

INTRODUCTION TO XQUERY

XQUERYINSTITUTE.ORG

Jonathan.Robie @ ibiblio.org / @jonathan_robie

MARKUP

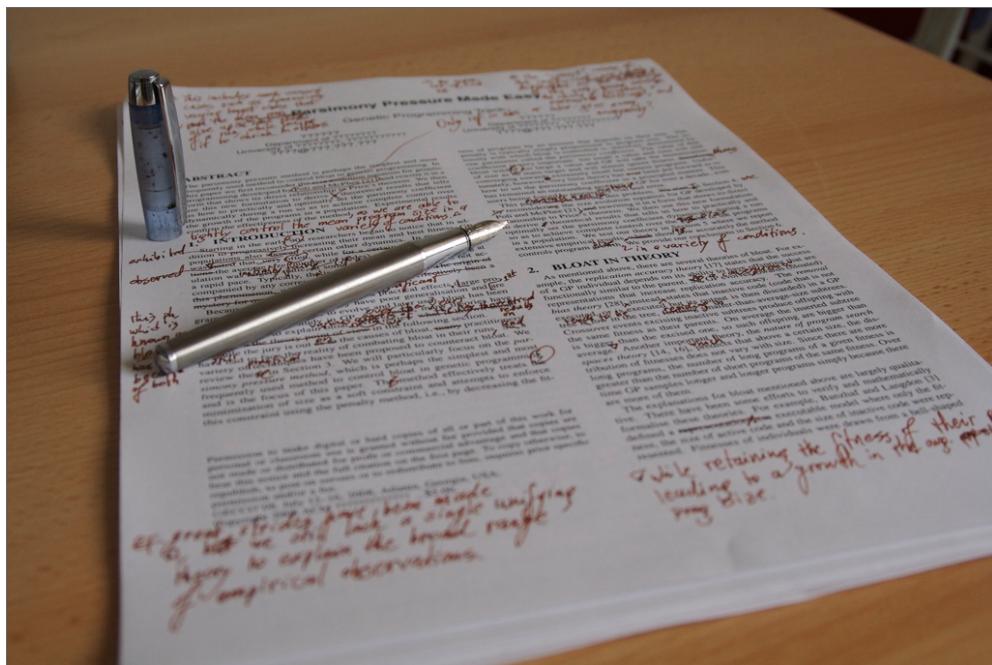


Photo by Nick McPhee, (CC BY-SA 2.0)

TEXT WITHOUT MARKUP (FOLGER EDITION)

Scene 1

*Enter Flavius, Marullus, and certain Commoners,
including a Carpenter and a Cobbler, over the stage.*

FLAVIUS

Hence! Home, you idle creatures, get you home!
Is this a holiday? What, know you not,
Being mechanical, you ought not walk
Upon a laboring day without the sign
Of your profession?—Speak, what trade art thou?

CARPENTER Why, sir, a carpenter.

MARULLUS

Where is thy leather apron and thy rule?
What dost thou with thy best apparel on?—
You, sir, what trade are you?

COBBLER Truly, sir, in respect of a fine workman, I am
but, as you would say, a cobbler.

MARULLUS

But what trade art thou? Answer me directly.
COBBLER A trade, sir, that I hope I may use with a safe
conscience, which is indeed, sir, a mender of bad
soles.

FLAVIUS

What trade, thou knave? Thou naughty knave, what
trade?

TEXT WITH MARKUP (JON BOSAK)

```
<ACT><TITLE>ACT I</TITLE>

<SCENE><TITLE>SCENE I. Rome. A street.</TITLE>
<STAGEDIR>Enter FLAVIUS, MARULLUS, and certain Commoners</STAGEDIR>

<SPEECH>
<SPEAKER>FLAVIUS</SPEAKER>
<LINE>Hence! home, you idle creatures get you home:</LINE>
<LINE>Is this a holiday? what! know you not,</LINE>
<LINE>Being mechanical, you ought not walk</LINE>
<LINE>Upon a labouring day without the sign</LINE>
<LINE>Of your profession? Speak, what trade art thou?</LINE>
</SPEECH>
```

TEXT WITH MARKUP (FOLGER EDITION)

```
<sp xml:id="sp-0001" who="#Flavius_JC">
  <speaker xml:id="spk-0001">
    <w xml:id="w0000400">FLAVIUS</w>
  </speaker>
  <ab xml:id="ab-0001">
    <lb xml:id="lb-00008"/>
    < milestone unit="ftln" xml:id="ftln-0001" n="1.1.1" ana="#verse"
      corresp="#w0000410 #p0000420 #c0000430 #w0000440 #p0000450 #c0000460 #w0000470 #c0000480">
      <w xml:id="w0000410" n="1.1.1">Hence</w>
      <pc xml:id="p0000420" n="1.1.1">!</pc>
      <c xml:id="c0000430" n="1.1.1"> </c>
      <w xml:id="w0000440" n="1.1.1">Home</w>
      <pc xml:id="p0000450" n="1.1.1">, </pc>
      <c xml:id="c0000460" n="1.1.1"> </c>
      <w xml:id="w0000470" n="1.1.1">you</w>
      <c xml:id="c0000480" n="1.1.1"> </c>
      <w xml:id="w0000490" n="1.1.1">idle</w>
      <c xml:id="c0000500" n="1.1.1"> </c>
      <w xml:id="w0000510" n="1.1.1">creatures</w>
      <pc xml:id="p0000520" n="1.1.1">!</pc>
      <c xml:id="c0000530" n="1.1.1"> </c>
      <w xml:id="w0000540" n="1.1.1">get</w>
      <c xml:id="c0000550" n="1.1.1"> </c>
      <w xml:id="w0000560" n="1.1.1">you</w>
      <c xml:id="c0000570" n="1.1.1"> </c>
      <w xml:id="w0000580" n="1.1.1">home</w>
      <pc xml:id="p0000590" n="1.1.1">!</pc>
    <lb xml:id="lb-00010"/>
    < milestone unit="ftln" xml:id="ftln-0002" n="1.1.2" ana="#verse"
      corresp="#w0000600 #p0000610 #c0000620 #w0000630 #p0000640 #c0000650 #w0000660 #c0000670 #w0000680 #c0000690">
```

TEXT WITH MARKUP (HTML)

```
<br/>
<span class="speaker">FLAVIUS</span>
<span class="indentInline">&nbsp;;</span>
<br/>
<a name="ftln-0001"> </a>
<span class="ftln" name="ftln_0001">FTLN 0001</span>
<span class="alignment indentVerse">&nbsp;</span>
<span id="line-1.1.1" title="1.1.1">Hence! Home, you idle creatures, get you home!</span>
<br/>
<a name="ftln-0002"> </a>
<span class="ftln" name="ftln_0002">FTLN 0002</span>
<span class="alignment indentVerse">&nbsp;</span>
<span id="line-1.1.2" title="1.1.2">Is this a holiday? What, know you not,</span>
<br/>
```

