HORIZONTAL AXIS

attribute

The attribute axis contains the attributes of the context node; the axis will be empty unless the context node is an element.

Examples:

?? **attribute::name** selects the name attribute of the context node [@name]

?? **attribute::*** selects all the attributes of the context node [@*]

following

The following axis contains all nodes in the same document as the context node that are after the context node in document order, excluding any descendants and excluding attribute nodes and namespace nodes.

following-sibling

The following-sibling axis contains all the following siblings of the context node; if the context node is an attribute node or namespace node, the following-sibling axis is empty.

Example:

?? **following-sibling::chapter[position()=1]** selects the next chapter sibling of the context node

namespace

The namespace axis contains the namespace nodes of the context node; the axis will be empty unless the context node is an element.

preceding

The preceding axis contains all nodes in the same document as the context node that are before the context node in document order, excluding any ancestors and excluding attribute nodes and namespace nodes.

preceding-sibling

The preceding-sibling axis contains all the preceding siblings of the context node; if the context node is an attribute node or namespace node, the preceding-sibling axis is empty.

Example:

?? **preceding-sibling::chapter[position()=1]** selects the previous chapter sibling of the context node

VERTICAL AXIS

ancestor

The ancestor axis contains the ancestors of the context node; the ancestors of the context node consist of the parent of context node and the parent's parent and so on; thus, the ancestor axis will always include the root node, unless the context node is the root node.

Example:

?? **ancestor::div** selects all div ancestors of the context node

ancestor-or-self

The ancestor-or-self axis contains the context node and the ancestors of the context node; thus, the ancestor axis will always include the root node.

Example:

?? **ancestor-or-self::div** selects the div ancestors of the context node and, if the context node is a div element, the context node as well

child

The child axis contains the children of the context node.

Examples:

?? **child::para** selects the para element children of the context node [para]

?? **child::*** selects all element children of the context node [*]

?? **child::text()** selects all text node children of the context node [text()]

?? **child::node()** selects all the children of the context node, whatever their node type

?? **child::chapter/descendant::para** selects the para element descendants of the chapter element children of the context node [chapter//para]

?? **child::*/child::para** selects all para grandchildren of the context node [*/para]

?? **child::para[position()=1]** selects the first para child of the context node [para[position()=1]]

?? **child::para[position()=last()]** selects the last para child of the context node [para[position()=last()]]

?? **child::para[position()=last()-1]** selects the last but one para child of the context node [para[position()=last()-1]]

?? **child::para[position()>1]** selects all the para children of the context node other than the first para child of the context node [para[position()>1]]

?? **/child::doc/child::chapter[position()=5]/child::section[position()=2]** selects the second section of the fifth chapter of the doc document element [/doc/chapter[5]/section[2]]

?? **child::para[attribute::type="warning"]** selects all para children of the context node that have a type attribute with value warning [para[@type="warning"]]

?? **child::para[attribute::type='warning'][position()=5]** selects the fifth para child of the context node that has a type attribute with value warning [para[@type="warning"][5]]

?? **child::para[position()=5][attribute::type="warning"]** selects the fifth para child of the context node if that child has a type attribute with value warning [para[5] [@type="warning"]]

?? **child::chapter[child::title='Introduction']** selects the chapter children of the context node that have one or more title children with string-value equal to Introduction [chapter[title="Introduction"]]

?? **child::chapter[child::title]** selects the chapter children of the context node that have one or more title children [chapter[title]]

?? **child::*[self::chapter or self::appendix]** selects the chapter and appendix children of the context node [chapter|children]

?? **child::*[self::chapter or self::appendix][position()=last()]** selects the last chapter or appendix child of the context node [[chapter|appendix][position()=last()]]

descendant

The descendant axis contains the descendants of the context node; a descendant is a child or a child of a child and so on; thus the descendant axis never contains attribute or namespace nodes.

Example:

?? **descendant::para** selects the para element descendants of the context node [.//para]

?? **/descendant::para** selects all the para elements in the same document as the context node [//para]

?? **/descendant::olist/child::item** selects all the item elements that have an olist parent and that are in the same document as the context node [//olist/item]

?? **/descendant::figure[position()=42]** selects the forty-second figure element in the document [//figure[position()=42]]

descendant-or-self

The descendant-or-self axis contains the context node and the descendants of the context node.

Example:

?? **descendant-or-self::para** selects the para element descendants of the context node and, if the context node is a para element, the context node as well

parent

The parent axis contains the parent of the context node, if there is one.

Example:

?? **parent::** select the parent element [..]

?? / selects the document root (which is always the parent of the document element) [//]

self

The self axis contains just the context node itself.

Example:

?? **self::para** selects the context node if it is a para element, and otherwise selects nothing [.]

