XPath, XQuery, and XSLT Functions

http://www.w3schools.com/xpath/xpath_functions.asp

The following reference library defines the functions required for XPath 2.0, XQuery 1.0 and XSLT 2.0.

Functions Reference

- Accessor
- Error and Trace
- Numeric
- String
- AnyURI
- Boolean
- Duration/Date/Time
- QName
- Node
- Sequence
- Context

The default prefix for the function namespace is fn:, and the URI is: http://www.w3.org/2005/02/xpath-functions.

Accessor Functions

Name	Description
fn:node-name(*node*)	Returns the node-name of the argument node
fn:nilled(*node*)	Returns a Boolean value indicating whether the argument node is nilled
fn:data(*item.item,...*)	Takes a sequence of items and returns a sequence of atomic values
fn:base-uri() fn:base-uri(*node*)	Returns the value of the base-uri property of the current or specified node
fn:document-uri(*node*)	Returns the value of the document-uri property for the specified node

Error and Trace Functions

Name	Description
fn:error() fn:error(*error*) fn:error(*error,description*) fn:error(*error,description,error-object*)	Example: error(fn:QName('http://example.com/test', 'err:toohigh'), 'Error: Price is too high') Result: Returns http://example.com/test#toohigh and the string "Error: Price is too high" to the external processing environment
fn:trace(*value,label*)	Used to debug queries

Functions on Numeric Values

Name	Description
fn:number(*arg*)	Returns the numeric value of the argument. The argument could be a boolean, string, or node-set Example: number('100') Result: 100
fn:abs(*num*)	Returns the absolute value of the argument Example: abs(3.14) Result: 3.14 Example: abs(-3.14) Result: 3.14

fn:ceiling(*num*)	Returns the smallest integer that is greater than the number argument Example: ceiling(3.14) Result: 4
fn:floor(*num*)	Returns the largest integer that is not greater than the number argument Example: floor(3.14) Result: 3
fn:round(*num*)	Rounds the number argument to the nearest integer Example: round(3.14) Result: 3
fn:round-half-to-even()	Example: round-half-to-even(0.5) Result: 0 Example: round-half-to-even(1.5) Result: 2 Example: round-half-to-even(2.5) Result: 2

Functions on Strings

Name	**Description**
fn:string(*arg*)	Returns the string value of the argument. The argument could be a number, boolean, or node-set Example: string(314) Result: "314"
fn:codepoints-to-string(*int,int,...*)	Returns a string from a sequence of code points Example: codepoints-to-string(84, 104, 233, 114, 232, 115, 101) Result: 'Thérèse'
fn:string-to-codepoints(*string*)	Returns a sequence of code points from a string Example: string-to-codepoints("Thérèse") Result: 84, 104, 233, 114, 232, 115, 101
fn:codepoint-equal(*comp1,comp2*)	Returns true if the value of comp1 is equal to the value of comp2, according to the Unicode code point collation (http://www.w3.org/2005/02/xpath-functions/collation/codepoint), otherwise it returns false
fn:compare(*comp1,comp2*) fn:compare(*comp1,comp2,collation*)	Returns -1 if comp1 is less than comp2, 0 if comp1 is equal to comp2, or 1 if comp1 is greater than comp2 (according to the rules of the collation that is used) Example: compare('ghi', 'ghi') Result: 0
fn:concat(*string,string,...*)	Returns the concatenation of the strings Example: concat('XPath ','is ','FUN!') Result: 'XPath is FUN!'
fn:string-join(*(string,string,...),sep*)	Returns a string created by concatenating the string arguments and using the sep argument as the separator Example: string-join(('We', 'are', 'having', 'fun!'), ' ') Result: ' We are having fun! '

