

## XSLT 2.0 Quick reference. 2013-09-03+01:00

<http://www.dpawson.co.uk/xsl/rev2/rev2.html>  
Produced with DiType from RenderX

There are a number of standard attributes that may appear on any XSLT element: specifically version, exclude-result-prefixes, extension-element-prefixes, xpath-default-namespace, default-collation, and use-when.

**Element `xsl:analyze-string`**

Attributes:

- select as *expression*
- regex{ as *string* }
- flags as *string* }

```
<--Content:( xsl:matching-substring?, xsl:non-matching-substring?,
xsl:fallback* )-->
```

**Element `xsl:apply-imports`**

```
<--Content:( xsl:with-param* )-->
```

**Element `xsl:apply-templates`**

Attributes:

- select as *expression*
- mode as *token*

```
<--Content:( xsl:sort | xsl:with-param )* -->
```

**Element `xsl:attribute`**

(*sequence-constructor*)

Attributes:

- name{ as *qname* }
- namespace{ as *uri-reference* }
- select as *expression*
- separator{ as *string* }
- type as *qname*
- validation "strict|lax|preserve|strip"

```
<--Content:( sequence constructor )-->
```

**Element `xsl:attribute-set`**

Attributes:

- name as *qname*
- use-attribute-sets as *qnames*

```
<--Content:( xsl:attribute* )-->
```

**Element `xsl:call-template`**

Attributes:

- name as *qname*

```
<--Content:( xsl:with-param* )-->
```

**Element `xsl:character-map`**

Attributes:

- name as *qname*
- use-character-maps as *qnames*

```
<--Content:( xsl:output-character* )-->
```

**Element `xsl:choose`**

```
<--Content:( xsl:when+, xsl:otherwise? )-->
```

**Element `xsl:comment`**

(*sequence-constructor*)

Attributes:

- select as *expression*

```
<--Content:( sequence constructor )-->
```

**Element `xsl:copy`**

(*sequence-constructor*)

Attributes:

- copy-namespaces "yes|no"
- inherit-namespaces "yes|no"
- use-attribute-sets as *qnames*
- type as *qname*
- validation "strict|lax|preserve|strip"

```
<--Content:( sequence constructor )-->
```

**Element `xsl:copy-of`**

Attributes:

- select as *expression*
- copy-namespaces "yes|no"
- type as *qname*
- validation "strict|lax|preserve|strip"

**Element `xsl:decimal-format`**

Attributes:

- name as *qname*
- decimal-separator as *char*
- grouping-separator as *char*
- infinity as *string*
- minus-sign as *char*
- NaN as *string*
- percent as *char*
- per-mille as *char*
- zero-digit as *char*
- digit as *char*
- pattern-separator as *char*

**Element `xsl:document`**

(*sequence-constructor*)

Attributes:

- validation "strict|lax|preserve|strip"
- type as *qname*

```
<--Content:( sequence constructor )-->
```

**Element `xsl:element`**

(*sequence-constructor*)

Attributes:

- name{ as *qname* }
- namespace{ as *uri-reference* }
- inherit-namespaces "yes|no"
- use-attribute-sets as *qnames*
- type as *qname*
- validation "strict|lax|preserve|strip"

```
<--Content:( sequence constructor )-->
```

**Element `xsl:fallback`**

(*sequence-constructor*) <--Content:( sequence constructor )-->

**Element `xsl:for-each`**

Attributes:

- select as *expression*

```
<--Content:( xsl:sort*, sequence constructor )-->
```

**Element `xsl:for-each-group`**

Attributes:

- select as *expression*
- group-by as *expression*
- group-adjacent as *expression*
- group-starting-with as *pattern*
- group-ending-with as *pattern*
- collation{ as *uri* }

```
<--Content:( xsl:sort*, sequence constructor )-->
```

**Element `xsl:function`**

Attributes:

- name as *qname*
- as as *sequence-type*
- override "yes|no"

```
<--Content:( xsl:param*, sequence constructor )-->
```

**Element `xsl:if`**

(*sequence-constructor*)

Attributes:

- test as *expression*

```
<--Content:( sequence constructor )-->
```

**Element `xsl:import`**

Attributes:

- href as *uri-reference*

**Element `xsl:import-schema`**

Attributes:

- namespace as *uri-reference*
- schema-location as *uri-reference*

```
<--Content:( xsl:xs:schema? )-->
```

**Element `xsl:include`**

Attributes:

- href as *uri-reference*

**Element `xsl:key`**

(*sequence-constructor*)

Attributes:

- name as *qname*
- match as *pattern*
- use as *expression*
- collation as *uri*

<--Content:( sequence constructor )-->

#### Element **xsl:matching-substring**

(sequence-constructor) <--Content:( sequence constructor )-->

#### Element **xsl:message**

(sequence-constructor)

Attributes:

- select as *expression*
- terminate{ "yes|no" }

<--Content:( sequence constructor )-->

#### Element **xsl:namespace**

(sequence-constructor)

Attributes:

- name{ as *ncname* }
- select as *expression*

<--Content:( sequence constructor )-->

#### Element **xsl:namespace-alias**

Attributes:

- stylesheet-prefix as *prefix* "#default"
- result-prefix as *prefix* "#default"

#### Element **xsl:next-match**

<--Content:( xsl:with-param | xsl:fallback )\* -->

#### Element **xsl:non-matching-substring**

(sequence-constructor) <--Content:( sequence constructor )-->

#### Element **xsl:number**

Attributes:

- value as *expression*
- select as *expression*
- level "single|multiple|any"
- count as *pattern*
- from as *pattern*
- format{ as *string* }
- lang{ as *nmtoken* }
- letter-value{ "alphabetic|traditional" }
- ordinal{ as *string* }
- grouping-separator{ as *char* }
- grouping-size{ as *number* }

#### Element **xsl:otherwise**

(sequence-constructor) <--Content:( sequence constructor )-->

#### Element **xsl:output**

Attributes:

- name as *qname*
- method "xml|html|xhtml|text" as *qname-but-not-ncname*
- byte-order-mark "yes|no"
- cdata-section-elements as *qnames*
- doctype-public as *string*
- doctype-system as *string*
- encoding as *string*
- escape-uri-attributes "yes|no"
- include-content-type "yes|no"
- indent "yes|no"
- media-type as *string*
- normalization-form "NFC|NFD|NFKC|NFKD|fully-normalized|none" as *nmtoken*
- omit-xml-declaration "yes|no"
- standalone "yes|no|omit"
- undeclare-prefixes "yes|no"
- use-character-maps as *qnames*
- version as *nmtoken*

#### Element **xsl:output-character**

Attributes:

- character as *char*
- string as *string*

#### Element **xsl:param**

(sequence-constructor)

Attributes:

- name as *qname*
- select as *expression*
- as as *sequence-type*
- required "yes|no"
- tunnel "yes|no"

<--Content:( sequence constructor )-->

#### Element **xsl:perform-sort**

Attributes:

- select as *expression*

<--Content:( xsl:sort+, sequence constructor )-->

#### Element **xsl:preserve-space**

Attributes:

- elements as *tokens*

#### Element **xsl:processing-instruction**

(sequence-constructor)

Attributes:

- name{ as *ncname* }
- select as *expression*

<--Content:( sequence constructor )-->

#### Element **xsl:result-document**

(sequence-constructor)

Attributes:

- format{ as *qname* }
- href{ as *uri-reference* }
- validation "strict|lax|preserve|strip"
- type as *qname*
- method{ "xml|html|xhtml|text" as *qname-but-not-ncname* }
- byte-order-mark{ "yes|no" }
- cdata-section-elements{ as *qnames* }
- doctype-public{ as *string* }
- doctype-system{ as *string* }
- encoding{ as *string* }
- escape-uri-attributes{ "yes|no" }
- include-content-type{ "yes|no" }
- indent{ "yes|no" }
- media-type{ as *string* }
- normalization-form{ "NFC|NFD|NFKC|NFKD|fully-normalized|none" as *nmtoken* }
- omit-xml-declaration{ "yes|no" }
- standalone{ "yes|no|omit" }
- undeclare-prefixes{ "yes|no" }
- use-character-maps as *qnames*
- output-version{ as *nmtoken* }

<--Content:( sequence constructor )-->

#### Element **xsl:sequence**

Attributes:

- select as *expression*

<--Content:( xsl:fallback\* )-->

#### Element **xsl:sort**

(sequence-constructor)

Attributes:

- select as *expression*
- lang{ as *nmtoken* }
- order{ "ascending|descending" }
- collation{ as *uri* }
- stable{ "yes|no" }
- case-order{ "upper-first|lower-first" }
- data-type{ "text|number" as *qname-but-not-ncname* }

<--Content:( sequence constructor )-->

#### Element **xsl:strip-space**

Attributes:

- elements as *tokens*

#### Element **xsl:stylesheet**

Attributes:

- id as *id*
- extension-element-prefixes as *tokens*
- exclude-result-prefixes as *tokens*
- version as *number*
- xpath-default-namespace as *uri*
- default-validation "preserve|strip"
- default-collation as *uri-list*
- input-type-annotations "preserve|strip|unspecified"

<--Content:( xsl:import\*, other declarations )-->

#### Element **xsl:template**

Attributes:

- match as *pattern*
- name as *qname*
- priority as *number*
- mode as *tokens*
- as as *sequence-type*

<--Content:( xsl:param\*, sequence constructor )-->

#### Element **xsl:text**

Attributes:

- disable-output-escaping "yes|no" *Deprecated*

<--Content:( <text/> )-->

#### Element **xsl:transform**

Attributes:

- id as *id*
- extension-element-prefixes as *tokens*
- exclude-result-prefixes as *tokens*
- version as *number*
- xpath-default-namespace as *uri*
- default-validation "preserve|strip"
- default-collation as *uri-list*
- input-type-annotations "preserve|strip|unspecified"

<--Content:( xsl:import\*, other declarations )-->

#### Element **xsl:value-of**

(sequence-constructor)

Attributes:

- select as *expression*
- separator{ as *string* }
- disable-output-escaping "yes|no" *Deprecated*

<--Content:( sequence constructor )-->

#### Element **xsl:variable**

(sequence-constructor)

Attributes:

- name as *qname*
- select as *expression*
- as as *sequence-type*

<--Content:( sequence constructor )-->

#### Element **xsl:when**

(sequence-constructor)

Attributes:

- test as *expression*

<--Content:( sequence constructor )-->

#### Element **xsl:with-param**

(sequence-constructor)

Attributes:

- name as *qname*
- select as *expression*
- as as *sequence-type*
- tunnel "yes|no"

<--Content:( sequence constructor )-->

## XSLT functions

xslt: **current** () as *item()*

xslt: **current-group** () as *item()*

xslt: **current-grouping-key** () as *xs:anyAtomicType*

xslt: **document** (\$uri-sequence as *item()* [ \$base-node ] as *node()*) as *node()*

xslt: **element-available** (\$element-name as *xs:string*) as *xs:boolean*

xslt: **format-date** (\$value as *xs:date*, \$picture as *xs:string*, \$language as *xs:string*, \$calendar as *xs:string*, \$country as *xs:string*) as *xs:string*

xslt: **format-dateTime** (\$value as *xs:dateTime*, \$picture as *xs:string*, \$language as *xs:string*, \$calendar as *xs:string*, \$country as *xs:string*) as *xs:string*

xslt: **format-number** (\$value as *numeric*, \$picture as *xs:string* [ \$decimal-format-name ] as *xs:string*) as *xs:string*

xslt: **format-time** (\$value as *xs:time*, \$picture as *xs:string*, \$language as *xs:string*, \$calendar as *xs:string*, \$country as *xs:string*) as *xs:string*

xslt: **function-available** (\$function-name as *xs:string* [ \$arity ] as *xs:integer*) as *xs:boolean*

xslt: **generate-id** ([ \$node ] as *node()*) as *xs:string*

xslt: **key** (\$key-name as *xs:string*, \$key-value as *xs:anyAtomicType* [ \$stop ] as *node()*) as *node()*

xslt: **regex-group** (\$group-number as *xs:integer*) as *xs:string*

xslt: **system-property** (\$property-name as *xs:string*) as *xs:string*

xslt: **type-available** (\$type-name as *xs:string*) as *xs:boolean*

xslt: **unparsed-entity-public-id** (\$entity-name as *xs:string*) as *xs:string*

xslt: **unparsed-entity-uri** (\$entity-name as *xs:string*) as *xs:anyURI*

xslt: **unparsed-text** (\$href as *xs:string* [ \$encoding ] as *xs:string*) as *xs:string*

xslt: **unparsed-text-available** (\$href as *xs:string* [ \$encoding ] as *xs:string*) as *xs:boolean*