EDITORIAL MARKUP

function of t because both numerator and denominator are from generation to generation. Consider the more general case $g(\ell(x), t)$ where g is a real number (positive or negative). Here we can require that

$$c(t) = \text{Cov}(\ell, g) / \text{Cov}(\ell, \ell^k).$$

typo ~~1/2~~

Under the case $g(\ell(x), t) = c(t)(\ell(x) - \mu(t))$

Photo by Nick McPhee, **(CC BY-SA 2.0)**

EDITORIAL MARKUP - DIFF MARKUP

```
<item diff="add" at="2014-04-02">
  <p>The following comparison is <code>true</code> because atomization
  converts an array to its member sequence:</p>
  <eg role="parse-test">[ "Kennedy" ] = "Kennedy" </eg>
</item>
```

LINGUISTIC MARKUP

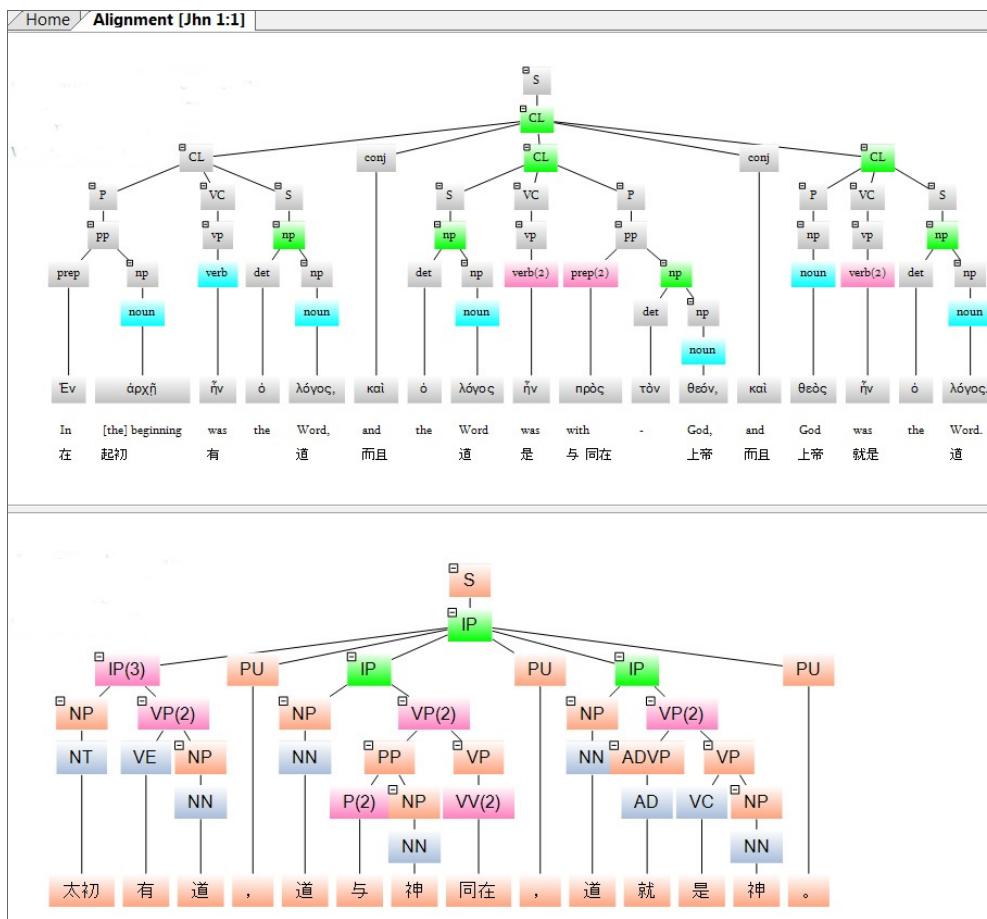


Image provided by **Global Bible Initiative**

LINGUISTIC MARKUP

```
<Tree>
<Node Cat="S" Head="0" nodeId="430010010010171">
  <Node Cat="CL" Start="0" End="16" Rule="Conj3CL" Head="0" nodeId="430010010010170">
    <Node Cat="CL" Start="0" End="4" Rule="P-VC-S" Head="0" nodeId="430010010010050">
      <Node Cat="P" Start="0" End="1" Rule="pp2P" Head="0" nodeId="430010010010021">
        <Node Cat="pp" Start="0" End="1" Rule="PrepNP" Head="1" nodeId="430010010010020">
          <Node Cat="prep" Start="0" End="0" StrongNumber="1722" UnicodeLemma="ἐν" FunctionalTag="PREP" ...>
            <Node Cat="np" Start="1" End="1" Rule="N2NP" Head="0" nodeId="430010010020011">
              <Node Cat="noun" Start="1" End="1" StrongNumber="746" UnicodeLemma="ἀρχή" Gender="Feminine" ...>
            </Node>
          </Node>
        </Node>
      </Node>
    <Node Cat="VC" Start="2" End="2" Rule="vp2VC" Head="0" nodeId="430010010030012">
      <Node Cat="vp" Start="2" End="2" Rule="V2VP" Head="0" nodeId="430010010030011">
        <Node Cat="verb" Start="2" End="2" StrongNumber="1510" UnicodeLemma="εἰμί" Person="Third" Mood="...>
      </Node>
    </Node>
  <Node Cat="S" Start="3" End="4" Rule="np2S" Head="0" HasDet="True" nodeId="430010010040021">
    <Node Cat="np" Start="3" End="4" Rule="DetNP" Head="1" HasDet="True" nodeId="430010010040020">
      <Node Cat="det" Start="3" End="3" StrongNumber="3588" UnicodeLemma="ὁ" Gender="Masculine" Number="...>
      <Node Cat="np" Start="4" End="4" Rule="N2NP" Head="0" nodeId="430010010050011">
        <Node Cat="noun" Start="4" End="4" StrongNumber="3056" UnicodeLemma="λόγος" Gender="Masculine" ...>
      </Node>
    </Node>
  </Node>
</Node>
<!-- This is the root node -->
```

XML available at **biblicalhumanities** [github](#)

PRESENTATIONAL MARKUP (HTML)

XQueryInstitute.org

This presentation's source ...

PRESENTATIONAL MARKUP (WIKI MARKUP)

The screenshot shows a Wikipedia edit page for the article "Editing Wiki markup". The page has a header with tabs for Article, Discussion, Read, Edit, View history, and Search. Below the header is the title "Editing Wiki markup". A "Preview" section follows, containing a paragraph about the definition of Wiki markup. The main content area contains sections for "External links" and "Advanced" editing tools. The "Advanced" section includes a toolbar with buttons for bold, italic, and other formatting, along with links to "Special characters" and "Help". Below the toolbar is a text area with the same content as the preview, followed by a section titled "External links" with two links to MediaWiki documentation.

Wiki markup or *wikitext language* is a [lightweight markup language](#) used to write pages in [wiki](#) websites, such as [Wikipedia](#), and is a simplified alternative/intermediate to [HTML](#). Its ultimate purpose is to be converted by [wiki software](#) into [HTML](#), which in turn is served to [web browsers](#).