	Example: string-join(('We', 'are', 'having', 'fun!')) Result: 'Wearehavingfun!' Example:string-join((), 'sep') Result: ''
fn:substring(*string*,*start*,*len*) fn:substring(*string*,*start*)	Returns the substring from the start position to the specified length. Index of the first character is 1. If length is omitted it returns the substring from the start position to the end Example: substring('Beatles',1,4) Result: 'Beat' Example: substring('Beatles',2) Result: 'eatles'
fn:string-length(*string*) fn:string-length()	Returns the length of the specified string. If there is no string argument it returns the length of the string value of the current node Example: string-length('Beatles') Result: 7
fn:normalize-space(*string*) fn:normalize-space()	Removes leading and trailing spaces from the specified string, and replaces all internal sequences of white space with one and returns the result. If there is no string argument it does the same on the current node Example: normalize-space(' The XML ') Result: 'The XML'
fn:normalize-unicode()	
fn:upper-case(*string*)	Converts the string argument to upper-case Example: upper-case('The XML') Result: 'THE XML'
fn:lower-case(*string*)	Converts the string argument to lower-case Example: lower-case('The XML') Result: 'the xml'
fn:translate(*string1*,*string2*,*string3*)	Converts string1 by replacing the characters in string2 with the characters in string3 Example: translate('12:30','30','45') Result: '12:45' Example: translate('12:30','03','54') Result: '12:45' Example: translate('12:30','0123','abcd') Result: 'bc:da'
fn:escape-uri(*stringURI*,*esc-res*)	Example: escape-uri("http://example.com/test#car", true()) Result: "http%3A%2F%2Fexample.com%2Ftest#car" Example: escape-uri("http://example.com/test#car", false()) Result: "http://example.com/test#car" Example: escape-uri ("http://example.com/~bébé", false()) Result: "http://example.com/~b%C3%A9b%C3%A9"
fn:contains(*string1*,*string2*)	Returns true if string1 contains string2, otherwise it returns false Example: contains('XML','XM') Result: true

fn:starts-with(*string1*,*string2*)	Returns true if string1 starts with string2, otherwise it returns false Example: starts-with('XML','X') Result: true
fn:ends-with(*string1*,*string2*)	Returns true if string1 ends with string2, otherwise it returns false Example: ends-with('XML','X') Result: false
fn:substring-before(*string1*,*string2*)	Returns the start of string1 before string2 occurs in it Example: substring-before('12/10','/') Result: '12'
fn:substring-after(*string1*,*string2*)	Returns the remainder of string1 after string2 occurs in it Example: substring-after('12/10','/') Result: '10'
fn:matches(*string*,*pattern*)	Returns true if the string argument matches the pattern, otherwise, it returns false Example: matches("Merano", "ran") Result: true
fn:replace(*string*,*pattern*,*replace*)	Returns a string that is created by replacing the given pattern with the replace argument Example: replace("Bella Italia", "l", "*") Result: 'Be**a Ita*ia' Example: replace("Bella Italia", "l", "") Result: 'Bea Itaia'
fn:tokenize(*string*,*pattern*)	Example: tokenize("XPath is fun", "\s+") Result: ("XPath", "is", "fun")

Functions for anyURI

Name	Description
fn:resolve-uri(*relative*,*base*)	

Functions on Boolean Values

Name	Description
fn:boolean(*arg*)	Returns a boolean value for a number, string, or node-set
fn:not(*arg*)	The argument is first reduced to a boolean value by applying the boolean() function. Returns true if the boolean value is false, and false if the boolean value is true Example: not(true()) Result: false
fn:true()	Returns the boolean value true Example: true() Result: true
fn:false()	Returns the boolean value false Example: false() Result: false

