       Wizardry approaches, 15-year-old Harry Potter is.....
10    </description>
11  </book>
12 </bookstore>
```

XPath

XInclude
XPointer
XLink
XPath
XQuery
XSLT

Definition

XPATH is another standard part of the XML-family.

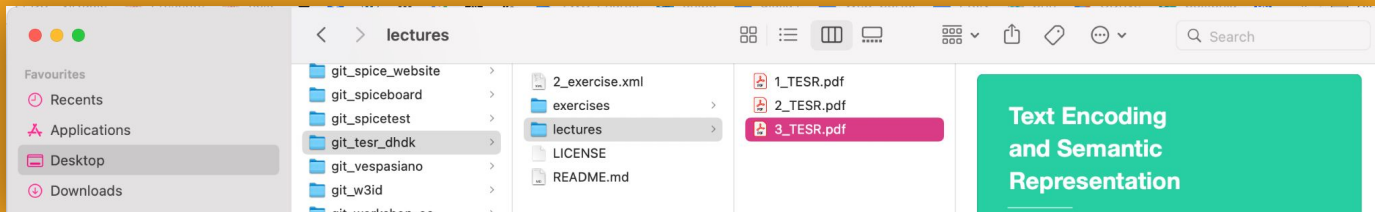
Like XPointer, XPath provides a notation and a series of functions that can be used to navigate through elements, text, and attributes in an XML document. It is a fundamental building block of XQuery and XSLT. Currently, XPath 3.0 is available.

XPath

XInclude
XPointer
XLink
XPath
XQuery
XSLT

Paths and nodes

Similarly to when you browse your file system, XPath provides a syntax to express a path to reach a resource. In this case the resource is not a file on your laptop but a node.



XPath

Nodes

In XPath there are 7 types of nodes: **element**, **attribute**, **text**, namespace, processing-instruction, comment, and **root** nodes.

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5     <author>J K. Rowling</author>
6     <year>2005</year>
7     <price>29.99</price>
8   </book>
9 </bookstore>
```

XPath

Axes

Parent: title > book

Children: book > title, author, year, price

Siblings: title > author, year, price

Ancestors: title > book, bookstore

Descendants: bookstore > book, title, author ...

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5     <author>J K. Rowling</author>
6     <year>2005</year>
7     <price>29.99</price>
8   </book>
9 </bookstore>
```

XPath

Paths

In XPath you always start the journey from the root element, if not specified differently.

Some examples of XPATH syntax:

bookstore

Returns the node that matches the name

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5     <author>J K. Rowling</author>
6     <year>2005</year>
7     <price>29.99</price>
8   </book>
9 </bookstore>
```

XPath

Paths

bookstore/book/author

Select the last children node in the above string. The path starts from the current element (bookstore).

Notice the path returns an element. To return its text use:

bookstore/book/author/text()

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5     <author>J K. Rowling</author>
6     <year>2005</year>
7     <price>29.99</price>
8   </book>
9 </bookstore>
```

XPath

Paths

bookstore//author

Select the last children node in the string, regardless of its level in the hierarchy (i.e. author does not have to be a direct child, rather it can be a descendant)

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5     <author>J K. Rowling</author>
6     <year>2005</year>
7     <price>29.99</price>
8   </book>
9 </bookstore>
```

XPath

Paths

//title/@lang

Returns the value of the attribute of the selected element

//@lang

Returns all the values of all the attributes @lang (even when several elements have the @lang attribute)

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5     <author>J K. Rowling</author>
6     <year>2005</year>
7     <price>29.99</price>
8   </book>
9 </bookstore>
```

XPath

Paths

•
The current node

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5     <author>J K. Rowling</author>
6     <year>2005</year>
7     <price>29.99</price>
8   </book>
9 </bookstore>
```

XPath

Paths

..