NODE SET FUNCTIONS

number count**(node-set?)** **§ 4.1**

The count function returns the number of nodes in the argument node-set.

node-set id**(object)** **§ 4.1**

The id function selects elements by their unique ID.

number last**()** **§ 4.1**

The last function returns a number equal to the context size from the expression evaluation context.

string local-name**(node-set?)** **§ 4.1**

The local-name function returns the local part of the expanded-name of the node in the argument node-set that is first in document order.

string name**(node-set?)** **§ 4.1**

The name function returns a string containing a QName representing the expanded-name of the node in the argument node-set that is first in document order.

string namespace-uri**(node-set?)** **§ 4.1**

The namespace-uri function returns the namespace URI of the expanded-name of the node in the argument node-set that is first in document order.

number position**()** **§ 4.1**

The position function returns a number equal to the context position from the expression evaluation context.

STRING FUNCTIONS

string concat**(string, string, string*)** **§ 4.2**

The concat function returns the concatenation of its arguments.

boolean contains**(string, string)** **§ 4.2**

The contains function returns true if the first argument string contains the second argument string, and otherwise returns false.

string normalize-space**(string?)** **§ 4.2**

The normalize-space function returns the argument string with whitespace normalized by stripping leading and trailing whitespace and replacing sequences of whitespace characters by a single space.

boolean starts-with**(string, string)** **§ 4.2**

The starts-with function returns true if the first argument string starts with the second argument string, and otherwise returns false.

string string**(object?)** **§ 4.2**

The string function converts an object to a string:

?? NaN is converted to the string NaN,
?? positive zero is converted to the string 0,
?? negative zero is converted to the string 0,
?? positive infinity is converted to the string Infinity,
?? negative infinity is converted to the string –Infinity,
?? if the number is an integer, the number is represented in decimal form as a Number with no decimal point and no leading zeros, preceded by a minus sign (-) if the number is negative,
?? otherwise, the number is represented in decimal form as a Number including a decimal point with at least one digit before the decimal point and at least one digit after the decimal point, preceded by a minus sign (-) if the number is negative.

number string-length**(string?)** **§ 4.2**

The string-length returns the number of characters in the string

string substring**(string, number, number?)** **§ 4.2**

The substring function returns the substring of the first argument starting at the position specified in the second argument with length specified in the third argument.

string substring-after**(string, string)** **§ 4.2**

The substring-after function returns the substring of the first argument string that follows the first occurrence of the second argument string in the first argument string, or the empty string if the first argument string does not contain the second argument string.

string substring-before**(string, string)** **§ 4.2**

The substring-before function returns the substring of the first argument string that precedes the first occurrence of the second argument string in the first argument string, or the empty string if the first argument string does not contain the second argument string.

string translate**(string, string, string)** **§ 4.2**

The translate function returns the first argument string with occurrences of characters in the second argument string replaced by the character at the corresponding position in the third argument string.

BOOLEAN FUNCTIONS

boolean boolean**(object)** **§ 4.3**

The boolean function converts its argument to a Boolean:

?? a number is true if and only if it is neither positive or negative zero nor NaN,
?? a node-set is true if and only if it is non-empty,
?? a string is true if and only if its length is non-zero,
?? an object of a type other than the four basic types is converted to a boolean in a way that is dependent on that type.

boolean false**()** **§ 4.3**

The false function returns false.

boolean lang**(string)** **§ 4.3**

The lang function returns true or false depending on whether the language of the context node as specified by *xml:lang* attributes is the same as or is a sublanguage of the language specified by the argument string.

boolean not**(boolean)** **§ 4.3**

The not function returns true if its argument is false, and false otherwise.

boolean true**()** **§ 4.3**

The true function returns true.

NUMBER FUNCTIONS

number ceiling**(number)** **§ 4.4**

The ceiling function returns the smallest (closest to negative infinity) number that is not less than the argument and that is an integer.

number floor**(number)** **§ 4.4**

The floor function returns the largest (closest to positive infinity) number that is not greater than the argument and that is an integer.

number number**(object?)** **§ 4.4**

The number function converts its argument to a number:
a string that consists of optional whitespace followed by an optional minus sign followed by a Number followed by whitespace is converted to the IEEE 754 number that is nearest (according to the IEEE 754 round-to-nearest rule) to the mathematical value represented by the string; any other string is converted to NaN,

?? boolean true is converted to 1; boolean false is converted to 0,
?? a node-set is first converted to a string as if by a call to the string function and then converted in the same way as a string argument,
?? an object of a type other than the four basic types is converted to a number in a way that is dependent on that type.

number round**(number)** **§ 4.4**

The round function returns the number that is closest to the argument and that is an integer.

number sum**(node-set)** **§ 4.4**

The sum function returns the sum, for each node in the argument node-set, of the result of converting the string-values of the node to a number.

Quick Reference

XML Path Language (XPath)

Version 1.0

W3C Recommendation
16 November 1999

http://www.w3.org/TR/xpath/

Table of Contents:

deepX Ltd.

Dublin, Ireland

info@deepX.com
http://www.deepX.com/