There is no commonly accepted standard wikitext language yet. The grammar, structure, justification, keywords and so on depend on the particular wiki software used on the particular [website](#), such as [MediaWiki](#) on Wikipedia.

External links

- www.mediawiki.org/wiki/Wikitext_standard
- Alternative MediaWiki parsers

B I [Advanced](#) [Special characters](#) [Help](#)

'''Wiki markup''' or '''wikitext language''' is a [[lightweight markup language]] used to write pages in [[wiki]] websites, such as [[Wikipedia]], and is a simplified alternative/intermediate to [[HTML]]. Its ultimate purpose is to be converted by [[wiki (software)|wiki software]] into [HTML](#), which in turn is served to [[web browser]]s.

There is no commonly accepted standard wikitext language yet. The grammar, structure, justification, keywords and so on depend on the particular wiki software used on the particular [[website]], such as [[MediaWiki]] on Wikipedia.

== External links ==

- * [http://www.mediawiki.org/wiki/Wikitext_standard www.mediawiki.org/wiki/Wikitext_standard]
- * [http://www.mediawiki.org/wiki/Alternative_parsers Alternative MediaWiki parsers]

Image from Wikimedia (CC BY-SA 3.0)

OCR OUTPUT (HOCR)

490

THE PASSIVE STEMS.

CH. XIX.

the *τυφλωθέν* of the M.SS. I have conjectured *τύφλωθεν*. We may quote also *ἐφίληθεν* (or *ἐφίλαθεν*) Theocr. vii. 60. For the shorter forms it is of importance to notice that they occur also on Doric inscriptions, where we may give them the Doric accentuation: *διελέγεν* C. I. G. 3050 l. 7, 3052 l. 10, for which in 3048, l. 8, certainly only from oversight, *διελέγην* has been written, which Boeckh with Buttmann alters into *διελέγεν*. *κατεδικάσθεν* Tab. Heracl. i. 122, 143, *διελέχθεν* treaty between the Cretan towns Hierapytna and Lyttus (Naber Mnemos. i. 105 l. 13). From

```
<div class="ocr_page" title="file /tmp/1KyOjfDG21/0001.bin.png">
<span class="ocr_line" title="bbox 227 3046 315 3086">490</span> <br />
<span class="ocr_line" title="bbox 787 3045 1285 3079">THE PASSIVE STEMS.</span> <br />
<span class="ocr_line" title="bbox 1680 3037 1832 3067">CH. IX.</span> <br />
<p></p>
<span class="ocr_line" title="bbox 228 2934 1830 2989">the τυφλωθέν of the M.SS. I have conjectured τύφλωθεν. We may quote</span> <br />
<span class="ocr_line" title="bbox 228 2882 1831 2937">also ἐφίληθεν (or ἐφίλαθεν) Theocr. vii. 60. For the shorter forms it ia</span> <br />
<span class="ocr_line" title="bbox 228 2832 1831 2883">of importance to notice that they occur also on Doric inscriptions, where</span> <br />
<span class="ocr_line" title="bbox 228 2780 1831 2835">we may give them the Doric accentuation: διελέγεν C. I. G. 3050 l. 1.,</span> <br />
<span class="ocr_line" title="bbox 231 2720 1831 2775">for which in 3048, l. 8, certainly only from oversight, διελέγην</span> <br />
<span class="ocr_line" title="bbox 229 2678 1828 2726">has been written, which Boeckh with Buttmann alters into διελέγεν.</span> <br />
<span class="ocr_line" title="bbox 230 2623 1828 2674">κατεδικάσθεν Tab. Heracl. i. 122, 143, διελέχθεν treaty between the Cretan towns</span> <br />
<span class="ocr_line" title="bbox 229 2565 1824 2621">towns Hierapytna and Lyttus (Naber Mnemos. i. 105 l. 13). From</span> <br />
```

Ocropolis OCR output from **Lace** project, Bruce Robertson, Federico Boschetti

ANNOTATING TEXTS

Leaf morphology

External leaf characteristics (such as shape, margin, hairs, etc.) are important for identifying plant species, and botanists have developed a rich terminology for describing leaf characteristics. These structures are a part of what makes leaves determinant; they grow and achieve a specific pattern and shape, then stop. Other plant parts like stems or roots are non-determinant, and will usually continue to grow as long as they have the resources to do so.

Basic leaf types

- Ferns have fronds
- Conifer leaves are typically needle-, awl-, or scale-shaped
- Angiosperm (flowering plant) leaves: the standard form includes stipules, a petiole, and a lamina
- Lycophytes have microphyll leaves.
- Sheath leaves (type found in most grasses)
- Other specialized leaves (such as those of Nepenthes)

Arrangement on the stem

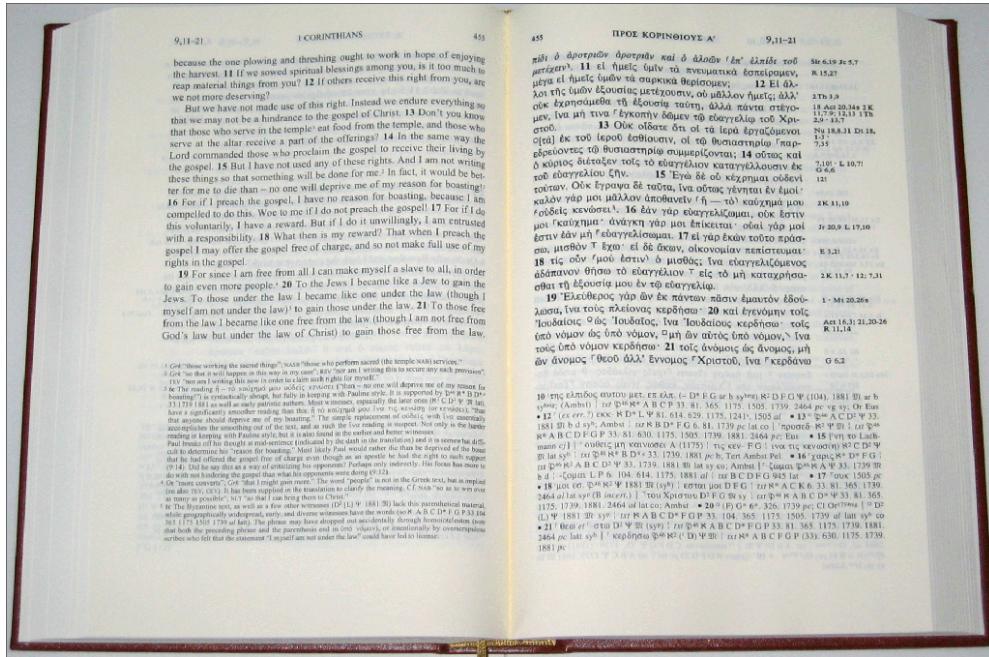
Different terms are usually used to describe leaf placement :