The parent node

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5     <author>J K. Rowling</author>
6     <year>2005</year>
7     <price>29.99</price>
8   </book>
9 </bookstore>
```


XPath

Predicates

/bookstore/book

All the children elements named book. The path starts from the root element.

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Predicates

/bookstore/book[1]

Returns only the n-th (first in this case) of the children elements called book

The horror!!

Notice the index in XPath starts from 1, while in Python (and many other programming languages) it starts from 0.

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Predicates

`/bookstore/book[last()]`

Only the last of the children
elements called book

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Predicates

`/bookstore/book[last()-1]`

Only the last but one of the
children elements called book

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12 </bookstore>
```

XPath

Predicates

`//title[@lang]`

All the elements title (at any point of the hierarchy) that have an attribute @lang

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Predicates

`//title[@lang='it']`

Only the elements title (at any point of the hierarchy) that have an attribute @lang with value 'it'

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Wildcards

*

Any element node

*@

Any attribute node

node()

Any kind of node (elements, attributes, etc.)

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Wildcards

//*

Any element in the document

/bookstore/*

Any element child of bookstore

node()

Any kind of node (elements, attributes, etc.)

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```


XPath

Multipaths

`//book/title | //book/title/@lang`

Select all elements title AND all the values of the attribute @lang of title elements

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Axes

`/x` or `//x` or `x` ?

When a path starts with `/`, the first element name specified is the **root** element.

When a path starts with `//`, the element name specified is a **descendant** element

When the path does not start with any `/`, the element is the **child** of the current one.

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Axes

/x or //x or x ?

Knowing which one to use is important when **traversing** a document.

This means that several iterations over the document hierarchy may happen, and the starting point may not be always the same.

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Axes

For instance, the first iteration over a document would require us to access the root element at first

/bookstore/book

At this point the, say, “cursor” is lower in the hierarchy, at the level of book elements.

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Axes

To keep traversing the tree, and e.g. access the value of @lang, we can either specify and **absolute path** (which starts from the root)

/bookstore/book/title/@lang

Or we can specify **relative path**, starting from the current element (book)

title/@lang

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Axes

XPATH provides also some magic words to traverse XML axes. Context: **/bookstore**

child::title/text()

Returns the text of title, if direct child of the context node (bookstore). Here it does not return anything

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XPath

Axes

Context: **/bookstore**

descendant::title

Returns all the title elements that are children or grandchildren of the context node (bookstore).

child::*

Select all direct children (*) of the context node (i.e. all book elements)

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

See more examples [here](#), [here](#) or [here](#)

XPath

Axes

Context: **title**

parent::title

Returns all the book elements, if direct parent of the context node (bookstore).

Context: **book[3]**

preceding-sibling::book

Select all previous book elements at the same level of the hierarchy if their name is book

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

See more examples [here](#), [here](#) or [here](#)

Exercise

Practice with XPath

Consider the example XML document. Write an XPATH expression to return the following elements:

- The root element
- The value of the attribute @lang of the second book title
- The first AND the last book title
- All children of the first book element
- The text of all the title elements

Try your paths [here](#) or [here](#)

Read more

Practice with XPath

Online resources to keep practicing with XPath

- Mozilla [Guide](#)
- W3School [tutorial](#)
- [Others](#)

XQuery

In pills

XQuery is the official query language of XML. **It uses XPath** to select and retrieve XML nodes in one or more documents. It is supported by most NOSQL databases, meaning, databases designed to store XML documents.

It can be used to extract and retrieve information from a collection of XML documents, or to transform XML documents into other XML-like documents (e.g. into HTML documents).

NB. In this tutorial we see only examples of how to use XQuery to query XML documents, while we will use XSLT to transform XML documents into HTML documents. For the sake of your project, you are free to choose between XQuery, XSLT, or python only, to transform XML into HTML.

- XInclude
- XPointer
- XLink
- XPath
- XQuery**
- XSLT

XQuery

Example

`doc("books.xml")/bookstore/book/title`

First, the function `doc()` specifies the name of the file to open. The following XPath specifies which elements to return.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XInclude

XPointer

XLink

XPath

XQuery

XSLT

XQuery

FLWOR

FOR LET WHILE ORDER BY RETURN

FLWOR (read “flower”) is an acronym that specifies more sophisticated constructs that can be used in XQuery.

Similar constructs can be found in Python (and any other programming language). Like in Python you do not need to use all the constructs at the same time, but only when applicable.