## XPATH functions

xpath: **ENTITY** (\$arg as *xs:anyAtomicType*) as *xs:ENTITY*

xpath: **ID** (\$arg as *xs:anyAtomicType*) as *xs:ID*

xpath: **IDREF** (\$arg as *xs:anyAtomicType*) as *xs:IDREF*

xpath: **NCName** (\$arg as *xs:anyAtomicType*) as *xs:NCName*

xpath: **NMTOKEN** (\$arg as *xs:anyAtomicType*) as *xs:NMTOKEN*

xpath: **Name** (\$arg as *xs:anyAtomicType*) as *xs>Name*

xpath: **QName** (\$arg as *xs:anyAtomicType* [ \$paramURI ] as *xs:string*, [ \$paramQName ] as *xs:string*) as *xs:QName*

xpath: **abs** (\$arg as *numeric*) as *numeric*

xpath: **adjust-date-to-timezone** (\$arg as *xs:date* [ \$timezone ] as *xs:dayTimeDuration*) as *xs:date*

xpath: **adjust-dateTime-to-timezone** (\$arg as *xs:dateTime* [ \$timezone ] as *xs:dayTimeDuration*) as *xs:dateTime*

xpath: **adjust-time-to-timezone** (\$arg as *xs:time* [ \$timezone ] as *xs:dayTimeDuration*) as *xs:time*

xpath: **anyURI** (\$arg as *xs:anyAtomicType*) as *xs:anyURI*

xpath: **avg** (\$arg as *xs:anyAtomicType\**) as *xs:anyAtomicType*

xpath: **base-uri** ([ \$arg ] as *node()*) as *xs:anyURI*

xpath: **base64Binary** (\$arg as *xs:anyAtomicType*) as *xs:base64Binary*

xpath: **boolean** (\$arg as *xs:anyAtomicType*) as *xs:boolean*

xpath: **byte** (\$arg as *xs:anyAtomicType*) as *xs:byte*

xpath: **ceiling** (\$arg as *numeric*) as *numeric*

xpath: **codepoint-equal** (\$comparand1 as *xs:string*, \$comparand2 as *xs:string*) as *xs:boolean*

xpath: **codepoints-to-string** (\$arg as *xs:integer\**) as *xs:string*

xpath: **collection** ([ \$arg ] as *xs:string*) as *node()\**

xpath: **compare** (\$comparand1 as *xs:string*, \$comparand2 as *xs:string* [ \$collation ] as *xs:string*) as *xs:integer*

xpath: **concat** (\$arg1 as *xs:anyAtomicType*, \$arg2 as *xs:anyAtomicType*, \$... as ) as *xs:string*

xpath: **contains** (\$arg1 as *xs:string*, \$arg2 as *xs:string* [ \$collation ] as *xs:string*) as *xs:boolean*

xpath: **count** (\$arg as *item()\**) as *xs:integer*

xpath: **current-date** () as *xs:date*

xpath: **current-dateTime** () as *xs:dateTime*

xpath: **current-time** () as *xs:time*

xpath: **data** (\$arg as *item()\**) as *xs:anyAtomicType\**

xpath: **date** (\$arg as *xs:anyAtomicType*) as *xs:date*

xpath: **dateTime** (\$arg as *xs:anyAtomicType* [ \$arg1 ] as *xs:date*, [ \$arg2 ] as *xs:time*) as *xs:dateTime*

xpath: **day-from-date** (\$arg as *xs:date*) as *xs:integer*

xpath: **day-from-dateTime** (\$arg as *xs:dateTime*) as *xs:integer*

xpath: **dayTimeDuration** (\$arg as *xs:anyAtomicType*) as *xs:dayTimeDuration*

xpath: **days-from-duration** (\$arg as *xs:duration*) as *xs:integer*

xpath: **decimal** (\$arg as *xs:anyAtomicType*) as *xs:decimal*

xpath: **deep-equal** (\$parameter1 as *item()\**, \$parameter2 as *item()\** [ \$collation ] as *string*) as *xs:boolean*

