

Text/String Functions

codepoint-equal(xs:string?, xs:string?) as xs:boolean?
codepoints-to-string(xs:integer*) as xs:string
compare(xs:string?, xs:string?) as xs:integer?
compare(xs:string?, xs:string?, xs:string) as xs:integer?
concat(xs:anyAtomicType?, xs:anyAtomicType?,) as xs:string
contains(xs:string?, xs:string?) as xs:boolean
contains(xs:string?, xs:string?, xs:string) as xs:boolean
current-date() as xs:date
current-dateTime() as xs:dateTime
current-time() as xs:time
default-collation() as xs:string
encode-for-uri(xs:string?) as xs:string
ends-with(xs:string?, xs:string?) as xs:boolean
ends-with(xs:string?, xs:string?, xs:string) as xs:boolean
escape-html-uri(xs:string?) as xs:string
lower-case(xs:string?) as xs:string
normalize-space() as xs:string
normalize-space(xs:string?) as xs:string
normalize-unicode(xs:string?) as xs:string
normalize-unicode(xs:string?, xs:string) as xs:string
starts-with(xs:string?, xs:string?) as xs:boolean
starts-with(xs:string?, xs:string?, xs:string) as xs:boolean
string() as xs:string
string(item()) as xs:string
string-join(xs:string*, xs:string) as xs:string
string-length() as xs:integer
string-length(xs:string?) as xs:integer
string-to-codepoints(xs:string?) as xs:integer*
substring(xs:string?, xs:double) as xs:string
substring(xs:string?, xs:double, xs:double) as xs:string
substring-after(xs:string?, xs:string?) as xs:string
substring-after(xs:string?, xs:string?, xs:string) as xs:string
substring-before(xs:string?, xs:string?) as xs:string
substring-before(xs:string?, xs:string?, xs:string) as xs:string
translate(xs:string?, xs:string, xs:string) as xs:string
upper-case(xs:string?) as xs:string

XSL-List:

<http://www.mulberrytech.com/xsl/xsl-list>

REGEX Functions

matches(xs:string?, xs:string) as xs:boolean
matches(xs:string?, xs:string, xs:string) as xs:boolean
replace(xs:string?, xs:string, xs:string) as xs:string
replace(xs:string?, xs:string, xs:string, xs:string) as xs:string
tokenize(xs:string?, xs:string) as xs:string*
tokenize(xs:string?, xs:string, xs:string) as xs:string*

Arithmetic Operators

+ (numeric) as ~numeric
(numeric) + (numeric) as ~numeric
- (numeric) as ~numeric
(numeric) - (numeric) as ~numeric
(numeric) * (numeric) as ~numeric
(numeric) **div** (numeric) as ~numeric
(numeric) **idiv** (numeric) as xs:integer
(numeric) **mod** (numeric) as ~numeric

Arithmetic Functions

abs(numeric?) as ~numeric?
avg(xs:anyAtomicType*) as ~xs:anyAtomicType?
ceiling(numeric?) as ~numeric?
floor(numeric?) as ~numeric?
number() as xs:double
number(xs:anyAtomicType?) as xs:double
round(numeric?) as ~numeric?
round-half-to-even(numeric?) as ~numeric?
round-half-to-even(numeric?, xs:integer) as ~numeric?
sum(xs:anyAtomicType*) as ~xs:anyAtomicType?
sum(xs:anyAtomicType*, xs:anyAtomicType?) as ~xs:anyAtomicType?

The **eq**, **ne**, **lt**, **gt**, **le** and **ge** comparisons are supported for the numeric types.

Sequence Operators

(item()*), (item()* as ~item()*
(node()* **union** (node()* as ~node()*
(node()* **intersect** (node()* as ~node()*
(node()* **except** (node()* as ~node()*
(xs:integer) **to** (xs:integer) as xs:integer*

Node Comparisons

(node()) **is** (node()) as xs:boolean
(node()) << (node()) as xs:boolean
(node()) >> (node()) as xs:boolean

Sequence and Node Functions

collection() as node()*
collection(xs:string?) as node()*
count(item()* as xs:integer
data(item()* as ~xs:anyAtomicType*
deep-equal(item()* as xs:boolean
deep-equal(item()* as xs:boolean
distinct-values(xs:anyAtomicType*) as ~xs:anyAtomicType*
distinct-values(xs:anyAtomicType*, xs:string) as ~xs:anyAtomicType*
doc(xs:string?) as document-node()?
empty(item()* as xs:boolean
exactly-one(item()* as ~item()*
exists(item()* as xs:boolean
index-of(xs:anyAtomicType*, xs:anyAtomicType) as xs:integer*
index-of(xs:anyAtomicType*, xs:anyAtomicType, xs:string) as xs:integer*
insert-before(item()* as xs:integer, item()* as ~item()*
last() as xs:integer
nilled(node()) as xs:boolean?
node-name(node()) as xs:QName?
one-or-more(item()* as ~item()+
position() as xs:integer
remove(item()* as xs:integer) as ~item()*
reverse(item()* as ~item()*
root() as node()
root(node()) as node()?
subsequence(item()* as xs:double) as ~item()*
subsequence(item()* as xs:double, xs:double) as ~item()*
unordered(item()* as ~item()*
zero-or-one(item()* as ~item()*