Functions on Durations, Dates and Times

Component Extraction Functions on Durations, Dates and Times

Name	Description
fn:dateTime(*date*,*time*)	Converts the arguments to a date and a time
fn:years-from-duration(*datetimedur*)	Returns an integer that represents the years component in the canonical lexical representation of the value of the argument
fn:months-from-duration(*datetimedur*)	Returns an integer that represents the months component in the canonical lexical representation of the value of the argument
fn:days-from-duration(*datetimedur*)	Returns an integer that represents the days component in the canonical lexical representation of the value of the argument
fn:hours-from-duration(*datetimedur*)	Returns an integer that represents the hours component in the canonical lexical representation of the value of the argument
fn:minutes-from-duration(*datetimedur*)	Returns an integer that represents the minutes component in the canonical lexical representation of the value of the argument
fn:seconds-from-duration(*datetimedur*)	Returns a decimal that represents the seconds component in the canonical lexical representation of the value of the argument
fn:year-from-dateTime(*datetime*)	Returns an integer that represents the year component in the localized value of the argument Example: year-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10")) Result: 2005
fn:month-from-dateTime(*datetime*)	Returns an integer that represents the month component in the localized value of the argument Example: month-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10")) Result: 01
fn:day-from-dateTime(*datetime*)	Returns an integer that represents the day component in the localized value of the argument Example: day-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10")) Result: 10
fn:hours-from-dateTime(*datetime*)	Returns an integer that represents the hours component in the localized value of the argument Example: hours-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10")) Result: 12
fn:minutes-from-dateTime(*datetime*)	Returns an integer that represents the minutes component in the localized value of the argument Example: minutes-from-dateTime(xs:dateTime("2005-01-10T12:30-04:10")) Result: 30
fn:seconds-from-dateTime(*datetime*)	Returns a decimal that represents the seconds component in the localized value of the argument Example: seconds-from-dateTime(xs:dateTime("2005-01-10T12:30:00-04:10")) Result: 0
fn:timezone-from-dateTime(*datetime*)	Returns the time zone component of the argument if any
fn:year-from-date(*date*)	Returns an integer that represents the year in the localized value of the argument Example: year-from-date(xs:date("2005-04-23")) Result: 2005

fn:month-from-date(*date*)	Returns an integer that represents the month in the localized value of the argument Example: month-from-date(xs:date("2005-04-23")) Result: 4
fn:day-from-date(*date*)	Returns an integer that represents the day in the localized value of the argument Example: day-from-date(xs:date("2005-04-23")) Result: 23
fn:timezone-from-date(*date*)	Returns the time zone component of the argument if any
fn:hours-from-time(*time*)	Returns an integer that represents the hours component in the localized value of the argument Example: hours-from-time(xs:time("10:22:00")) Result: 10
fn:minutes-from-time(*time*)	Returns an integer that represents the minutes component in the localized value of the argument Example: minutes-from-time(xs:time("10:22:00")) Result: 22
fn:seconds-from-time(*time*)	Returns an integer that represents the seconds component in the localized value of the argument Example: seconds-from-time(xs:time("10:22:00")) Result: 0
fn:timezone-from-time(*time*)	Returns the time zone component of the argument if any
fn:adjust-dateTime-to-timezone(*datetime*,*timezone*)	If the timezone argument is empty, it returns a dateTime without a timezone. Otherwise, it returns a dateTime with a timezone
fn:adjust-date-to-timezone(*date*,*timezone*)	If the timezone argument is empty, it returns a date without a timezone. Otherwise, it returns a date with a timezone
fn:adjust-time-to-timezone(*time*,*timezone*)	If the timezone argument is empty, it returns a time without a timezone. Otherwise, it returns a time with a timezone

Functions Related to QNames

Name	Description
fn:QName()	
fn:local-name-from-QName()	
fn:namespace-uri-from-QName()	
fn:namespace-uri-for-prefix()	
fn:in-scope-prefixes()	
fn:resolve-QName()	

Functions on Nodes

Name	Description
fn:name() fn:name(*nodeset*)	Returns the name of the current node or the first node in the specified node set
fn:local-name() fn:local-name(*nodeset*)	Returns the name of the current node or the first node in the specified node set - without the namespace prefix
fn:namespace-uri() fn:namespace-uri(*nodeset*)	Returns the namespace URI of the current node or the first node in the specified node set
fn:lang(*lang*)	Returns true if the language of the current node matches the language of the specified language Example: Lang("en") is true for ...

