XQuery equivalents of Oxygen Author's default actions and operations

Functions and variables in Oxygen namespace
Correspondences between Oxygen Author Operations and XQuery Update operations
Correspondences between XQuery Update and Oxygen Operations operations
5.1 Insert
5.3.1 Replacing a Node
5.3.2 Replacing the Value of a Node
Example XQuery Script
References

Functions and variables in Oxygen namespace

oxy:execute-action-by-name()
oxy:execute-action-by-class()
$oxy:caret-position
$oxy:selection
$oxy:selection-parent

Correspondences between Oxygen Author Operations and XQuery Update operations

N. B.: What is in green is implemented and tested.

Oxygen Author Default Operation	XQuery Update Example	XQuery Update Expression
ChangeAttributeOperation	replace node @type with attribute simple {"true"}	**replace**

	replace value of node @type with “new value”	
ChangePseudoClassesOperation		
DeleteElementOperation	delete node .	**delete**
DeleteElementsOperation		
ExecuteMultipleActionsOperation	(insert nodes (<node1>content1</node1>, <node2>content2</node2>) after element()[last()] , replace node . with ./* . delete node ./element()[1])	**insert** **delete** **replace** **rename**
ExecuteTransformationScenariosOperation	oxy:execute-action-by-class(“ro.sync.ecss.extensions.commons.operations.ExecuteTransformationScenariosOperation”, $parameters)	**extension function**
InsertEquationOperation	insert node <math xmlns="http://www.w3.org/1998/Math/MathML"> <mi>π<!-- π --></mi> <mo>⁢<!-- ⁢ --></mo> <msup> <mi>r</mi> <mn>2</mn> </msup> </math> after .	**insert**
InsertFragmentOperation	insert nodes (<node1>content1</node1>, <node2>content2</node2>) after element()[last()]	**insert**
InsertFragmentOperation	insert nodes (<node1>content1</node1>, <node2>content2</node2>) after $oxy:caret-position	**insert**
InsertOrReplaceFragmentOperation	replace node . with <node1>{.}</node1> replace node . with (<node1>{.}</node1>, <node2>{.}</node2>)	**replace**

InsertOrReplaceTextOperation	if (string-length($oxy:selection) > 0) then (replace value of node $oxy:selection-parent with replace($oxy:selection-parent/text, $oxy:selection, “replacement”)) else (replace value of node $oxy:selection-parent with concat(substring(0, $oxy:caret-position), “replacement”, substring($oxy:caret-position + 1)))	**replace**
InsertXIncludeOperation	insert node <xi:include xmlns:xi="http://www.w3.org/2001/XInclude" href="gramGrp.xml" /> after .	**insert**
MoveCaretUtil	oxy:move-caret($position as xs:string)	
MoveElementOperation		
OpenInSystemAppOperation	oxy:execute-action-by-class(“ro.sync.ecss.extensions.commons.operations.OpenInSystemAppOperation”, $parameters)	**extension function**
RemovePseudoClassOperation		
RenameElementOperation	rename node . as "placeName"	**rename**
SetPseudoClassOperation		
ShowElementDocumentationOperation		
SurroundWithFragmentOperation	replace node element()[last()] with (<node1>content1</node1>, ., <node2>content2</node2>)	**replace**
SurroundWithTextOperation	replace value of node $oxy:selection-parent with concat(“prepended-content”, replace($oxy:selection-parent/text, $oxy:selection, “replacement”), “appended-content”)	**replace**
TogglePseudoClassOperation		
ToggleSurroundWithElementOperation	if (local-name(.) = ‘a’) then (replace . with {.}) else (), if (local-name(.) = ‘b’) then (replace . with ./a) else ()	**replace**
TransformOperation		
UnwrapTagsOperation	replace node . with ./*	**replace**

XQueryOperation	insert node doc('doc.xml') after .	**insert**
XSLTOperation	oxy:transform($parameters)	**extension function**
Open a page in Author mode	oxy:execute-action-by-name(“Author/No_tags”)	

Correspondences between XQuery Update and Oxygen Operations operations

5.1 Insert

Semantics:

