

Location Paths [XPath §2]

Optional '/', zero or more **location steps**, separated by '/'

Location Steps [XPath §2.1]

Axis specifier, **node test**, zero or more **predicates**

Axis Specifiers [XPath §2.2]

ancestor::	following-sibling::
ancestor-or-self::	namespace::
attribute::	parent::
child::	preceding::
descendant::	preceding-sibling::
descendant-or-self::	self::
following::	

Node Tests [XPath §2.3]

<i>name</i>	node()
<i>URI:name</i>	text()
<i>prefix:name</i>	comment()
*	processing-instruction()
<i>prefix:*</i>	processing-instruction(<i>literal</i>)

Abbreviated Syntax for Location Paths

(nothing)	child::
@	attribute::
//	/descendant-or-self::node()/
.	self::node()
..	parent::node()
/	Node tree root

Predicate [XPath §2.4]

[*expr*]

Variable Reference [XPath §3.7]

\$qname

Literal Result Elements [§7.1.1]

Any element not in the xsl: namespace and not an extension element

XSLT

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

XPath

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

XSL-List

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

XPath Operators

Parentheses may be used for grouping.

Node-sets [XPath §3.3]

| [expr] / //

Booleans [XPath §3.4]

<=, <, >=, > =, != and or

Numbers [XPath §3.5]

-*expr* *, div, mod +, -

XPath Core Function Library

Node Set Functions [XPath §4.1]

number last()
number position()
number count(*node-set*)
node-set id(*object*)
string local-name(*node-set*?)
string namespace-uri(*node-set*?)
string name(*node-set*?)

String Functions [XPath §4.2]

string string(*object*?)
string concat(*string*, *string*, *string**)
boolean starts-with(*string*, *string*)
boolean contains(*string*, *string*)
string substring-before(*string*, *string*)
string substring-after(*string*, *string*)
string substring(*string*, *number*, *number*?)
number string-length(*string*?)
string normalize-space(*string*?)
string translate(*string*, *string*, *string*)

Boolean Functions [XPath §4.3]

boolean boolean(*object*)
boolean not(*object*)
boolean true()
boolean false()
boolean lang(*string*)

Number Functions [XPath §4.4]

number number(*object*?)
number sum(*node-set*)
number floor(*number*)
number ceiling(*number*)
number round(*number*)

XSLT and XPath Quick Reference

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.

XSLT Functions [§12, §15]

node-set document(*object*, *node-set*?)
node-set key(*string*, *object*)
string format-number(*number*, *string*, *string*?)
node-set current()
string unparsed-entity-uri(*string*)
string generate-id(*node-set*?)
object system-property(*string*)
boolean element-available(*string*)
boolean function-available(*string*)

Node Types [XPath §5]

Root	Processing Instruction
Element	Comment
Attribute	Text
Namespace	

Object Types [§11.1, XPath §1]

boolean	True or false
number	Floating-point number
string	UCS characters
node-set	Set of nodes selected by a path
Result tree fragment	XSLT only. Fragment of the result tree

Expression Context [§4, XPath §1]

Context node (a node)

Context position (a number)

Context size (a number)

Variable bindings in scope

Namespace declarations in scope

Function library

Built-in Template Rules [§5.8]

```
<xsl:template match="*/" />
  <xsl:apply-templates/>
</xsl:template>

<xsl:template match="*/" mode="m">
  <xsl:apply-templates mode="m" />
</xsl:template>

<xsl:template match="text()|@"*>
  <xsl:value-of select="."/>
</xsl:template>

<xsl:template
  match="processing-instruction()|comment()" />
```

Built-in template rule for namespaces is to do nothing



Mulberry Technologies, Inc.

© 2000 Mulberry Technologies, Inc.



Mulberry Technologies, Inc.

© 2000 Mulberry Technologies, Inc. (20010522)



Mulberry Technologies, Inc.

© 2000 Mulberry Technologies, Inc.

XSLT Elements

Stylesheet Element [§2.2]

```
<xsl:stylesheet version="1.0" id="{id}"
  extension-element-prefixes="{tokens}"
  exclude-result-prefixes="{tokens}"
  xmlns:xsl="http://www.w3.org/1999/XSL/
  Transform"> xsl:import *, top-level elements
</xsl:stylesheet>
```

xsl:transform is a synonym for xsl:stylesheet

Combining Stylesheets [§2.6]

```
<xsl:include href="{uri-reference}"/>
<xsl:import href="{uri-reference}"/>
```

Whitespace Stripping [§3.4]

```
<xsl:strip-space elements="{tokens}"/>
<xsl:preserve-space elements="{tokens}"/>
```

Defining Template Rules [§5.3]

```
<xsl:template match="{pattern}" name="{qname}"
  priority="{number}" mode="{qname}">
  xsl:param* followed by text, literal result elements
  and/or XSL elements </xsl:template>
```

Applying Template Rules [§5.4]

```
<xsl:apply-templates select="{node-set-exp}"
  mode="{qname}"/>
<xsl:apply-templates select="{node-set-exp}"
  mode="{qname}">
  (xsl:sort | xsl:with-param)* </xsl:apply-templates>
```