xpath: **default-collation** () as *xs:string*

xpath: **distinct-values** (\$arg as *xs:anyAtomicType\** [ \$collation ] as *xs:string*) as *xs:anyAtomicType\**

xpath: **doc** (\$uri as *xs:string*) as *document-node()*

xpath: **doc-available** (\$uri as *xs:string*) as *xs:boolean*

xpath: **document-uri** (\$arg as *node()*) as *xs:anyURI*

xpath: **double** (\$arg as *xs:anyAtomicType*) as *xs:double*

xpath: **duration** (\$arg as *xs:anyAtomicType*) as *xs:duration*

xpath: **empty** (\$arg as *item()\**) as *xs:boolean*

xpath: **encode-for-uri** (\$uri-part as *xs:string*) as *xs:string*

xpath: **ends-with** (\$arg1 as *xs:string*, \$arg2 as *xs:string* [ \$collation ] as *xs:string*) as *xs:boolean*

xpath: **error** ([ \$error ] as *xs:QName* [ \$error ] as *xs:QName*, [ \$description ] as *xs:string* [ \$error ] as *xs:QName*, [ \$description ] as *xs:string*, [ \$error-object ] as *item()\**) as *none*

xpath: **escape-html-uri** (\$uri as *xs:string*) as *xs:string*

xpath: **exactly-one** (\$arg as *item()\**) as *item()*

xpath: **exists** (\$arg as item(\*) as *xs:boolean*  
 xpath: **false** () as *xs:boolean*  
 xpath: **float** (\$arg as xs:anyAtomicType) as *xs:float*  
 xpath: **floor** (\$arg as numeric) as *numeric*  
 xpath: **gDay** (\$arg as xs:anyAtomicType) as *xs:gDay*  
 xpath: **gMonth** (\$arg as xs:anyAtomicType) as *xs:gMonth*  
 xpath: **gMonthDay** (\$arg as xs:anyAtomicType) as *xs:gMonthDay*  
 xpath: **gYear** (\$arg as xs:anyAtomicType) as *xs:gYear*  
 xpath: **gYearMonth** (\$arg as xs:anyAtomicType) as *xs:gYearMonth*  
 xpath: **hexBinary** (\$arg as xs:anyAtomicType) as *xs:hexBinary*  
 xpath: **hours-from-dateTime** (\$arg as xs:dateTime) as *xs:integer*  
 xpath: **hours-from-duration** (\$arg as xs:duration) as *xs:integer*  
 xpath: **hours-from-time** (\$arg as xs:time) as *xs:integer*  
 xpath: **id** (\$arg as xs:string\* [ \$node] as node()) as *element()*\*  
 xpath: **idref** (\$arg as xs:string\* [ \$node] as node()) as *node()*\*  
 xpath: **implicit-timezone** () as *xs:dayTimeDuration*  
 xpath: **in-scope-prefixes** (\$element as element()) as *xs:string\**  
 xpath: **index-of** (\$seqParam as xs:anyAtomicType\*, \$srchParam as xs:anyAtomicType [ \$collation] as xs:string) as *xs:integer\**  
 xpath: **insert-before** (\$target as item()\*, \$position as xs:integer, \$inserts as item()\*) as *item()*\*  
 xpath: **int** (\$arg as xs:anyAtomicType) as *xs:int*  
 xpath: **integer** (\$arg as xs:anyAtomicType) as *xs:integer*  
 xpath: **iri-to-uri** (\$iri as xs:string) as *xs:string*  
 xpath: **lang** (\$stetlang as xs:string [ \$node] as node()) as *xs:boolean*  
 xpath: **language** (\$arg as xs:anyAtomicType) as *xs:language*  
 xpath: **last** () as *xs:integer*  
 xpath: **local-name** ([ \$arg] as node()) as *xs:string*  