- **Alternate** — leaf attachments are singular at nodes, and leaves alternate direction, to a greater or lesser degree, along the stem.
- **Opposite** — leaf attachments are paired at each node; **decussate** if, as typical, each successive pair is rotated 90° progressing along the stem; or **distichous** if not rotated, but two-ranked (in the same geometric flat-plane).
- **Whorled** — three or more leaves attach at each point or node on the stem. As with opposite leaves, successive whorls may or may not be decussate, rotated by half the angle between the leaves in the whorl (i.e., successive whorls of three rotated 60°, whorls of four rotated 45°, etc). Opposite leaves may appear whorled near the tip of the stem.
- **Rosulate** — leaves form a rosette

The diagram illustrates various leaf features and arrangements. At the top right, a bracket labeled 'Used to identify leaves' groups 'Shape' (with a jagged outline), 'Margin' (with a wavy line), and 'Hairs' (with a magnified view of fine lines). Below this, a central green circle labeled 'Basic leaf types.' has arrows pointing to several examples: 'Needle-shaped.' (a long thin line), 'Fronds' (a large fern-like leaf), 'Sheath' (a long tube-like leaf), 'Microphyll' (a small circular leaf), 'Flowering plants.' (a branched leaf), and 'Other' (a simple lanceolate leaf). To the left of the main circle, there are two yellow arrows pointing to sketches of stem arrangements: one for 'Alternate' (leaves at different heights) and one for 'Opposite' (leaves in pairs at the same height). Further down, a yellow arrow points to a sketch of a 'Whorled' arrangement (three leaves at the same height), and another yellow arrow points to a sketch of a 'Rosulate' rosette (multiple leaves radiating from a central point).

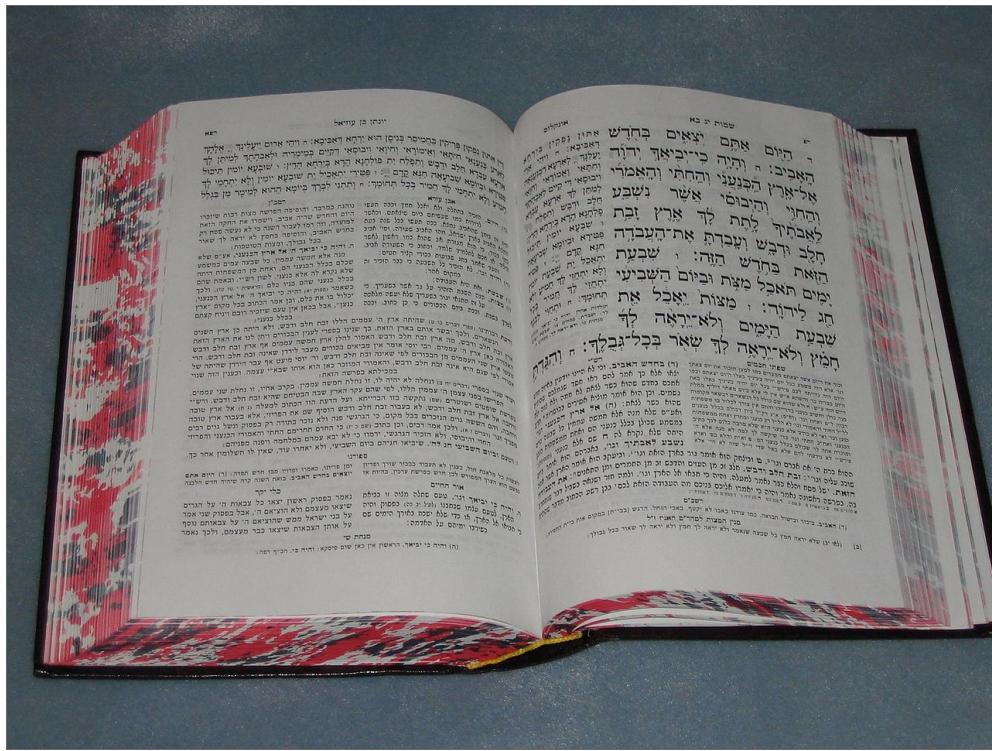
Image from Wikimedia (**CC BY-SA 3.0**)

ANNOTATING TEXTS



NET Diglot Greek New Testament

ANNOTATING TEXTS



Mikaoz Gedolot, image from [Wikipedia](#)

ANNOTATING TEXTS

Click on a word to bring up parses, dictionary entries, and frequency statistics

Hom. Od. 1.1

Notes (W. Walter Merry, James Riddell, D. B. Monro, 1886)

ένθρο μοι ἔνεπε μοῦσα, πολύτροπον, ὃς μάλα πολλὰ πλάγχθη ἐπεὶ τοῖς ἱερὸις ιππόλιθον ἐπερεψεν· πολλὰ δὲ, ἀνθρώποιν θεούς θυτα καὶ νόσον ἔγνω, πολλὰ δὲ, ὃς ἐν πόνοις τοῖς ἄλλος δικαίου πολλά, ἀμφίστεντος ἦν τε φυγὴ καὶ νόσος ἔταιρον· ὅλος δὲ, ὃς ἔπειτας ἐρύσσατο, περιπολέσσατο· αὐτούν γάρ σφετέρουν ἀπάσθατον δλοντο· γῆνιοι, οἱ κατὰ θοῦς Υπερβόντος Ήλείοιο ποθιον: αὐτάρ δ τοῖσιν ἀφείτετο νόστιμον ἡμαρ. τῶν ἀμοθεν γε, θεά, θυγατέρ διός, εἴπε καὶ ήμιν.

Ἔνθρο μέλοι μέλαπτες, δοσι φύγον αἰτὺν διελθοντο, οικοί έσαν, πολέμους τε περιεγούντος ήδη θάλασσαν· τὸν δὲ οἰον νόσοτον κεχρημένον ἡδε γυναικός νύμφη ποτνίης· ἔρυκε Καλυνώ διὺς θεῶν· ἐν σπέσαις γλωφυροῖς, λιλαιομένην ποσὶν εἶναι, ὅλη· ὅτε δὲ έτος ήλθε περιπολομένων ἑναυτῶν, τῷ οἱ ἐπεκλώσαντο θεοὶ οἰκονόδες νέσοθι· εἰς θιάκην, οοδό· ἔνθε περιγυμένος ἡγεν ἀσθόλων καὶ μετὰ οἰσι φύλαισι θεοὶ δέ ἐλέαιρον ἀπαντες νόσοφι Ποσειδάνων· ὃ δέ ἀπερχές μενέανεν ὑντιθέων θεούητη πάρος ἦν γαῖαν ικέσθαι.