1. SourceExpr ~~is evaluated as though it were an enclosed expression in an element constructor (see Rule 1e in Section 3.9.1.3 Content [XQ30]). The result of this step is either an error or a sequence of nodes to be inserted, called the insertion sequence. If the insertion sequence contains a document node, the document node is replaced in the insertion sequence by its children. If the insertion sequence contains an attribute node following a node that is not an attribute node, a type error is raised [err:XUTY0004]. Let $alist be the sequence of attribute nodes in the insertion sequence. Let $clist be the remainder of the insertion sequence, in its original order.~~

   Note:

   ~~Either $alist or $clist or both may be empty.~~
2. TargetExpr is evaluated and checked as follows:
   1. ~~If the result is an empty sequence, [err:XUDY0027] is raised.~~
   2. ~~If any form of into is specified, the result must be a single element or document node; any other non-empty result raises a type error [err:XUTY0005].~~
   3. ~~If before or after is specified, the result must be a single element, text, comment, or processing instruction node; any other non-empty result raises a type error [err:XUTY0006].~~
   4. ~~If before or after is specified, the node returned by the target expression must have a non-empty parent property [err:XUDY0029].~~
3. Let $target be the node returned by the target expression.
4. If $alist is not empty and any form of into is specified, the following checks are performed:

1. ~~$target must be an element node [err:XUTY0022].~~
2. No attribute node in $alist may have a QName whose implied namespace binding conflicts with a namespace binding in the "namespaces" property of $target [err:XUDY0023].
3. Multiple attribute nodes in $alist may not have QNames whose implied namespace bindings conflict with each other [err:XUDY0024].

5. If $alist is not empty and before or after is specified, the following checks are performed:
    1. ~~parent($target) must be an element node [err:XUDY0030].~~
    2. No attribute node in $alist may have a QName whose implied namespace binding conflicts with a namespace binding in the "namespaces" property of parent($target) [err:XUDY0023].
    3. Multiple attribute nodes in $alist may not have QNames whose implied namespace bindings conflict with each other [err:XUDY0024].
6. The result of the insert expression is an empty XDM instance and a pending update list constructed by merging the pending update lists returned by the SourceExpr and TargetExpr with the following update primitives using upd:mergeUpdates:
    1. If as first into is specified, the pending update list includes the following update primitives:
        1. ~~If $alist is not empty, upd:insertAttributes($target, $alist)~~
        2. ~~If $clist is not empty, upd:insertIntoAsFirst($target, $clist)~~ (remained for "TEXT", "COMMENT", "PROCESSING_INSTRUCTION")
    2. If as last into is specified, the pending update list includes the following update primitives:
        1. ~~If $alist is not empty, upd:insertAttributes($target, $alist)~~
        2. ~~If $clist is not empty, upd:insertIntoAsLast($target, $clist)~~ (remained for "TEXT", "COMMENT", "PROCESSING_INSTRUCTION")
    3. If into is specified with neither as first nor as last, the pending update list includes the following update primitives:
        1. ~~If $alist is not empty, upd:insertAttributes($target, $alist)~~
        2. ~~If $clist is not empty, upd:insertInto($target, $clist)~~ (remained for "TEXT", "COMMENT",

"PROCESSING_INSTRUCTION")

4. If before is specified, let $parent be the parent node of $target. The pending update list includes the following update primitives:
    1. ~~If $alist is not empty, upd:insertAttributes($parent, $alist)~~
    2. ~~If $clist is not empty, upd:insertBefore($target, $clist)~~ (remained for "TEXT", "COMMENT", "PROCESSING_INSTRUCTION")
5. If after is specified, let $parent be the parent node of $target. The pending update list includes the following update primitives:
    1. ~~If $alist is not empty, upd:insertAttributes($parent, $alist)~~
    2. ~~If $clist is not empty, upd:insertAfter($target, $clist)~~ (remained for "TEXT", "COMMENT", "PROCESSING_INSTRUCTION")

5.3.1 Replacing a Node

Semantics:

1. ~~The expression following the keyword with is evaluated as though it were an enclosed expression in an element constructor (see Rule 1e in Section 3.9.1.3 Content [XQ30]). Let $rlist be the node sequence that results from this evaluation. If $rlist contains a document node, the document node is replaced in $rlist by its children~~.
2. TargetExpr is evaluated and checked as follows:
    1. ~~If the result is an empty sequence, [err:XUDY0027] is raised.~~
    2. ~~If the result is non-empty and does not consist of a single element, attribute, text, comment, or processing instruction node, [err:XUTY0008] is raised.~~
    3. ~~If the result consists of a node whose parent property is empty, [err:XUDY0009] is raised.~~
3. ~~Let $target be the node returned by the target expression, and let $parent be its parent node.~~
4. ~~If $target is an element, text, comment, or processing instruction node, then $rlist must consist~~

exclusively of zero or more element, text, comment, or processing instruction nodes [err:XUTY0010].

5. If $target is an attribute node, then:
    1. $rlist must consist exclusively of zero or more attribute nodes [err:XUTY0011].
    2. No attribute node in $rlist may have a QName whose implied namespace binding conflicts with a namespace binding in the "namespaces" property of $parent [err:XUDY0023].
    3. Multiple attribute nodes in $rlist may not have QNames whose implied namespace bindings conflict with each other [err:XUDY0024].
6. The result of the replace expression is an empty XDM instance and a pending update list constructed by merging the pending update lists returned by the TargetExpr and the expression following the keyword with with the following update primitives using upd:mergeUpdates: upd:replaceNode($target, $rlist).

5.3.2 Replacing the Value of a Node

Semantics:

1. The expression following the keyword with is evaluated as though it were the content expression of a text node constructor (see Section 3.7.3.4 of [XQuery 3.0: An XML Query Language].) The result of this step, in the absence of errors, is either a single text node or an empty sequence. Let $text be the result of this step.
2. TargetExpr is evaluated and checked as follows:
    1. If the result is an empty sequence, [err:XUDY0027] is raised.
    2. If the result is non-empty and does not consist of a single element, attribute, text, comment, or processing instruction node, [err:XUTY0008] is raised.
3. Let $target be the node returned by the target expression.
4. If $target is an element node, the result of the replace expression is an empty XDM instance and a pending update list constructed by merging the pending update lists returned by the TargetExpr and the expression following the keyword with with the following update primitives using upd:mergeUpdates:

~~upd:replaceElementContent($target, $text)~~

5. If $target is an attribute, text, comment, or processing instruction node, let $string be the string value of the text node constructed in Step 1. If Step 1 did not construct a text node, let $string be a zero-length string. Then:
    1. If $target is a comment node, and $string contains two adjacent hyphens or ends with a hyphen, a dynamic error is raised [err:XQDY0072].
    2. If $target is a processing instruction node, and $string contains the substring "?>", a dynamic error is raised [err:XQDY0026].
    3. In the absence of errors, the result of a replace expression is an empty XDM instance and a pending update list constructed by merging the pending update lists returned by the TargetExpr and the expression following the keyword with with the following update primitives using upd:mergeUpdates: upd:replaceValue($target, $string).

Example XQuery Script

```
xquery version "3.0";

import module "http://expath.org/ns/user-agent";

declare variable $ua:document external;
declare variable $oxy:caret-position external;
declare variable $oxy:selection external;
declare variable $oxy:selection-parent external;

ua:action(
        "XQueryOperation",
```

```
        map {
        "name" := "XQueryOperation"
        },
        insert node doc('content-models/def.xml') after def[last()]
),
ua:action(
        "InsertFragmentOperation",
        map {
        "name" := "InsertFragmentOperation"
        },
        insert nodes (<node1>content1</node1>, <node2>content2</node2>) after element()[last()]
)
```

References

http://www.xqib.org/js/BGColors_source.html
event object
http://www.xqib.org/js/MouseButton_source.html

```
<create path="//*[@orekatua]//w[@cert='sure']/@lemma" type="xs:string"/>
```

is Invalid, you probably want to break this down into smaller range

indexes for each predicate...

<create path="//@orekatua" type="xs:string"/>

<create path="//w/@cert" type="xs:string"/>

<create path="//w/@lemma" type="xs:string"/>