	Example: Lang("de") is false for ...
fn:root() fn:root(*node*)	Returns the root of the tree to which the current node or the specified belongs. This will usually be a document node

Functions on Sequences

General Functions on Sequences

Name	**Description**
fn:index-of(*(item,item,...),searchitem*)	Returns the positions within the sequence of items that are equal to the searchitem argument Example: index-of ((15, 40, 25, 40, 10), 40) Result: (2, 4) Example: index-of (("a", "dog", "and", "a", "duck"), "a") Result (1, 4) Example: index-of ((15, 40, 25, 40, 10), 18) Result: ()
fn:remove(*(item,item,...),position*)	Returns a new sequence constructed from the value of the item arguments - with the item specified by the position argument removed Example: remove(("ab", "cd", "ef"), 0) Result: ("ab", "cd", "ef") Example: remove(("ab", "cd", "ef"), 1) Result: ("cd", "ef") Example: remove(("ab", "cd", "ef"), 4) Result: ("ab", "cd", "ef")
fn:empty(*item,item,...*)	Returns true if the value of the arguments IS an empty sequence, otherwise it returns false Example: empty(remove(("ab", "cd"), 1)) Result: false
fn:exists(*item,item,...*)	Returns true if the value of the arguments IS NOT an empty sequence, otherwise it returns false Example: exists(remove(("ab"), 1)) Result: false
fn:distinct-values(*(item,item,...),collation*)	Returns only distinct (different) values Example: distinct-values((1, 2, 3, 1, 2)) Result: (1, 2, 3)
fn:insert-before(*(item,item,...),pos,inserts*)	Returns a new sequence constructed from the value of the item arguments - with the value of the inserts argument inserted in the position specified by the pos argument Example: insert-before(("ab", "cd"), 0, "gh") Result: ("gh", "ab", "cd") Example: insert-before(("ab", "cd"), 1, "gh") Result: ("gh", "ab", "cd") Example: insert-before(("ab", "cd"), 2, "gh")

	Result: ("ab", "gh", "cd") Example: insert-before(("ab", "cd"), 5, "gh") Result: ("ab", "cd", "gh")
fn:reverse(*(item,item,...)*)	Returns the reversed order of the items specified Example: reverse(("ab", "cd", "ef")) Result: ("ef", "cd", "ab") Example: reverse(("ab")) Result: ("ab")
fn:subsequence(*(item,item,...),start,len*)	Returns a sequence of items from the position specified by the start argument and continuing for the number of items specified by the len argument. The first item is located at position 1 Example: subsequence(($item1, $item2, $item3,...), 3) Result: ($item3, ...) Example: subsequence(($item1, $item2, $item3, ...), 2, 2) Result: ($item2, $item3)
fn:unordered(*(item,item,...)*)	Returns the items in an implementation dependent order

Functions That Test the Cardinality of Sequences

Name	**Description**
fn:zero-or-one(*item,item,...*)	Returns the argument if it contains zero or one items, otherwise it raises an error
fn:one-or-more(*item,item,...*)	Returns the argument if it contains one or more items, otherwise it raises an error
fn:exactly-one(*item,item,...*)	Returns the argument if it contains exactly one item, otherwise it raises an error

Equals, Union, Intersection and Except

Name	**Description**
fn:deep-equal(*param1,param2,collation*)	Returns true if param1 and param2 are deep-equal to each other, otherwise it returns false

Aggregate Functions

Name	**Description**
fn:count(*(item,item,...)*)	Returns the count of nodes
fn:avg(*(arg,arg,...)*)	Returns the average of the argument values Example: avg((1,2,3)) Result: 2
fn:max(*(arg,arg,...)*)	Returns the argument that is greater than the others Example: max((1,2,3)) Result: 3 Example: max(('a', 'k')) Result: 'k'
fn:min(*(arg,arg,...)*)	Returns the argument that is less than the others Example: min((1,2,3))

	Result: 1 Example: min(('a', 'k')) Result: 'a'
fn:sum(*arg,arg,...*)	Returns the sum of the numeric value of each node in the specified node-set