 xpath: **local-name-from-QName** (\$arg as xs:QName) as *xs:NCName*  
 xpath: **long** (\$arg as xs:anyAtomicType) as *xs:long*  
 xpath: **lower-case** (\$arg as xs:string) as *xs:string*  
 xpath: **matches** (\$input as xs:string, \$pattern as xs:string [ \$flags] as xs:string) as *xs:boolean*  
 xpath: **max** (\$arg as xs:anyAtomicType\* [ \$collation] as string) as *xs:anyAtomicType*  
 xpath: **min** (\$arg as xs:anyAtomicType\* [ \$collation] as string) as *xs:anyAtomicType*  
 xpath: **minutes-from-dateTime** (\$arg as xs:dateTime) as *xs:integer*  
 xpath: **minutes-from-duration** (\$arg as xs:duration) as *xs:integer*  
 xpath: **minutes-from-time** (\$arg as xs:time) as *xs:integer*  
 xpath: **month-from-date** (\$arg as xs:date) as *xs:integer*  
 xpath: **month-from-dateTime** (\$arg as xs:dateTime) as *xs:integer*  
 xpath: **months-from-duration** (\$arg as xs:duration) as *xs:integer*  
 xpath: **my:hatSize** (\$arg as xs:anyAtomicType) as *my:hatSize*  
 xpath: **name** ([ \$arg] as node()) as *xs:string*  
 xpath: **namespace-uri** ([ \$arg] as node()) as *xs:anyURI*  
 xpath: **namespace-uri-for-prefix** (\$prefix as xs:string, \$element as element()) as *xs:anyURI*  
 xpath: **namespace-uri-from-QName** (\$arg as xs:QName) as *xs:anyURI*  
 xpath: **negativeInteger** (\$arg as xs:anyAtomicType) as *xs:negativeInteger*  
 xpath: **nilled** (\$arg as node()) as *xs:boolean*  
 xpath: **node-name** (\$arg as node()) as *xs:QName*  
 xpath: **nonNegativeInteger** (\$arg as xs:anyAtomicType) as *xs:nonNegativeInteger*  
 xpath: **nonPositiveInteger** (\$arg as xs:anyAtomicType) as *xs:nonPositiveInteger*  
 xpath: **normalize-space** ([ \$arg] as xs:string) as *xs:string*  
 xpath: **normalize-unicode** (\$arg as xs:string [ \$normalizationForm] as xs:string) as *xs:string*  
 xpath: **normalizedString** (\$arg as xs:anyAtomicType) as *xs:normalizedString*  
 xpath: **not** (\$arg as item(\*) as *xs:boolean*  
 xpath: **number** ([ \$arg] as xs:anyAtomicType) as *xs:double*  
 xpath: **one-or-more** (\$arg as item()\*) as *item()*+  
 xpath: **position** () as *xs:integer*  
 xpath: **positiveInteger** (\$arg as xs:anyAtomicType) as *xs:positiveInteger*  
 xpath: **prefix-from-QName** (\$arg as xs:QName) as *xs:NCName*  
 xpath: **remove** (\$target as item()\*, \$position as xs:integer) as *item()*\*  
 xpath: **replace** (\$input as xs:string, \$pattern as xs:string, \$replacement as xs:string [ \$flags] as xs:string) as *xs:string*  
 xpath: **resolve-QName** (\$qname as xs:string, \$element as element()) as *xs:QName*  
 xpath: **resolve-uri** (\$relative as xs:string [ \$base] as xs:string) as *xs:anyURI*  
 xpath: **reverse** (\$arg as item()\*) as *item()*\*  
 xpath: **root** ([ \$arg] as node()) as *node()*