ὅλλ' δέ μὲν Αιθίοπας μετεκάθετη τηλόθι ἔόντας, Αιθιόπας τα διχθι δεσδαίται, έχατοι ἄνδρων, οἱ μὲν δυσομένου Υπερίονος οἱ δέ ἀνινθος· δύντεν ταύρουν τε καὶ δρυειάς εκατερήθης· Ἐνθ' δέ ἐτέρετο δύο πατρομένος, οἱ δέ διή ἀλλοι Σηνός ἐν μεγάροις Οὐλακοῖς μέσοις ἤσθαι· τούτοις δέ τοισι δύος πετρῷοι άνδρους τε θεοὺς τε· μηνιστο τούτος γάρ κατέ θυμὸν αὐτούς μοις Αιγύδοιο· τὸν δέ Ἀγαμεμνονίδης τηλεκλυτός ἔκταν' Ὁρέστης· τὸν δέ γ' ἐπιμνηθεὶς ἐπεί αδαντοῖσι μετηύσα:

ένθρητες is the assimilated form of **ένθρητη** (from stem "**ΘΡΗΝ-**"), as the Aeolic aorist "**ένθρευτη**" stands for "**ένθρευτη**". We may compare the Lat. word *insecce*, which is actually used in the translation of this line by Livius Andronicus. "**Virum mili, Canevia, insecce verutum.**"

5 **μοι** is enclitic = "prihce"; as distinguished from the emphatic dative **εἰπε καὶ ημῖν** I. 10. For the order observed as in the plene and succession of Enclitics in Homeric Greek, see Monro, *Homeric Grammar, Appendix E.*

Μούρα = "Μούρια, Μούροι", from root "**ΜΕΥ-**", to think." In I.10 the muse is called "**Θύγατερ Διός**", as in II. 2. 491 "**Οὐλυμπίαδες Μούροι, Διός αιγάλειοι θυγατέρες**". They are represented as nine in Od. 24. 607, but their names are first given in Hesiod.

10 **πολύτροπος** 'of many devices, versatilis'. This epithet of Odysseus recurs only Od. 10. 330, but it has many equivalents in Il. and Od., e.g. **πολύμητις, πολύερων, πολυμήχανος, ποικιλόμητις**; the general sense of which seems to fit its meaning. Cf. the phrase by which Odysseus characterizes himself, Od. 9. 19 τούτῳ "**Οὐσορτος λαρετάδης δέ πτοι δόλαιον**".

15 **ἀνθρώπους μέλοι**: Nietzsche explains it as equivalent to "**πολύπλακτος**", and takes the words "**Ἐς μάλα πολλὰ πλάγηται**" as its epistemesis. Cf. Int. 300 πατροφούμενος... δέ οι πετέρα κάποιον Εκτός· Od. 18. 1 πτυσός πανδίμοιος δέ κατά δάστυ πτυχεύονται.. IL 5. 63. 9, 124. 11. 475. 12. 295. 13. 452, which suggests that the Homeric usage is, in some cases, to repeat some portion of the word, at least, in the exegetical clause. See *Lehrs. Rhein. Mus.* 1864, p. 303, and Nietzsche, *De Odyssee Exordia*, Hannov. 1824.

20 English (Samuel Butler, Based on public domain edition, revised by Timothy Power and Gregory Nagy, 1990) focus lead
English (1919) focus show
References (27 total) hide

25 • Commentary references to this page (7):
o Sappho, Carmen Omnia, 1
o Thomas W. Allen, E. E. Sieck, Commentary on the Homeric Hymns, ΗΛΜΗ ΤΟ ΑΡΗΙΟΤΗΤΟ
o W. Walter Merry, James Riddell, D. B. Monro, Commentary on the Odyssey (1886), 1.366
o W. Walter Merry, James Riddell, D. B. Monro, Commentary on the Odyssey (1886), 1.328
o Walter Leaf, Commentary on the Iliad (1900), 1.1
o W. Walter Merry, James Riddell, D. B. Monro, Commentary on the Iliad (1900), 2.484
o Thomas D. Seymour, Commentary on Homer's Iliad, Books I-III, 1.3
• Cross-references to this page (6):
o Aristotle, Rhetoric, Arbst. Rh. 3.14

30

Homer's Odyssey, from the **Perseus Project**

ANNOTATING TEXTS

```
<publicationStmt>
  <idno>JC</idno>
  <publisher>Folger Shakespeare Library</publisher>
  <address>
    <addrLine>201 East Capitol Street, SE</addrLine>
    <addrLine>Washington, DC 20003</addrLine>
    <addrLine>http://www.folgerdigitaltexts.org</addrLine>
    <addrLine>folgertexts@folger.edu</addrLine>
  </address>
  <availability>
    <licence target="http://creativecommons.org/licenses/by-nc/3.0/deed.en_US">Distributed
      under a Creative Commons Attribution-NonCommercial 3.0 Unported License</licence>
  </availability>
  <date>April, 2014</date>
</publicationStmt>
```

WHY XQUERY?

Abstract

XML is a versatile markup language, capable of labeling the information content of diverse data sources including structured and semi-structured documents, relational databases, and object repositories. A query language that uses the structure of XML intelligently can express queries across all these kinds of data, whether physically stored in XML or viewed as XML via middleware. This specification describes a query language called XQuery, which is designed to be broadly applicable across many types of XML data sources.

XQuery 1.0: An XML Query Language

WHAT CAN XQUERY QUERY?

- XML in documents or XML databases (e.g. eXist, BaseX, Marklogic, xDB)
- HTML (normalized using HTML Tidy, etc.)
- Word / Open Office documents (saved as XML)
- XML serialization of database data or programming objects
- XML views of relational data
- Coming soon - JSON Documents (XQuery 3.1)

PATH EXPRESSIONS

FIND THINGS IN XML DOCUMENTS

- /TEI/teiHeader
- //person
- //person[persName]
- //tagUsage
- //tagUsage[@gi="stage"]
- //sp[@who = '#Caesar_JC']

FLWOR EXPRESSIONS

- Acronym: For / Let / Where / Return
- Iteration
- Binding variables to intermediate results
- Identifying related data ("joins")
- Restructuring

EXAMPLE: FLWOR EXPRESSIONS

"SUMMARIZE THE SPEECHES FOR EACH PERSON"

```
declare default element namespace "http://www.tei-c.org/ns/1.0";

for $person in //person[persName]
let $id := $person/@xml:id
let $idref := concat('#',$id) (: bad Folger convention :)
let $speeches := //sp[@who=$idref]/ab
let $words := $speeches//w
let $nwords := count($words)
order by $nwords descending, $nspeeches descending
return
    <person xml:id="{{$id}}"
        speeches="{{$nspeeches}}"
        words="{{$nwords}}"/>
```

XML CONSTRUCTORS

- Create any XML instance
- Literal syntax: same as XML

```
<person
  xml:id="Brutus_JC"
  speeches="193"
  words="5475">
Farewell, good Strato. Brutus runs on his swor
Caesar, now be still. I killed not thee with t
so good a will.
</person>
```

- {} lets you add expressions

```
<person xml:id="${id}"
  speeches="${nspeeches}"
  words="${nwords}">
{
  normalize-space(string($speeches[last()]))
}
</person>
```