Functions that Generate Sequences

Name	**Description**
fn:id((*string,string,...),node*)	Returns a sequence of element nodes that have an ID value equal to the value of one or more of the values specified in the string argument
fn:idref((*string,string,...),node*)	Returns a sequence of element or attribute nodes that have an IDREF value equal to the value of one or more of the values specified in the string argument
fn:doc(*URI*)	
fn:doc-available(*URI*)	Returns true if the doc() function returns a document node, otherwise it returns false
fn:collection() fn:collection(*string*)	

Context Functions

Name	**Description**
fn:position()	Returns the index position of the node that is currently being processed Example: //book[position()<=3] Result: Selects the first three book elements
fn:last()	Returns the number of items in the processed node list Example: //book[last()] Result: Selects the last book element
fn:current-dateTime()	Returns the current dateTime (with timezone)
fn:current-date()	Returns the current date (with timezone)
fn:current-time()	Returns the current time (with timezone)
fn:implicit-timezone()	Returns the value of the implicit timezone
fn:default-collation()	Returns the value of the default collation
fn:static-base-uri()	Returns the value of the base-uri

The XSLT elements from the W3C Recommendation (XSLT Version 1.0).

http://www.w3schools.com/Xsl/xsl_w3celementref.asp

The links in the "Element" column point to attributes and more useful information about each specific element.

- **FF**: indicates the earliest version of Firefox that supports the tag
- **IE**: indicates the earliest version of Internet Explorer that supports the tag

Note: Elements supported in IE 5 may have NON-standard behavior, because IE 5 was released before XSLT became an official W3C Recommendation.

Element	**Description**	**IE**	**FF**
apply-imports	Applies a template rule from an imported style sheet	6.0	1.0
apply-templates	Applies a template rule to the current element or to the current element's child nodes	5.0	1.0
attribute	Adds an attribute	5.0	1.0

attribute-set	Defines a named set of attributes	6.0	1.0
call-template	Calls a named template	6.0	1.0
choose	Used in conjunction with <when> and <otherwise> to express multiple conditional tests	5.0	1.0
comment	Creates a comment node in the result tree	5.0	1.0
copy	Creates a copy of the current node (without child nodes and attributes)	5.0	1.0
copy-of	Creates a copy of the current node (with child nodes and attributes)	6.0	1.0
decimal-format	Defines the characters and symbols to be used when converting numbers into strings, with the format-number() function	6.0	1.0
element	Creates an element node in the output document	5.0	1.0
fallback	Specifies an alternate code to run if the processor does not support an XSLT element	6.0	
for-each	Loops through each node in a specified node set	5.0	1.0
if	Contains a template that will be applied only if a specified condition is true	5.0	1.0
import	Imports the contents of one style sheet into another. **Note:** An imported style sheet has lower precedence than the importing style sheet	6.0	1.0
include	Includes the contents of one style sheet into another. **Note:** An included style sheet has the same precedence as the including style sheet	6.0	1.0
key	Declares a named key that can be used in the style sheet with the key() function	6.0	1.0
message	Writes a message to the output (used to report errors)	6.0	1.0
namespace-alias	Replaces a namespace in the style sheet to a different namespace in the output	6.0	
number	Determines the integer position of the current node and formats a number	6.0	1.0
otherwise	Specifies a default action for the <choose> element	5.0	1.0
output	Defines the format of the output document	6.0	1.0
param	Declares a local or global parameter	6.0	1.0
preserve-space	Defines the elements for which white space should be preserved	6.0	1.0
processing-instruction	Writes a processing instruction to the output	5.0	1.0
sort	Sorts the output	6.0	1.0
strip-space	Defines the elements for which white space should be removed	6.0	1.0
stylesheet	Defines the root element of a style sheet	5.0	1.0
template	Rules to apply when a specified node is matched	5.0	1.0
text	Writes literal text to the output	5.0	1.0
transform	Defines the root element of a style sheet	6.0	1.0
value-of	Extracts the value of a selected node	5.0	1.0
variable	Declares a local or global variable	6.0	1.0
when	Specifies an action for the <choose> element	5.0	1.0
with-param	Defines the value of a parameter to be passed into a template	6.0	1.0