Miscellaneous Functions

error() as none
error(xs:QName) as none
error(xs:QName?, xs:string) as none
error(xs:QName?, xs:string, item()* as none
lang(xs:string?) as xs:boolean
lang(xs:string?, node()) as xs:boolean
max(xs:anyAtomicType*) as ~xs:anyAtomicType*
max(xs:anyAtomicType*, string) as ~xs:anyAtomicType*
min(xs:anyAtomicType*) as ~xs:anyAtomicType?
min(xs:anyAtomicType*, string) as ~xs:anyAtomicType*
trace(item()* as xs:string) as ~item()*

Boolean Functions

boolean(item()* as xs:boolean
false() as xs:boolean
not(item()* as xs:boolean
true() as xs:boolean
The **eq**, **ne**, **lt**, **gt**, **le** and **ge** comparisons are supported for the **xs:boolean** type.
URI, ID and XML Name Functions
base-uri() as xs:anyURI?
base-uri(node()) as xs:anyURI?
document-uri(node()) as xs:anyURI?
doc-available(xs:string?) as xs:boolean
in-scope-prefixes(element()) as xs:string*
id(xs:string*) as element()*
id(xs:string*, node()) as element()*
idref(xs:string*) as node()*
idref(xs:string*, node()) as node()*
iri-to-uri(xs:string?) as xs:string
local-name() as xs:string
local-name(node()) as xs:string
local-name-from-QName(xs:QName?) as xs:NCName?
name() as xs:string
name(node()) as xs:string
namespace-uri() as xs:anyURI
namespace-uri(node()) as xs:anyURI
namespace-uri-for-prefix(xs:string?, element()) as xs:anyURI?
namespace-uri-from-QName(xs:QName?) as xs:anyURI?
prefix-from-QName(xs:QName?) as xs:NCName?
QName(xs:string?, xs:string) as xs:QName
resolve-QName(xs:string?, element()) as xs:QName?
resolve-uri(xs:string?) as xs:anyURI?
resolve-uri(xs:string?, xs:string) as xs:anyURI?
static-base-uri() as xs:anyURI?

Built-In Schema Types

These types are available in all implementations.

xs:anyAtomicType	xs:gMonth
xs:anySimpleType	xs:anyURI
xs:anyType	xs:gMonthDay
xs:base64Binary	xs:Year
xs:boolean	xs:gYearMonth
xs:date	xs:hexBinary
xs:dateTime	xs:integer
xs:dayTimeDuration	xs:QName
xs:decimal	xs:string
xs:double	xs:time
xs:duration	xs:untyped
xs:float	xs:untypedAtomic
xs:gDay	xs:yearMonthDuration

Date/Time Functions

adjust-date-to-timezone(xs:date?) as xs:date?
adjust-date-to-timezone(xs:date?, xs:dayTimeDuration?) as xs:date?
adjust-dateTime-to-timezone(xs:dateTime?) as xs:dateTime?
adjust-dateTime-to-timezone(xs:dateTime?, xs:dayTimeDuration?) as xs:dateTime?
adjust-time-to-timezone(xs:time?) as xs:time?
adjust-time-to-timezone(xs:time?, xs:dayTimeDuration?) as xs:time?
dateTime(xs:date?, xs:time?) as xs:dateTime?
day-from-date(xs:date?) as xs:integer?
day-from-dateTime(xs:dateTime?) as xs:integer?
days-from-duration(xs:duration?) as xs:integer?
hours-from-dateTime(xs:dateTime?) as xs:integer?
hours-from-duration(xs:duration?) as xs:integer?
hours-from-time(xs:time?) as xs:integer?
implicit-timezone() as xs:dayTimeDuration
minutes-from-dateTime(xs:dateTime?) as xs:integer?
minutes-from-duration(xs:duration?) as xs:integer?
minutes-from-time(xs:time?) as xs:integer?
month-from-date(xs:date?) as xs:integer?
month-from-dateTime(xs:dateTime?) as xs:integer?
months-from-duration(xs:duration?) as xs:integer?
seconds-from-dateTime(xs:dateTime?) as xs:decimal?
seconds-from-duration(xs:duration?) as xs:decimal?
seconds-from-time(xs:time?) as xs:decimal?
timezone-from-date(xs:date?) as xs:dayTimeDuration?
timezone-from-dateTime(xs:dateTime?) as xs:dayTimeDuration?
timezone-from-time(xs:time?) as xs:dayTimeDuration?
year-from-date(xs:date?) as xs:integer?
year-from-dateTime(xs:dateTime?) as xs:integer?
years-from-duration(xs:duration?) as xs:integer?

