

QT4 Status Update

Norm Tovey-Walsh

Saxonica

02 Nov 2023

What?

QT4CG is the informal name of the W3C *XSLT Extensions Community Group*. It has:

Coincidentally, we will have had 52 meetings just before Declarative Amsterdam (if things go according to plan after this was recorded). That's a calendar year of meetings, spread over a bit more than a calendar year of actual time.

- a community group page.
- weekly meetings with agendas and minutes.
- a website, <https://qt4cg.org/>
- a mailing list, `public-xslt-40@w3.org` with archives.
- a repository, <https://github.com/qt4cg/qtspecs/> with issues, pull requests, and working drafts.
- a channel (`#xpath-ng`) on the XML.com Slack.

Why?

The 3.0/3.1 specifications were published in 2017. Not long after, there were discussions about extensions that users would like to see. In 2020 the extensions group was formed. In 2022, we began regular meetings.

Some of the recent changes have been to support features that are popular in other programming languages.

- *XML Path Language (XPath) 4.0*
- *XPath and XQuery Functions and Operators 4.0*
- *XQuery 4.0: An XML Query Language*
- *XQuery and XPath Data Model 4.0*
- *XSLT Transformations (XSLT) Version 4.0*
- *XSLT and XQuery Serialization 4.0*

The W3C has ended the XML Activity that sponsored the original work on XML Query and Transformation standards, but there is still work to do. User requirements are ever evolving, technology is ever evolving, no family of specifications as large as QT is ever “complete”, there’s always more that could be done.

Who?

The (current) regular participants at the weekly meetings are:

- Reece Dunn
- Sasha Firsov
- Christian Grün
- Joel Kalvesmaki
- Michael Kay
- John Lumley
- Dimitre Novatchev
- Wendell Piez
- Ed Porter
- C. M. Sperberg-McQueen, *co-chair*
- Norm Tovey-Walsh, *co-chair*

There are additional participants writing issues and comments, participating in discussions on the mailing list, chatting in Slack, etc. All are invited!

What's new in 4.0?

There are more new things than I could enumerate in 15 minutes. Way more things. There's a changelog in each specification.

I've cherry-picked some things I thought would be of interest. No offense intended if I've left out your favorite.

They're intentionally in a random sort of order to keep things interesting.

A lot of things!

Each specification has a "changes since 3.1 appendix" with a (usually, mostly) up-to-date list of all the changes.

The following slides list a few cherry-picked examples, they are not even vaguely comprehensive.

Optional and keyword arguments

XQuery and XSLT have had functions for a long time. And different function arities have been possible. Two popular missing features are default values (allowing default arguments) and keyword arguments.

```
declare function ex:point(  
  $x as xs:integer := 0,  
  $y as xs:integer := 0  
) {  
  <point x="{ $x } " y="{ $y } " />  
};
```

((ex:point(), ex:point(1), ex:point(y := 2)) returns

```
<point x="0 " y="0 " />  
<point x="1 " y="0 " />  
<point x="0 " y="2 " />
```

Function parameters, keyword arguments.

Alternatives in steps and tests

This is a simple syntactic convenience when writing node and axis step tests.

The `NodeTest` in an `AxisStep` allows alternatives:

```
ancestor::(db:section|db:appendix)
```

Instead of

```
ancestor::db:section|ancestor::db:appendix
```

Element and attribute tests can include alternative names:

```
element(db:chapter|db:section) or attribute(role|xlink:role).
```

String interpolation with backticks

XPath already includes functions and operators to construct string values, but string interpolation has proven very convenient in other languages.

```
let $name := "Norm"  
let $org := "Saxonica"  
return  
  `${$name} works for {$org}`
```

returns “Norm works for Saxonica”.

Mapping arrow operator

- The (existing) `=>` applies a function to an expression, using the value as the first argument to the function.
- The (new) `=!>` applies a function to each item in a sequence (think `=>` plus `!`).

These can be freely mixed:

```
"This is November"  
=> tokenize() =!> concat(".") =!> upper-case() => string-join("_")
```

returns

```
THIS._IS._NOVEMBER.
```

Attribute forms for if/when/otherwise

A syntactic convenience for conditionals in XSLT:

```
<p>
  <xsl:text>Today is an </xsl:text>
  <xsl:if test="day-from-date(current-date()) mod 2 = 0"
    then=" 'even' "
    else=" 'odd' "/>
  <xsl:text> day.</xsl:text>
</p>
```

Similarly, `select` is allowed on `xsl:when` and `xsl:otherwise`.