Overriding Template Rules [§5.6]

```
<xsl:apply-imports/>
```

Named Templates [§6]

```
<xsl:call-template name="{qname}"/>
<xsl:call-template name="{qname}">
  xsl:with-param* </xsl:call-template>
```

Namespace Alias [§7.1.1]

```
<xsl:namespace-alias result-prefix="{prefix}#default"
  stylesheet-prefix="{prefix}#default"/>
```

Creating Elements [§7.1.2]

```
<xsl:element name="{qname}"
  namespace="{uri-reference}"
  use-attribute-sets="{qnames}">...</xsl:element>
```

Creating Attributes [§7.1.3]

```
<xsl:attribute name="{qname}"
  namespace="{uri-reference}">...</xsl:attribute>
```

Named Attribute Sets [§7.1.4]

```
<xsl:attribute-set name="{qname}"
  use-attribute-sets="{qnames}">
  xsl:attribute* </xsl:attribute-set>
```

Creating Text [§7.2]

```
<xsl:text disable-output-escaping="{yes|no}">
  #PCDATA </xsl:text>
```

Processing Instructions [§7.3]

```
<xsl:processing-instruction name="{ncname}">
  ...</xsl:processing-instruction>
```

Creating Comments [§7.4]

```
<xsl:comment>...</xsl:comment>
```

Copying [§7.5]

```
<xsl:copy use-attribute-sets="{qnames}">
  ...</xsl:copy>
```

Generating Text [§7.6.1]

```
<xsl:value-of select="{string-expr}"
  disable-output-escaping="{yes|no}"/>
```

Attribute Value Templates [§7.6.2]

```
<element attribute="{expr}"/>
```

Numbering [§7.7]

```
<xsl:number level="{single|multiple|any}"
  count="{pattern}" from="{pattern}"
  value="{number-expr}" format="{string}"
  lang="{nmtoken}"
  letter-value="{alphabetic|traditional}"
  grouping-separator="{char}"
  grouping-size="{number}"/>
```

Repetition [§8]

```
<xsl:for-each select="{node-set-expr}">
  xsl:sort*, ...</xsl:for-each>
```

Conditional Processing [§9]

```
<xsl:if test="{boolean-expr}">...</xsl:if>
<xsl:choose>
  <xsl:when test="{expr}">...</xsl:when>+
  <xsl:otherwise>...</xsl:otherwise?>
</xsl:choose>
```

Sorting [§10]

```
<xsl:sort select="{string-expr}" lang="{nmtoken}"
  data-type="{text|number|qname-but-not-
  ncname}" order="{ascending|descending}"
  case-order="{upper-first|lower-first}"/>
```

Variables and Parameters [§11]

```
<xsl:variable name="{qname}" select="{expr}"/>
<xsl:variable name="{qname}">...</xsl:variable>
<xsl:param name="{qname}" select="{expr}"/>
<xsl:param name="{qname}">...</xsl:param>
```

Using Values [§11.3]

```
<xsl:copy-of select="{expr}"/>
```

Passing Parameters [§11.6]

```
<xsl:with-param name="{expr}" select="{expr}"/>
<xsl:with-param name="{expr}">...</xsl:with-param>
```

Keys [§12.2]

```
<xsl:key name="{qname}" match="{pattern}"
  use="{expr}"/>
```

Number Formatting [§12.3]

```
<xsl:decimal-format name="{qname}"
  decimal-separator="{char}"
  grouping-separator="{char}" infinity="{string}"
  minus-sign="{char}" NaN="{string}"
  percent="{char}" per-mille="{char}"
  zero-digit="{char}" digit="{char}"
  pattern-separator="{char}"/>
```

Messages [§13]

```
<xsl:message terminate="{yes|no}">
  ...</xsl:message>
```

Fallback [§15]

```
<xsl:fallback>...</xsl:fallback>
```

Output [§16]

```
<xsl:output
  method="{xml|html|text|qname-but-not-ncname}"
  version="{nmtoken}" encoding="{string}"
  omit-xml-declaration="{yes|no}"
  doctype-public="{string}" doctype-system="{string}"
  standalone="{yes|no}" indent="{yes|no}"
  cdata-section-elements="{qnames}"
  media-type="{string}"/>
```

Key

xsl:stylesheet	Element
version=	Required attribute
version=	Optional attribute
{ <i>expr</i> }	Attribute value template. Text between any { and } is evaluated as an expression. Attribute value must evaluate to indicated attribute type.
...	Anything allowed in a template
	Separates alternative values
?	Zero or one occurrences
*	Zero or more occurrences
+	One or more occurrences
#PCDATA	Character data

Attribute Value Types

1.0	Literal value
<i>boolean-expr</i>	Expression returning boolean value
<i>char</i>	Single character
<i>expr</i>	Expression
<i>id</i>	XML name used as identifier
<i>ncname</i>	XML name not containing a colon (:)
<i>node-set-expr</i>	Expression returning a node set
<i>number-expr</i>	Expression returning a number
<i>pattern</i>	XSLT pattern
<i>prefix</i>	Namespace prefix
<i>qname</i>	Namespace-qualified XML name comprising local part and optional prefix
<i>qname-but-not-ncname</i>	Namespace-qualified name comprising local part and prefix
<i>token</i>	Meaning varies with context. See Rec.
<i>uri-reference</i>	Reference to Universal Resource Identifier