XPath 2.0:

<http://www.w3.org/TR/xpath20/>

XQuery 1.0:

<http://www.w3.org/TR/xquery/>

XQuery 1.0 & XPath 2.0 Functions & Operators:

<http://www.w3.org/TR/xpath-functions/>

XSLT-Only Functions

current() as item()
current-group() as item()*
current-grouping-key() as xs:anyAtomicType?
document(item()* as node()*
document(item()* as node() as node()*
element-available(xs:string) as xs:boolean
format-dateTime(xs:dateTime?, xs:string, xs:string?, xs:string?, xs:string?) as xs:string?
format-dateTime(xs:dateTime?, xs:string) as xs:string?
format-date(xs:date?, xs:string, xs:string?, xs:string?, xs:string?) as xs:string?
format-date(xs:date?, xs:string) as xs:string?
format-number(numeric?, xs:string) as xs:string
format-number(numeric?, xs:string, xs:string) as xs:string
format-time(xs:time?, xs:string, xs:string?, xs:string?, xs:string?) as xs:string?
format-time(xs:time?, xs:string) as xs:string?
function-available(xs:string) as xs:boolean
function-available(xs:string, xs:integer) as xs:boolean
generate-id() as xs:string
generate-id(node()) as xs:string
key(xs:string, xs:anyAtomicType*) as node()*
key(xs:string, xs:anyAtomicType*, node()) as node()*
regex-group(xs:integer) as xs:string
system-property(xs:string) as xs:string
type-available(xs:string) as xs:boolean
unparsed-text(xs:string?) as xs:string?
unparsed-text(xs:string?, xs:string) as xs:string?
unparsed-text-available(xs:string?) as xs:boolean
unparsed-text-available(xs:string?, xs:string?) as xs:boolean
unparsed-entity-uri(xs:string) as xs:anyURI
unparsed-entity-public-id(xs:string) as xs:string

Argument Notation

numeric Any of xs:integer, xs:decimal, xs:float or xs:double.
* A sequence of the indicated type.
? The indicated type or empty sequence.
~ The result type varies depending on the arguments.
xs: <http://www.w3.org/2001/XMLSchema>

2008-07-21

XQuery 1.0 & XPath 2.0 Functions & Operators Quick Reference

Sam Wilmott

sam@wilmott.ca

<http://www.wilmott.ca>

and

Mulberry Technologies, Inc.

17 West Jefferson Street, Suite 207

Rockville, MD 20850 USA

Phone: +1 301/315-9631

Fax: +1 301/315-8285

info@mulberrytech.com

<http://www.mulberrytech.com>



Mulberry Technologies, Inc.

© 2007-2008 Sam Wilmott and Mulberry Technologies, Inc.

Date/Time Operators

(xs:date) + (xs:dayTimeDuration) as xs:date
(xs:date) + (xs:yearMonthDuration) as xs:date
(xs:dateTime) + (xs:dayTimeDuration) as xs:dateTime
(xs:dateTime) + (xs:yearMonthDuration) as xs:dateTime
(xs:dayTimeDuration) + (xs:dayTimeDuration) as xs:dayTimeDuration
(xs:time) + (xs:dayTimeDuration) as xs:time
(xs:yearMonthDuration) + (xs:yearMonthDuration) as xs:yearMonthDuration
(xs:date) - (xs:date) as xs:dayTimeDuration
(xs:date) - (xs:dayTimeDuration) as xs:date
(xs:date) - (xs:yearMonthDuration) as xs:date
(xs:dateTime) - (xs:dateTime) as xs:dayTimeDuration
(xs:dateTime) - (xs:dayTimeDuration) as xs:dateTime
(xs:dateTime) - (xs:yearMonthDuration) as xs:dateTime
(xs:dayTimeDuration) - (xs:dayTimeDuration) as xs:dayTimeDuration
(xs:time) - (xs:dayTimeDuration) as xs:time
(xs:time) - (xs:time) as xs:dayTimeDuration
(xs:yearMonthDuration) - (xs:yearMonthDuration) as xs:yearMonthDuration
(xs:dayTimeDuration) * (xs:double) as xs:dayTimeDuration
(xs:yearMonthDuration) * (xs:double) as xs:yearMonthDuration
(xs:dayTimeDuration) div (xs:dayTimeDuration) as xs:decimal
(xs:dayTimeDuration) div (xs:double) as xs:dayTimeDuration
(xs:yearMonthDuration) div (xs:double) as xs:yearMonthDuration
(xs:yearMonthDuration) div (xs:yearMonthDuration) as xs:decimal
The **eq**, **ne**, **lt**, **gt**, **le** and **ge** comparisons are supported for the types: **xs:date** and **xs:time**.
The **eq** and **ne** (only) comparisons are supported for the types: **xs:duration**, **xs:gDay**, **xs:gMonth**, **xs:gMonthDay**, **xs:gYear** and **xs:gYearMonth**.
The **lt**, **gt**, **le** and **ge** (only) comparisons are supported for the types: **xs:dayTimeDuration** and **xs:yearMonthDuration**.

Other Comparisons

The **eq** and **ne** (only) comparisons are supported for the types: **xs:base64Binary**, **xs:hexBinary**, **xs:NOTATION** and **xs:QName**.