xpath: **round** (\$arg as numeric) as *numeric*  
 xpath: **round-half-to-even** (\$arg as numeric [ \$precision] as xs:integer) as *numeric*  
 xpath: **seconds-from-dateTime** (\$arg as xs:dateTime) as *xs:decimal*  
 xpath: **seconds-from-duration** (\$arg as xs:duration) as *xs:decimal*  
 xpath: **seconds-from-time** (\$arg as xs:time) as *xs:decimal*  
 xpath: **short** (\$arg as xs:anyAtomicType) as *xs:short*  
 xpath: **starts-with** (\$arg1 as xs:string, \$arg2 as xs:string [ \$collation] as xs:string) as *xs:boolean*  
 xpath: **static-base-uri** () as *xs:anyURI*  
 xpath: **string** ([ \$arg] as item() [ \$arg] as xs:anyAtomicType) as *xs:string*  
 xpath: **string-join** (\$arg1 as xs:string\*, \$arg2 as xs:string) as *xs:string*  
 xpath: **string-length** ([ \$arg] as xs:string) as *xs:integer*  
 xpath: **string-to-codepoints** (\$arg as xs:string) as *xs:integer\**  
 xpath: **subsequence** (\$sourceSeq as item()\*, \$startingLoc as xs:double [ \$length] as xs:double) as *item()*\*  
 xpath: **substring** (\$sourceString as xs:string, \$startingLoc as xs:double [ \$length] as xs:double) as *xs:string*  
 xpath: **substring-after** (\$arg1 as xs:string, \$arg2 as xs:string [ \$collation] as xs:string) as *xs:string*  
 xpath: **substring-before** (\$arg1 as xs:string, \$arg2 as xs:string [ \$collation] as xs:string) as *xs:string*  
 xpath: **sum** (\$arg as xs:anyAtomicType\* [ \$zero] as xs:anyAtomicType) as *xs:anyAtomicType*  
 xpath: **time** (\$arg as xs:anyAtomicType) as *xs:time*  
 xpath: **timezone-from-date** (\$arg as xs:date) as *xs:dayTimeDuration*  
 xpath: **timezone-from-dateTime** (\$arg as xs:dateTime) as *xs:dayTimeDuration*  
 xpath: **timezone-from-time** (\$arg as xs:time) as *xs:dayTimeDuration*  
 xpath: **token** (\$arg as xs:anyAtomicType) as *xs:token*  
 xpath: **tokenize** (\$input as xs:string, \$pattern as xs:string [ \$flags] as xs:string) as *xs:string\**  
 xpath: **trace** (\$value as item()\*, \$label as xs:string) as *item()*\*  
 xpath: **translate** (\$arg as xs:string, \$mapString as xs:string, \$transString as xs:string) as *xs:string*  
 xpath: **true** () as *xs:boolean*  
 xpath: **unordered** (\$sourceSeq as item()\*) as *item()*\*  
 xpath: **unsignedByte** (\$arg as xs:anyAtomicType) as *xs:unsignedByte*  
 xpath: **unsignedInt** (\$arg as xs:anyAtomicType) as *xs:unsignedInt*  
 xpath: **unsignedLong** (\$arg as xs:anyAtomicType) as *xs:unsignedLong*  
 xpath: **unsignedShort** (\$arg as xs:anyAtomicType) as *xs:unsignedShort*  
 xpath: **untypedAtomic** (\$arg as xs:anyAtomicType) as *xs:untypedAtomic*  
 xpath: **upper-case** (\$arg as xs:string) as *xs:string*  
 xpath: **year-from-date** (\$arg as xs:date) as *xs:integer*  
 xpath: **year-from-dateTime** (\$arg as xs:dateTime) as *xs:integer*  
 xpath: **yearMonthDuration** (\$arg as xs:anyAtomicType) as *xs:yearMonthDuration*  
 xpath: **years-from-duration** (\$arg as xs:duration) as *xs:integer*  
 xpath: **zero-or-one** (\$arg as item()\*) as *item()*

## Precedence Order

1	, (comma)	left-to-right
3	for, some, every, if	left-to-right
4	or	left-to-right
5	and	left-to-right
6	eq, ne, lt, le, gt, ge, =, !=, <, <=, >, >=, is, <<, >>	left-to-right
7	to	left-to-right
8	+, -	left-to-right
9	*, div, idiv, mod	left-to-right
10	union,	left-to-right
11	intersect, except	left-to-right
12	instance of	left-to-right
13	treat	left-to-right
14	castable	left-to-right
15	cast	left-to-right
16	-(unary), +(unary)	right-to-left
17	?, *(OccurrenceIndicator), +(OccurrenceIndicator)	left-to-right
18	/, //	left-to-right
19	[ ], ( ), { }	left-to-right

## Key

{Attribute Value Template}

Source (xslt or xpath), function name, (\$parameter as type), as function  
return type. E.g. xpath: seconds-from-dateTime (\$arg as xs:dateTime) as  
xs:decimal

optional arguments to functions are shown as [\$parameter as type]