XInclude
XPointer
XLink
XPath
XQuery
XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XQuery

FLWOR

FOR iterate over a sequence of nodes, e.g.

for $\$x$ in doc("books.xml")/bookstore/book

LET binds a sequence to a variable, e.g.

let $\$enLang$:= 'en'

WHERE filters nodes, e.g.

where $\$x/title[@lang = \$enLang]$

XInclude

XPointer

XLink

XPath

XQuery

XSLT

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XQuery

FLWOR

ORDER BY sorts results, e.g.
order by \$x/title

RETURN defines results to be returned, e.g.
return \$x/title

XInclude
XPointer
XLink
XPath
XQuery
XSLT

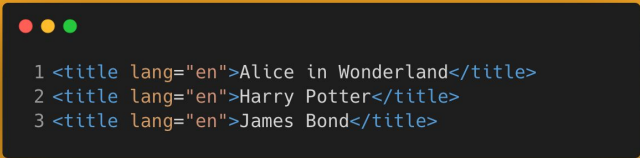
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <bookstore>
3   <book>
4     <title lang="en">Harry Potter</title>
5   </book>
6   <book>
7     <title lang="it">Harry Potter</title>
8   </book>
9   <book>
10    <title lang="en">James Bond</title>
11  </book>
12  <book>
13    <title lang="en">Alice in Wonderland</title>
14  </book>
15 </bookstore>
```

XQuery

FLWOR

```
for $x in doc("books.xml")/bookstore/book
let $enLang := 'en'
where $x/title[@lang = $enLang]
order by $x/title
return $x/title
```

XInclude
XPointer
XLink
XPath
XQuery
XSLT



```
1 <title lang="en">Alice in Wonderland</title>
2 <title lang="en">Harry Potter</title>
3 <title lang="en">James Bond</title>
```


XQuery

FLWOR

If I want to change the structure of the output XML I can specify new elements in the return clause.

```
for $x in doc("books.xml")/bookstore/book
let $enLang := 'en'
where $x/title[@lang = $enLang]
order by $x/title
return <li>{data($x/title)}</li>
```



```
1 <li>Alice in Wonderland</li>
2 <li>Harry Potter</li>
3 <li>James Bond</li>
```

Rather than returning entire element nodes (e.g. title) I can return only their text content using the function **data()**

- XInclude
- XPointer
- XLink
- XPath
- XQuery**
- XSLT

Exercise

Practice with XQuery

Consider the example XML document. Write a query in XQuery to return the following result:

```
en  
it  
en  
en
```

Try your query [here](#)

Read more

Practice with XQuery

Online resources to keep practicing with XQuery

- Official [Website](#)
- W3School [tutorial](#)
- [Others](#)

HTML

In pills

HyperText Markup Language is the w3c standard markup language of the web. Unlike XML, it is used to encode texts to be **presented** on a web page.

Like XML, it has a prolog, a root element `<html>`, two mandatory elements `<head>` and `<body>` and plenty of optional ones.

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>The title of the webpage on the browser tab!</title>
5   </head>
6   <body>
7     <h1>The main title on the web page</h1>
8     <p>This is an example paragraph. Everything in the
9     <strong>body</strong> element will appear on the page.</p>
10  </body>
11 </html>
```

Save this content in a file called index.html and open it in a browser

HTML

HTML vs XML

The syntax (i.e. the grammatical rules) is very similar to that of XML.

However HTML uses its own **vocabulary** of elements and attributes, while XML can be used with any schema/DTD, and it has its own **rules** (e.g. a title should not include a paragraph).

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>The title of the webpage on the browser tab!</title>
5   </head>
6   <body>
7     <h1>The main title on the web page</h1>
8     <p>This is an example paragraph. Everything in the
9     <strong>body</strong> element will appear on the page.</p>
10  </body>
11 </html>
```

Save this content in a file called index.html and open it in a browser

HTML

HTML vocabulary

The elements and attributes that can be used in HTML address **presentational** aspects (e.g. `` represents a text in bold) and **logical** elements of the page structure (`<h1>` is a title, `<p>` a paragraph, etc.).

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>The title of the webpage on the browser tab!</title>
5   </head>
6   <body>
7     <h1>The main title on the web page</h1>
8     <p>This is an example paragraph. Everything in the
9     <strong>body</strong> element will appear on the page.</p>
10  </body>
11 </html>
```

Save this content in a file called index.html and open it in a browser

Save this content in a file called index.html

HTML

HTML vs CSS

The objective of HTML is to inform a client application (i.e. a browser) on how content of a web page should be **formatted**.

