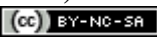




[newtFire {dhlds}](#)

Authored by: Alexandra Krongel (alk157 at pitt.edu) Edited and maintained by: Elisa E. Beshero-Bondar (ebb8 at pitt.edu)  Last modified: Tuesday, 10-Oct-2017 15:30:24 EDT. [Powered by firebellies](#).

---

## XPath Exercise 4

You can find an XML (TEI) version of our Fall 2015 DH class syllabus at <http://newtfire.org/dh/dhCDA-2015.xml>. Right-click to download and save this file locally on your computer, and open it in <oXygen/>.

You should consult [The XPath Functions We Use Most](#) page and especially its section III. on Strings. Also, if you have the Michael Kay text, it may be useful to you here. As always, consult our class notes and our introductory guide [Follow the XPath!](#). After you've completed your homework, save your answers to a file and upload it to CourseWeb as an attachment. (Please use an attachment! If you paste your answer into the text box, CourseWeb may munch the angle brackets.) Some of these tasks are thought-provoking, and even difficult. If you get stuck, do the best you can, and if you can't get a working answer, give the answers you tried and explain where they failed to get the results you wanted. Sometimes doing that will help you figure out what's wrong, and even when it doesn't, it will help us identify the difficult moments. These tasks require the use of path expressions, predicates, and functions, and in this exercise we concentrate on functions that manipulate strings. There may be more than one possible answer.

Using the syllabus XML document and the XPath browser window in <oXygen/>, construct XPath expressions that will do the following (give the full XPath expressions in your answers, and not just the results):

1. There are two books referenced in the syllabus using the tag <bibl>. What XPath will return a semicolon-separated list of the authors?
2.
  1. Which 'div' elements contain references to 'homework'? How many results do you return?
  2. Can you figure out how to retrieve the immediate parent element containing the word "homework"? (Hint: it involves looking for any element below the div and then its text() node, since we need the element whose \*text\* contains the word "homework.")
3.
  1. What XPath returns all the Fridays on the syllabus? (Scroll through the document looking for the date elements to help determine this.)
  2. Now, what if we want to return those dates in their ISO format, as yyyy-mm-dd? Can you retrieve that with XPath? (Hint: Look at the attribute values on the date elements.)
  3. Return a string-joined list of all these dates, separated with a comma and a space.
4.
  1. How many div elements of @type 'assign' contain references to word "GitHub"?
  2. Find the longest and shortest div elements of this type (that contain the word "GitHub") in the document. How long and short are they? Hint: You will need to use min(), max(), and the string-length() function here, as well as some complex predicates.

5. **Reformatting Dates:** In Question 3, you located the Fridays on the syllabus, most likely through XPath to reach element text contents. What if you only worked with the @when attribute values on <date>, without looking at the text element content? Could you determine the Fridays on the syllabus just from that content alone? Yes, you can, and though this may not seem practical since we already have days of the week noted in our document, consider this a challenge, *as if those M, W, and F designations were missing*, or as if you had to work with a list of numerical dates without knowing their days of the week. To retrieve this, you need an XPath function you probably have not seen before: `format-date()`, to use with `xs:date`. You can read about these in the Michael Kay book on pages 781-788, or on [the W3C specifications page](#) for XSLT functions, as well as in [the Safari XSLT book online](#).

The function `format-date()` can take a string of text that it identifies as a date and can be set to reformat that date in many different ways: It could give the name of the month, the year as the phrase "Two Thousand and Fifteen", and, yes, the day of the week to accompany a day, output in upper or lower case letters as you wish. It might also be converted and expressed as a Buddhist, Mohammedan, or Japanese calendar date (among many others).

For this last task, do some reading on `format-date` in one or more of the resources we've linked here. Then try the following:

1. Working **only** with the @when attribute on the date elements on our syllabus file, convert those dates in the return window so that they display days of the week (and anything else you wish from the various available date-formatting codes. To do this, you'll need to see how to work with `format-date()`. It works with at minimum, two arguments (though it can take up to five). At minimum it needs to contain 1) something that it understands **as a date**, and 2) a **picture string** to designate (or "picture" if you will) the output, using a special notation of letter codes inside square brackets, examples of which you can look up in the references we've mentioned and linked here. Here is a model of how `format-date` looks with the minimum two arguments, working on a string of text marked as a dot: `.` (to indicate the self:: axis—the current context node in an XPath expression):  

```
//some-XPath-here-that-leads-to-the-dates-you-want/format-date(xs:date(.), '[FNn]/[MNn]/[Dwo]/[YWw]')
```

The first argument of `format-date()` takes the current XPath date (each date attribute on the syllabus) and recognizes it as a date via `xs:date(.)`. Then, the second argument (after the comma) sits inside the single quotation marks. The values inside `[]`s or square brackets are the picture strings which designate FNn (day of the week—initially capitalized to lower case, then Month (same thing with the capitalization), then a day of the month spelled out as a word, and finally the year converted from a number to a string of words. We have positioned a `/` in between each picture string as a separator. Try appending the `format-date()` code from this example into your XPath that reaches into the @when attribute on date, plug it into your XPath window and notice what it outputs in the return window. And try tinkering with it to change it, using different picture strings.  
**Record at least two different ways you adjusted this code to output different formats of date that you tried here, and their output.**
2. Now that you see how picture strings work in the `format-date()` function, we think you now know enough to take a numerical string of text from the @when attribute, convert it to retrieve days of the week, and then output **only the Friday dates on the syllabus**. We don't care how you decide to format the rest of the date, as that is up to you, but we would like you to write an XPath that first filters to find the dates you need, and then outputs those dates however you wish to output them. **Hints:** You will need to use a predicate, and you want to put the `contains()` or `matches()` function around the `format-date()` function, after the part of the XPath expression where you drill down for dates.

3. One last challenge: Modifying your functions, can you output **all of the Friday dates in October only**? Record the XPath. Hint: You probably want multiple predicates for this.