XSLT <xsl:template> Element

Attributes

Attribute	Value	Description
name	name	Optional. Specifies a name for the template. **Note:** If this attribute is omitted there must be a match attribute
match	pattern	Optional. The match pattern for the template. **Note:** If this attribute is omitted there must be a name attribute
mode	mode	Optional. Specifies a mode for this template
priority	number	Optional. A number which indicates the numeric priority of the template

Example

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
  <h2>My CD Collection</h2>
  <xsl:apply-templates/>
  </body>
  </html>
</xsl:template>

<xsl:template match="cd">
  <p>
  <xsl:apply-templates select="title"/>
  <xsl:apply-templates select="artist"/>
  </p>
</xsl:template>

<xsl:template match="title">
  Title: <span style="color:#ff0000">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

<xsl:template match="artist">
  Artist: <span style="color:#00ff00">
  <xsl:value-of select="."/></span>
  <br />
</xsl:template>

</xsl:stylesheet>
```

XSLT <xsl:apply-templates> Element

Attributes

Attribute	Value	Description
select	expression	Optional. Specifies the nodes to be processed. An asterisk selects the entire node-set. If this attribute is omitted, all child nodes of the current node will be selected
mode	name	Optional. If there are multiple ways of processing defined for the same element, distinguishes among them

Example 1

Wrap a single h1 element around each title element in the document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="title">
  <h1><xsl:apply-templates/></h1>
</xsl:template>

</xsl:stylesheet>
```

XSLT <xsl:choose> Element

Attributes

None

XSLT <xsl:for-each> Element

Attributes

Attribute	Value	Description
select	expression	Required. The node set to be processed

Example 1

Loop through each "cd "element and use <xsl:value-of> to write each title and artist to the output:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <tr>
        <td><xsl:value-of select="title"/></td>
        <td><xsl:value-of select="artist"/></td>
      </tr>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>
```

XSLT <xsl:if> Element

Attributes

Attribute	Value	Description
test	expression	Required. Specifies the condition to be tested

Example 1

Select the values of title and artist IF the price of the CD is higher than 10:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <html>
  <body>
    <h2>My CD Collection</h2>
    <table border="1">
      <tr bgcolor="#9acd32">
        <th>Title</th>
        <th>Artist</th>
      </tr>
      <xsl:for-each select="catalog/cd">
      <xsl:if test="price &gt; 10">
        <tr>
          <td><xsl:value-of select="title"/></td>
          <td><xsl:value-of select="artist"/></td>
        </tr>
      </xsl:if>
      </xsl:for-each>
    </table>
  </body>
  </html>
</xsl:template>

</xsl:stylesheet>
```

XSLT <xsl:call-template> Element

Attributes

Attribute	Value	Description
name	templatename	Required. Specifies the name of the template to be called

Example 1

Call a template named "description" when the processor finds a car element:

```
<xsl:template match="car">
  <xsl:call-template name="description"/>
</xsl:template>
```

XSLT <xsl:element> Element

Attributes

Attribute	Value	Description
name	name	Required. Specifies the name of the element to be created (the value of the name attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" />
namespace	URI	Optional. Specifies the namespace URI of the element (the value of the namespace attribute can be set to an expression that is computed at run-time, like this: <xsl:element name="{$country}" namespace="{$someuri}"/>
use-attribute-sets	namelist	Optional. A white space separated list of attribute-sets containing attributes to be added to the element

Example 1

Create a "singer" element that contains the value of each artist element:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
```

```
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="/">
  <xsl:for-each select="catalog/cd">
    <xsl:element name="singer">
      <xsl:value-of select="artist" />
    </xsl:element>
    <br />
  </xsl:for-each>
</xsl:template>

</xsl:stylesheet>
```

XSLT <xsl:attribute> Element

Attributes

Attribute	Value	Description
name	attributename	Required. Specifies the name of the attribute
namespace	URI	Optional. Defines the namespace URI for the attribute

Example 1

Add a source attribute to the picture element:

```
<picture>
  <xsl:attribute name="source"/>
</picture>
```