EXAMPLE: SPEAKERS AND THEIR LINES

```
declare default element namespace "http://www.tei-c.org/ns/1.0";

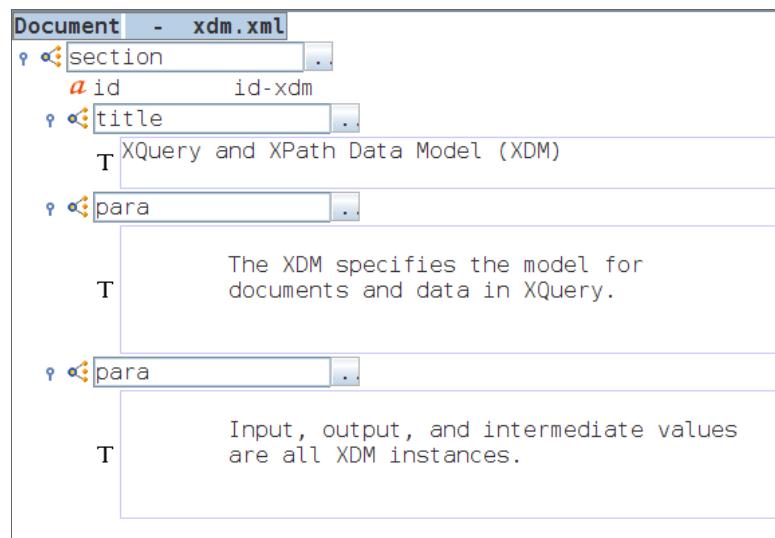
><personae>
> {
    for $person in //person[persName]
    let $name := $person/persName/name
    let $id := $person/@xml:id
    let $idref := concat('#',$id) (: bad Folger convention :)
    let $speeches := //sp[@who=$idref]/ab
    let $lines := //sp[@who=$idref]/ab//lb
    let $words := $speeches//w
    let $nspeeches := count($speeches)
    let $nlines := count($lines)
    let $nwords := count($words)
    order by $nwords descending, $nlines, $nspeeches descending
    return
        <person xml:id="{{$id}}"
            speeches="{{$nspeeches}}"
            lines="{{$nlines}}"
            words="{{$nwords}}>
    {
        $name
    }
    </person>
}
</personae>
```

WHY XQUERY?

- Find things in marked up data
- Explore relationships among marked up data
- Find related resources
- Create new markup that explores new representations and relationships
- Support close reading by answering questions as you read
- **Distant reading?**
- Convert to HTML and other formats
(XSLT is very good at this)

XML AND XPATH DATA MODEL (XDM)

```
<section id="id-xdm">
  <title>XQuery and XPath Data Model (XDM)</title>
  <para>
    The XDM specifies the model for
    documents and data in XQuery.
  </para>
  <para>
    Input, output, and intermediate values
    are all XDM instances.
  </para>
</section>
```



PATH EXPRESSIONS

XQUERYINSTITUTE.ORG

Jonathan.Robie @ ibiblio.org / @jonathan_robie

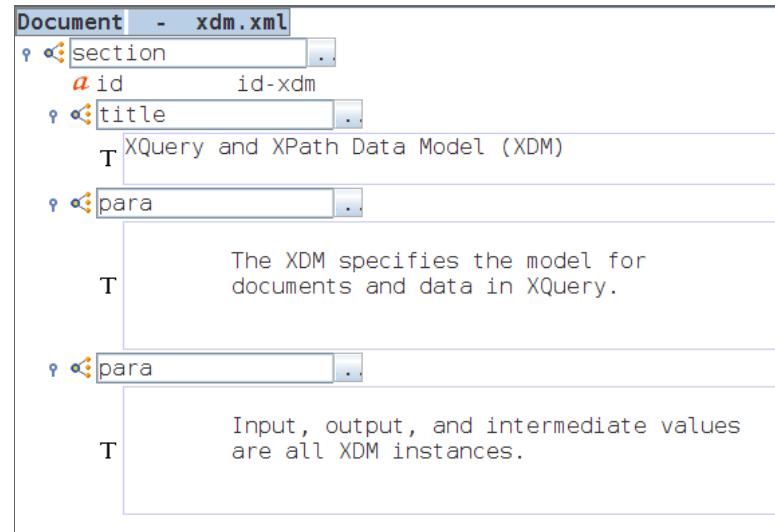
PATH EXPRESSIONS

FIND THINGS IN XML DOCUMENTS

- /TEI/teiHeader
- //person
- //sp[@who = '#Caesar_JC']

XML AND XPATH DATA MODEL (XDM)

```
<section id="id-xdm">
  <title>XQuery and XPath Data Model (XDM)</title>
  <para>
    The XDM specifies the model for
    documents and data in XQuery.
  </para>
  <para>
    Input, output, and intermediate values
    are all XDM instances.
  </para>
</section>
```



KINDS OF NODES

- Document Nodes
- Element Nodes
- Attribute Nodes
- Text Nodes
- Comment Nodes
- Namespace Nodes
- Processing Instruction Nodes (rarely used today ...)

KINDS OF NODES

```
<?xml version="1.0" encoding="utf-8"?>
<!!-- The xmlstylesheet is a processing instruction. -->
<?xml-stylesheet type="text/xsl" href="fdt.xsl"?>
<TEI xmlns="http://www.tei-c.org/ns/1.0">
  <teiHeader>
    <fileDesc>
      <titleStmt>
        <title>Julius Caesar</title>
        <author>William Shakespeare</author>
        <editor xml:id="BAM">Barbara A. Mowat</editor>
        <editor xml:id="PW">Paul Werstine</editor>
        <respStmt>
          <resp>Edited for XML and encoded by</resp>
          <persName xml:id="MSP">Michael Poston</persName>
          <persName xml:id="RLN">Rebecca Niles</persName>
        </respStmt>
      </titleStmt>
      <editionStmt>
        <edition n="0.9.0">Text released without textual notes</edition>
      </editionStmt>
      <publicationStmt>
        <idno>JC</idno>
        <publisher>Folger Shakespeare Library</publisher>
        <address>
          <addrLine>201 East Capitol Street, SE</addrLine>
```

PATH EXPRESSION BASICS

LOCATING NODES IN TREES

- Paths may start with "/" or "//" (or any other expression)

Leading "/" selects document root

Leading "//" selects document root and all descendants

- Name Tests

Elements by name: /TEI, //speaker

Any element: /*, ///*

Attributes by name: //@who

Any attribute: //@*

- Steps are separated by "/" or "//"

Example: //sp/@who

Example: /TEI//speaker/w

EXPLORING A DOCUMENT WITH XPATH

- Select the root node
- Select the author
- Select all elements, at any level
- Select all attributes, at any level
- Find the "tagsDecl" element - what is it for?
- Find the elements that describe what the "lb" and "ab" elements are for
- Find all the "lb" elements, all the "ab" elements

NODE TESTS (KIND OF NODE)

- `document-node()`
- `element()`
- `element(stage)`
- `attribute()`
- `attribute(who)`
- `text()`
- `comment()`
- `namespace-node()`
- `processing-instruction()`

HEADS UP! DOCUMENT NODES!