Browsers associate HTML elements with **basic styles** - e.g. if you open a HTML file in the browser you will see titles in bold and bigger font, even though it not specified anywhere how titles should be displayed.

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>The title of the webpage on the browser tab!</title>
5   </head>
6   <body>
7     <h1>The main title on the web page</h1>
8     <p>This is an example paragraph. Everything in the
9     <strong>body</strong> element will appear on the page.</p>
10  </body>
11 </html>
```

Save this content in a file called index.html and open it in a browser

HTML

HTML + CSS

Indeed, how to render a HTML file is delegated to the Cascading Style Sheets (CSS) language.

That is, HTML files are served on the web along with **.css files** in which are included instructions on how to style elements.

```
1 <!doctype html>
2 <html>
3   <head>
4     <title>The title of the webpage on the browser tab!</title>
5     <link rel="stylesheet" href="styles.css">
6   </head>
7   <body>
8     <h1>The main title on the web page</h1>
9     <p>This is an example paragraph. Everything in the
    <strong>body</strong> element will appear on the page.</p>
10  </body>
11 </html>
```

style.css

```
1 h1 {
2   font-size: 30px;
3   color: red;
4   border: solid 5px red;
5 }
6
7 p {
8   color: blue;
9   width: 30%;
10 }
```

So, HTML tells the browser what is a title or a paragraph, and CSS tells what color, font, margins, dimensions, etc. to use to show them.

HTML

HTML in practice

Most of the syntax rules that apply to XML apply also to HTML. A few differences include:

- The prolog
- The encoding declaration
- Tags are not case-sensitive in HTML (meta = META)

```
1 <!-- HTML -->
2 <!doctype html>
3 <html>
4   <head>
5     <meta charset="UTF-8">
6   </head>
7   <body>...</body>
8 </html>
```

```
1 <!-- XML -->
2 <?xml version="1.0" encoding="UTF-8"?>
3 <anyTag>
4   <anyChild>...</anyChild>
5 </anyTag>
```

HTML

HTML in practice

HTML requires 3 mandatory elements:

- **html** the root element
- **head** includes instructions (e.g. meta, links to css files). The element title is shown in the browser tab, all other children are not visible
- **body** includes texts and links to multimedia to be visualised on the web page

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Hello world</title>
6   </head>
7   <body>
8     <header>
9       <nav>
10        <ul>
11          <li><a href="about.html">About me</a></li>
12          <li><a href="contact.html">Contact me</a></li>
13        </ul>
14      </nav>
15      <h1>Hello world!</h1>
16    </header>
17    <main>
18      <p>This is my first <strong>website</strong>.</p>
19    </main>
20    <footer>
21      <p>Check my profile on
22        the <a href="https://www.unibo.it/sitoweb/marilena.daquino2">Unibo website</a></p>
23    </footer>
24  </body>
25 </html>
```

HTML

head

The element head includes among others:

- **meta** (recommended) to specify the encoding of characters (if you forget it special characters will not be correctly rendered)
- **title** (recommended) to create a title placeholder on the browser tab
- **link** (optional) to explain the browser where to find CSS instructions

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Hello world</title>
6   </head>
7   <body>
8     <header>
9       <nav>
10        <ul>
11          <li><a href="about.html">About me</a></li>
12          <li><a href="contact.html">Contact me</a></li>
13        </ul>
14      </nav>
15      <h1>Hello world!</h1>
16    </header>
17    <main>
18      <p>This is my first <strong>website</strong>.</p>
19    </main>
20    <footer>
21      <p>Check my profile on
22        the <a href="https://www.unibo.it/sitoweb/marilena.daquino2">Unibo website</a></p>
23    </footer>
24  </body>
25 </html>
```

HTML

body

The element body includes among others:

- **header** (optional) includes the visible heading of the page, such as the main title **h1** and the menu **nav** (both recommended)
- **main** (optional) includes the main content of the page (which is likely to change from one page to another)
- **footer** (optional) includes contacts and links

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Hello world</title>
6   </head>
7   <body>
8     <header>
9       <nav>
10        <ul>
11          <li><a href="about.html">About me</a></li>
12          <li><a href="contact.html">Contact me</a></li>
13        </ul>
14      </nav>
15      <h1>Hello world!</h1>
16    </header>
17    <main>
18      <p>This is my first <strong>website</strong>.</p>
19    </main>
20    <footer>
21      <p>Check my profile on
22        the <a href="https://www.unibo.it/sitoweb/marilena.daquino2">Unibo website</a></p>
23    </footer>
24  </body>
25 </html>
```