An xsl:switch instruction

Can be used with single values, as a lookup table, but also allows sequences of values:

```
<xsl:variable name="days" as="xs:integer">
  <xsl:switch select="$month">
    <xsl:when test="1, 3, 5, 7, 8, 10, 12" select="31"/>
    <xsl:when test="4, 6, 9, 11" select="30"/>
    <xsl:when test="2">
      <xsl:if test="$leap-year" then="29" else="28"/>
    </xsl:when>
  </xsl:switch>
</xsl:variable>
<p>There are {$days} days in this month.</p>
```

Enclosing modes

Group templates together in a mode.

```
<xsl:mode name="m:copyright">  
  <xsl:template match="db:year">...</xsl:template>  
  <xsl:template match="db:holder">...</xsl:template>  
</xsl:mode>
```

Escape-solidus for JSON serialization

```
<xsl:output method="json" />
<xsl:template name="xsl:initial-template">
  <xsl:sequence select="map{ '/path/to/file': 'text' }" />
</xsl:template>
```

Produces “`{"\/path\/to\/file":"text"}`”. 🎵

```
<xsl:output method="json" escape-solidus="false" />
<xsl:template name="xsl:initial-template">
  <xsl:sequence select="map{ '/path/to/file': 'text' }" />
</xsl:template>
```


Produces “`"/path/to/file":"text"}`”. 🎉

An otherwise operator

```
$optional otherwise $default
```

instead of

```
if (exists($optional)) then $optional else $default
```

 otherwise expression.

And also

- Context item generalized to context value
- Braced conditionals, optional else clause
- Abbreviated function syntax
- Arity-one focus functions
- Compact lookup syntax
- ...

Functions. Lots of functions.

`fn:all-different`, `fn:all-equal`, `fn:atomic-equal`, `fn:build-uri`, `fn:char`, `fn:characters`, `fn:contains-sequence`, `fn:decode-from-uri`, `fn:duplicate-values`, `fn:ends-with-sequence`, `fn:every`, `fn:expanded-QName`, `fn:foot`, `fn:highest`, `fn:identity`, `fn:in-scope-namespaces`, `fn:index-where`, `fn:interperse`, `fn:is-NaN`, `fn:items-after`, `fn:items-before`, `fn:items-ending-where`, `fn:items-starting-where`, `fn:iterate-while`, `fn:log`, `fn:lowest`, `fn:op`, `fn:parse-integer`, `fn:parse-QName`, `fn:parse-uri`, `fn:partition`, `fn:replicate`, `fn:slice`, `fn:some`, `fn:starts-with-sequence`, `fn:transitive-closure`, `fn:trunk`, `fn:void`, `fn:xdm-to-json`, `array:build`, `array:foot`, `array:members`, `array:of-members`, `array:slice`, `array:split`, `array:trunk`, `map:build`, `map:filter`, `map:of-pairs`, `map:pair`.

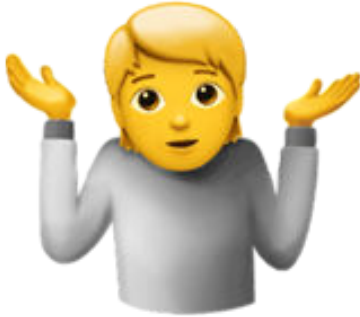
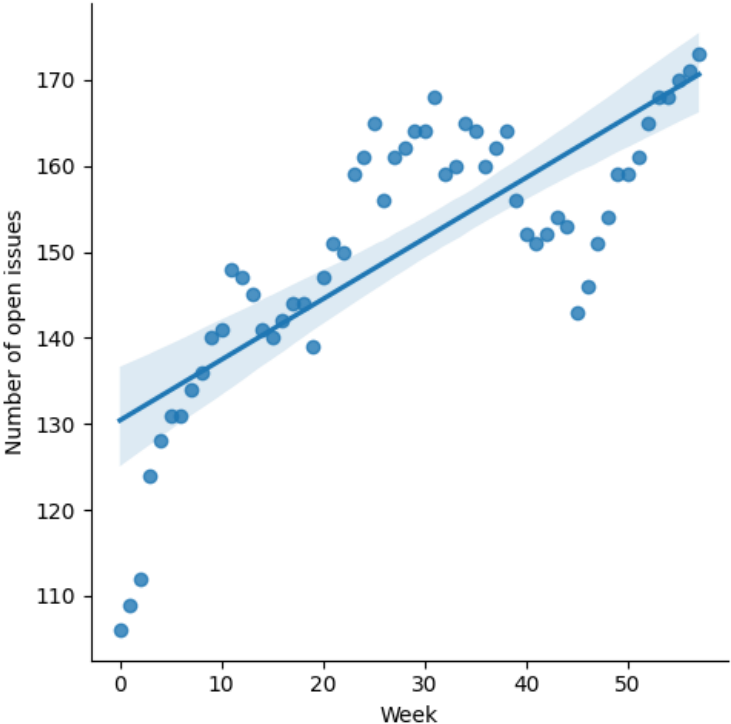
And maybe

- FLOWR expression extension for maps and arrays?
- Variadic function parameters?
- Destructuring assignments?
- Improved map/array navigation?
- A set data type?

There are 171 open issues and/or feature requests at the time of this writing.

When?

“Burndown” chart of open issues:



Thank you

- See also: the community group page.
- The website, <https://qt4cg.org/>
- The mailing list, `public-xslt-40@w3.org`.
- The repository, <https://github.com/qt4cg/qtspecs/>.
- The `#xpath-ng` channel on the XML.com Slack.

(And especially thanks to everyone on the community group and in the wider community devoting their valuable time and energy writing, reviewing, testing, and implementing QT4!)