What's the difference between this query ...

```
tei-doc.xml:  
<TEI xmlns="http://www.tei-c.org/ns/1.0">  
  <teiHeader/>  
</TEI>;  
  
query:  
  
document("tei-doc.xml")/*
```

... and this one?

```
declare variable $tei :=  
  <TEI xmlns="http://www.tei-c.org/ns/1.0">  
    <teiHeader/>  
  </TEI>;  
  
$tei/*
```

HEADS UP! DOCUMENT NODES!

hint ...

```
declare variable $tei :=  
  document {  
    <TEI xmlns="http://www.tei-c.org/ns/1.0">  
      <teiHeader/>  
    </TEI>  
  };  
$tei/*
```

HEADS UP! NAMESPACES!

What's the difference between this query ...

```
tei-doc.xml:  
  
<TEI xmlns="http://www.tei-c.org/ns/1.0">  
  <teiHeader/>  
</TEI>;  
  
query:  
  
document("tei-doc.xml")/TEI
```

... and this one?

```
query:  
  
declare default element namespace "http://www.tei-c.org/ns/1.0";  
document("tei-doc.xml")/TEI
```

FILTER EXPRESSIONS - SPEECHES

```
//sp[@who="#Caesar_JC"]  
//sp[@who="#Caesar_JC" or @who="#Brutus_JC"]  
//sp[@who="#Caesar_JC" and ./ab//w="Brutus"]  
//sp[@who="#Brutus_JC" and ./ab//w="Caesar"]  
//sp[@who="#Caesar_JC" and ./ab//w="Caesar"]  
//sp[@who="#Brutus_JC" and ./ab//w="Brutus"]  
//sp[@who="#Brutus_JC" and (.//ab//w="Brutus" or ./ab//w="Caesar")]  
//sp[@who="#Brutus_JC" and (.//ab//w="Brutus" and ./ab//w="Caesar")]  
//sp[@who="#Brutus_JC" and (.//ab//w="Caesar" and not(.//ab//w="Brutus"))]
```

FILTER EXPRESSIONS - STAGE DIRECTIONS

```
//stage[@type="entrance"]  
//stage[empty(@who)]  
//stage[@type="entrance" and empty(@who)]  
//stage[@type="entrance" and empty(*)]  
//stage[@type="entrance"]/*  
//stage[empty(*)]  
distinct-values(//stage/@type)  
//stage[@type="delivery"]
```

FILTER EXPRESSIONS - POSITION

```
//sp[1]
(//sp)[1]
//sp[last()]
(//sp)[last()]
//sp[position() = 1 to 3]
(//sp)[position() = 1 to 3]
(25 to 29)[3]
(//sp[@who="#Caesar_JC"])[1]
//sp[@who="#Caesar_JC"][1]
(//sp[@who="#Caesar_JC"])[last()]
//sp[@who="#Caesar_JC"][last()]
```

GENERAL COMPARISONS

- `5 = (1, 3, 5, 7, 9)`
- `(1, 3, 5, 7, 9) = 5`
- `(1, 2, 3) = (1, 3, 5)`
- `//sp[@who = "#Caesar_JC"]`
- `//sp[@who != "#Caesar_JC"]`
- `//ab//w[position() < 5]`
- `//ab//w[position() <= 5]`
- `//ab//w[position() = 1 to 5]`
- `//ab//w[position() = (1, 3, 5)]`
- `//ab//w[position() >= 5]`

PRECEDENCE AND COMPARISONS

- Puzzle: why is this an error?

```
//stage[@type="entrance", "exit"]
```

- Ask XQuery what this means:

```
"entrance" = "entrance", "exit"
```

- How can we correct the first expression?

- Ask XQuery what this means:

```
boolean("entrance" = "entrance", "exit")
```

EFFECTIVE BOOLEAN VALUE

- Many expressions test if a condition is `true` or `false`. These are "Boolean Values".

Effective Boolean Value of expression E = `boolean(E)`

- `boolean(true())`
- `boolean(false())`
- `boolean(0)`
- `boolean(1)`
- `boolean(one or more nodes)`
- `boolean(())`
- `boolean("non-empty string")`
- `boolean("")`

VALUE COMPARISONS

COMPARING SINGLE VALUES

- //sp[@who eq "#Caesar_JC"]
- //sp[@who ne "#Caesar_JC"]
- //ab//w[position() lt 5]
- //ab//w[position() le 5]
- //ab//w[position() gt 5]
- //ab//w[position() ge 5]

AGGREGATE FUNCTIONS

- count()
- avg()
- min()
- max()
- sum()
- //ab[count(.//w) > 100]
- max(//ab/count(.//w))
- sum(//ab/count(.//w))
- //ab[count(.//w) eq max(//ab/count(.//w))]

AXES: UP AND DOWN

- `self::*`
- `attribute::*`
- `child::*`
- `descendant::*`
- `parent::*`
- `ancestor::*`

AXES: FORWARD AND BACK

- `following::*`
- `following-sibling::*`
- `preceding::*`
- `preceding-sibling::*`

AXES: UP AND DOWN

```
(//stage)[1]
(//stage)[1]/
(//stage)[1]/self::*
(//stage)[1]/child::*
(//stage)[1]/child::node()
(//stage)[1]/attribute::*
(//stage)[1]/descendant::*
(//stage)[1]/descendant::*[4]
(//stage)[1]/descendant::*[position() = 2 to 5]
//ab//w[.= "Greek"]/ancestor::sp
//ab//w[.= "Greek"]/parent::*
```

AXES: FORWARD AND BACK

```
(//stage)[1]
(//stage)[1]/
(//stage)[1]/self::node()
(//stage)[1]/child::node()
(//stage)[1]/attribute::*
(//stage)[1]/descendant::node()
//ab//w[.= "Greek"]/ancestor::sp
//ab//w[.= "Greek"]/parent::*
```

ALL FORWARD AXES

- `self::*`
- `child::*`
- `attribute::*`
- `descendant::*`
- `descendant-or-self::*`
- `following-sibling::*`
- `following::*`
- `following-sibling::*`

ALL REVERSE AXES

- parent::*
- ancestor::*
- ancestor-or-self::*
- preceding::*
- preceding-sibling::*
- descendant::*
- descendant-or-self::*
- following::*
- following-sibling::*

FLWOR EXPRESSIONS

XQUERYINSTITUTE.ORG

Jonathan.Robie @ ibiblio.org / @jonathan_robie

WHAT ARE FLWOR EXPRESSIONS FOR?