HTML

nav

The **nav** element includes an unordered list (**ul**) of list items (**li**). If no style is applied to the HTML document, this is visualised as a bullet list.

`` elements, when included in `<nav>`, usually include the element anchor (**a**) which is the element used to create hypertext links.

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Hello world</title>
6   </head>
7   <body>
8     <header>
9       <nav>
10        <ul>
11          <li><a href="about.html">About me</a></li>
12          <li><a href="contact.html">Contact me</a></li>
13        </ul>
14      </nav>
15      <h1>Hello world!</h1>
16    </header>
17    <main>
18      <p>This is my first <strong>website</strong>.</p>
19    </main>
20    <footer>
21      <p>Check my profile on
22        the <a href="https://www.unibo.it/sitoweb/marilena.daquino2">Unibo website</a></p>
23    </footer>
24  </body>
25 </html>
```

HTML

a

The **value** of `<a>` can be a text node (e.g. “about me”) or a node to include multimedia (e.g. an image ``, a video `<video>`, an audio `<audio>`).

When it includes a text, the default browser behaviour is to visualise it **blue and underlined** (or violet if already clicked).

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Hello world</title>
6   </head>
7   <body>
8     <header>
9       <nav>
10        <ul>
11          <li><a href="about.html">About me</a></li>
12          <li><a href="contact.html">Contact me</a></li>
13        </ul>
14      </nav>
15      <h1>Hello world!</h1>
16    </header>
17    <main>
18      <p>This is my first <strong>website</strong>.</p>
19    </main>
20    <footer>
21      <p>Check my profile on
22        the <a href="https://www.unibo.it/sitoweb/marilena.daquino2">Unibo website</a></p>
23    </footer>
24  </body>
25 </html>
```

HTML

a/@href

The **value** of @href is the URL of an external document.

It can be an html page that is part of the same website (in this case the URL is **relative**, since the linked file is usually stored in the same folder of the linking file) or an external one (in this case the URL is **absolute** and includes also the http(s) protocol and the domain of the external website)

```
1 <!doctype html>
2 <html>
3   <head>
4     <meta charset="UTF-8">
5     <title>Hello world</title>
6   </head>
7   <body>
8     <header>
9       <nav>
10        <ul>
11          <li><a href="about.html">About me</a></li>
12          <li><a href="contact.html">Contact me</a></li>
13        </ul>
14      </nav>
15      <h1>Hello world!</h1>
16    </header>
17    <main>
18      <p>This is my first <strong>website</strong>.</p>
19    </main>
20    <footer>
21      <p>Check my profile on
22      the <a href="https://www.unibo.it/sitoweb/marilena.daquino2">Unibo website</a></p>
23    </footer>
24  </body>
25 </html>
```

HTML

img

To include images, the tag `` is used. The attribute `@src` (source), similarly to `@href`, provides the relative/absolute URL of the image file to be displayed. The `@alt` attribute includes a text description of the image, in case it could not be visualised properly.

The element cannot include text, therefore it is often written in the short form.

```
1 
```

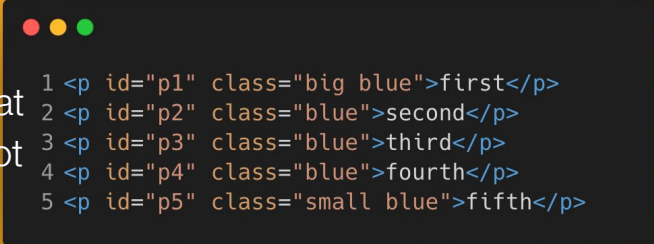

HTML

@id and @class

Every element in an html page can have the attributes @id and @class.