- Swiss army knife for manipulating XML structures
- Iterate over sequences
- Join sequences (e.g. "Speeches by each character")
- Grouping
- Windowing

FOR CLAUSES - ITERATION

```
declare default element namespace "http://www.tei-c.org/ns/1.0";
for $sp in //sp
return $sp
```

USING ITERATION TO RESTRUCTURE

```
declare default element namespace "http://www.tei-c.org/ns/1.0";
for $sp in //sp
return
<speech>
  <speaker>{ $sp/@who }</speaker>
  {
    for $ab in $sp/ab
    return
      <line>{ normalize-space(string($ab))}</line>
  }
</speech>
```

USING ITERATION TO RESTRUCTURE

```
declare default element namespace "http://www.tei-c.org/ns/1.0";

for $person in //person[persName]
let $id := $person/@xml:id
let $idref := concat('#',$id)  (: bad Folger convention :)
let $speeches := //sp[@who=$idref]/ab
let $words := $speeches//w
let $nspeeches := count($speeches)
let $nwords := count($words)
order by $nwords descending, $nspeeches descending
return
    <person xml:id="{{$id}}"
        speeches="{{$nspeeches}}"
        words="{{$nwords}}>
    {
        normalize-space(string($speeches[last()]))
    }
</person>
```

TUPLES

A tuple is like a row in a table

A tuple stream:

```
($x = 1003, $y = "Fred", $z = <age>21</age>)
($x = 1017, $y = "Mary", $z = <age>35</age>)
($x = 1020, $y = "Bill", $z = <age>18</age>)
($x = 1024, $y = "John", $z = <age>29</age>)
```

TUPLES

Each clause (before return) generates a tuple stream
for clauses

```
for $i in (1, 2, 3)
return <tuple>{ $i }</tuple>
```

```
for $i in (1, 2, 3)
for $j in (3, 4, 5)
return <tuple>{ $i, $j }</tuple>
```

where clause

```
for $i in (1, 2, 3)
for $j in (3, 4, 5)
where $i eq $j
return <tuple>{ $i, $j }</tuple>
```

TUPLES

Each clause (before return) generates a tuple stream

let clauses

```
let $i := (1, 2, 3)
return <tuple>{ $i }</tuple>
```

```
for $i in (1, 2, 3)
let $j := (3, 4, 5)
return <tuple>{ $i, $j }</tuple>
```

```
let $i := (1, 2, 3)
let $j := (3, 4, 5)
return <tuple>{ $i, $j }</tuple>
```

TUPLES

Each clause (before return) generates a tuple stream
order by clauses

```
for $i in (1, 2, 3)
let $j := (3, 4, 5)
order by $i descending
return <tuple>{ $i, $j }</tuple>
```

TUPLES

The return clause creates a result, using variable bindings from the tuple stream
return clause

```
declare default element namespace "http://www.tei-c.org/ns/1.0";

for $person in //person[persName]
let $id := $person/@xml:id
let $idref := concat('#',$id) (: bad Folger convention :)
let $speeches := //sp[@who=$idref]/ab
let $words := $speeches//w
let $nspeeches := count($speeches)
let $nwords := count($words)
order by $nwords descending, $nspeeches descending
return
  <person xml:id="{{$id}}"
    speeches="{{$nspeeches}}"
    words="{{$nwords}}>
  {
    normalize-space(string($speeches[last())))
  }
</person>
```

COUNT CLAUSE

```
for $speech in //sp
for $speaker in $speech/speaker
count $c
return
<speech>
  <speaker>
    {
      $speech/@who,
      $speaker/w/text()
    }
  </speaker>
  {
    for $block in $speech/ab
    return
      <line xml:id="line-{$c}">{ normalize-space(string($block))}</line>
  }
</speech>
```

GROUPING

```
for $speech in //sp
let $who := $speech/@who
group by $who
order by $who
return
  <person who="{$who}">
    <speeches>
      {
        for $s in $speech
        return
          <speech>{ $s/@xml:id }</speech>
      }
    </speeches>
  </person>
```

PRE-GROUPING TUPLES

```
declare default element namespace "http://www.tei-c.org/ns/1.0";
for $speech in //sp
let $who := $speech/@who
return
  <tuple><speech>{ $speech }</speech><who>{ $who }</who></tuple>
```

POST-GROUPING TUPLES

```
declare default element namespace "http://www.tei-c.org/ns/1.0";
for $speech in //sp
let $who := $speech/@who
group by $who
return
<tuple><who>{ $who }</who><speech>{ $speech }</speech></tuple>
```

- Each grouping variable has a single value
- Each group corresponds to a unique set of values for the grouping variables
- Each non-grouping variable is bound to all instances assigned to the group

POST-GROUPING TUPLES

```
declare default element namespace "http://www.tei-c.org/ns/1.0";

for $speech in //sp
let $who := $speech/@who
count $pregroup-count
group by $who
order by $who
count $postgroup-count
return
<person who="{$who}" count="{$postgroup-count}" pre="{$pregroup-count}">
<speeches>
{
  for $s in $speech
  return
    <speech>{ $s/@xml:id }</speech>
}
</speeches>
</person>
```

A SIMPLE CONCORDANCE

```
declare default element namespace "http://www.tei-c.org/ns/1.0";
for $w in //sp//ab//w/lower-case(.)
group by $w
order by $w
return
<word w="{$w}" />
```

CONCORDANCE IN WELL-FORMED XML

```
declare default element namespace "http://www.tei-c.org/ns/1.0";  
  
<lexicon>  
{  
    for $w in //sp//ab//w/lower-case(.)  
    group by $w  
    order by $w  
    return  
        <word w="{$w}" />  
}</lexicon>
```

CONCORDANCE WITH FREQUENCY

```
declare default element namespace "http://www.tei-c.org/ns/1.0";  
  
<lexicon>  
{  
    for $w in //sp//ab//w/lower-case(.)  
    count $occurrence  
    group by $w  
    order by $w  
    count $word-number  
    return  
        <word n="{{$word-number}" w="{{$w}}" freq="{{$occurrence}}"/>  
}  
</lexicon>
```

WHO USES WHAT WORDS? (FLAT)

```
declare default element namespace "http://www.tei-c.org/ns/1.0";  
  
<lexicon>  
{  
    for $sp in //sp  
    for $who in $sp/@who  
    for $w in $sp//ab//w/lower-case(.)  
    count $occurrence  
    group by $who, $w  
    order by $w, $who  
    return  
        <word who ="{$who}" w ="{$w}" freq ="{count($occurrence)}" />  
}  
</lexicon>
```

WHO USES WHAT WORDS? (NESTED)

```
declare default element namespace "http://www.tei-c.org/ns/1.0";  
  
<lexicon>  
{  
    for $sp in //sp  
    for $who in $sp/@who  
    for $w in $sp//ab//w/lower-case(.)  
    count $occurrence  
    group by $w  
    order by $w  
    return  
        <word w="{$w}" freq="{$occurrence}">  
            {  
                for $gwho in $who  
                group by $gwho  
                order by $gwho  
                return  
                    <who name="{$gwho}" />  
            }  
        </word>  
    }  
</lexicon>
```

FUNCTIONS

```
declare function local:lexicon($folger-play) as element(lexicon)
{
  <lexicon>
  {
    for $sp in $folger-play//sp
    for $w in $sp//ab//w/lower-case(.)
    count $occurrence
    group by $w
    order by $w
    return
      <word w="{{$w}} freq="{{$fn:count($occurrence)}}/>
  }
}</lexicon>
};

local:lexicon()
```