Similarly to @xml:id, **@id** specifies one user-defined value that is unique in the entire document (i.e. two elements should not have the same value for @id)

@class, instead, allows multiple values (separated by white space) and multiple elements can share the same values.



```
1 <p id="p1" class="big blue">first</p>
2 <p id="p2" class="blue">second</p>
3 <p id="p3" class="blue">third</p>
4 <p id="p4" class="blue">fourth</p>
5 <p id="p5" class="small blue">fifth</p>
```

HTML

@id and @class

@id and @class are mainly used in CSS to identify (groups of) HTML elements, when attributing styles.

Although there is no semantics associated to the values of such attributes, a good practice is to name them transparently, according to some internal convention, so as to facilitate their readability by external users.

```
1 <p id="p1" class="big blue">first</p>
2 <p id="p2" class="blue">second</p>
3 <p id="p3" class="blue">third</p>
4 <p id="p4" class="blue">fourth</p>
5 <p id="p5" class="small blue">fifth</p>
```

```
1 .big {
2     font-size: 100px;
3 }
4
5 #p3 {
6     text-decoration: underline;
7 }
```

HTML

inline and block elements

Indentation and new lines in HTML files are not parsed by the browser, that is, you cannot use “return” to create empty lines. Likewise, multiple white spaces are ignored. Instead block elements must be used to see the effect on the final web page.

`<nav>`, `<h1>`, `<section>`, `<p>`, etc are block elements (a new line is created after it is closed)
`<a>` and `` are inline elements (i.e. the text after this element is in the same line).

HTML

What you need to know to start

- Other than the 3 mandatory elements, any selection of HTML elements can be used.
- Any **rich text editor** would work to create a HTML document.
- If you open the HTML file in the **browser** you'll see the final result.
- If you modify the HTML file in the editor and refresh the browser you'll see **changes** immediately.
- If you call your file **index.html**, the browser will interpret the file as the homepage of the website (it is indeed mandatory to have a index.html file in every online website)
- If you upload your html and css files on a **github repository** and you activate [github pages](#), you'll be able to publish your website for free..

Exercise

Explore together an HTML page

Open the browser (chrome, firefox or safari) and go to the URL <https://runwebrun.com/themebau/>

Open the web inspector:

- Try folding/unfolding of elements
- Try commenting CSS instructions

Download the page (right click > Save as > “Web page, complete” to download page and folder)

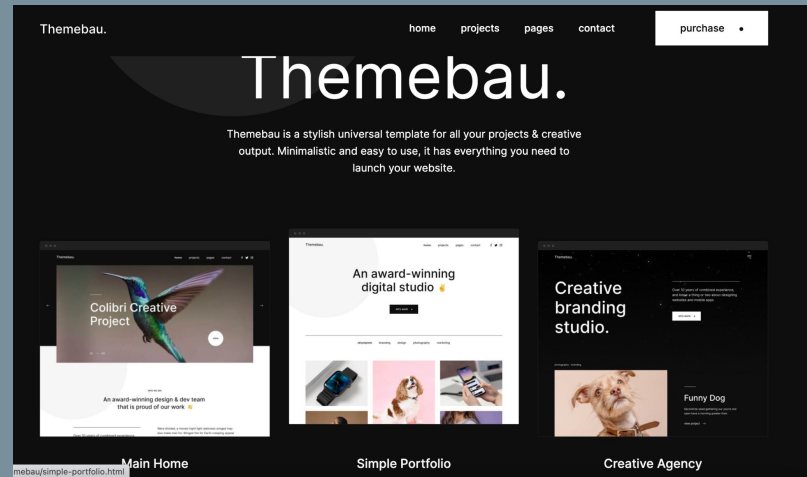
- Open the saved html page in the browser (double-click on the .html file)
- Rename the folder and refresh the page on the browser (where are all the styles?)

Exercise | Homework

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.



Read more

Practice with HTML

You will get to know more about HTML during the course “[Web technologies](#)” (prof. Vitali). If you want to study earlier:

- Mozilla [documentation](#) to learn all the elements (from here you also access CSS docs)
- w3School [tutorial](#) (from here you also access CSS tutorials)

For italian speakers there are also the lectures of the course “[Informatica Umanistica](#)”, in which we build a website from scratch, learning HTML, CSS and other tools.