ADDITIONAL FUNCTION ELEMENTS

xsl:decimal-format § 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 />
```

xsl:key § 12.2

```
<xsl:key
  name = qname
  match = pattern
  use = expression />
```

CONDITIONAL PROCESSING ELEMENTS

xsl:choose § 9.2

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

xsl:if § 9.1

```
<xsl:if
  test = boolean-expression>
  <!-- Content: template -->
</xsl:if>
```

xsl:otherwise § 9.2

```
<xsl:otherwise>
  <!-- Content: template -->
</xsl:otherwise>
```

xsl:when § 9.2

```
<xsl:when
  test = boolean-expression>
  <!-- Content: template -->
</xsl:when>
```

CREATING RESULT-TREE ELEMENTS

xsl:attribute § 7.1.3

```
<xsl:attribute
  name = { qname }
  namespace = { uri-reference }>
  <!-- Content: template -->
</xsl:attribute>
```

xsl:attribute-set § 7.1.4

```
<xsl:attribute-set
  name = qname
  use-attribute-sets = qnames>
  <!-- Content: xsl:attribute* -->
</xsl:attribute-set>
```

xsl:comment § 7.4

```
<xsl:comment>
  <!-- Content: template -->
</xsl:comment>
```

xsl:copy § 7.5

```
<xsl:copy
  use-attribute-sets = qnames>
  <!-- Content: template -->
</xsl:copy>
```

xsl:element § 7.1.4

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

xsl:namespace-alias § 7.1

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

xsl:number § 7.7

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

xsl:processing-instruction § 7.3

```
<xsl:processing-instruction
  name = { ncname }>
  <!-- Content: template -->
</xsl:processing-instruction>
```

xsl:text § 7.2

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

xsl:value-of § 7.6.1

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

DATA MODEL ELEMENTS

xsl:preserve-space § 3.3

```
<xsl:preserve-space
  elements = tokens />
```

xsl:strip-space § 3.3

```
<xsl:strip-space
  elements = tokens />
```

FALLBACK ELEMENT

xsl:fallback § 15

```
<xsl:fallback>
  <!-- Content: template -->
</xsl:fallback>
```

MESSAGE ELEMENT

xsl:message § 13

```
<xsl:message
  terminate = "yes" | "no">
  <!-- Content: template -->
</xsl:message>
```

NAMED TEMPLATE ELEMENT

xsl:call-template § 6

```
<xsl:call-template
  name = qname>
  <!-- Content: xsl:with-param* -->
</xsl:call-template>
```

OUTPUT ELEMENT

xsl:output § 16

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

REPETITON ELEMENT

xsl:for-each § 8

```
<xsl:for-each
  select = node-set-expression>
  <!-- Content: (xsl:sort*, template) -->
</xsl:for-each>
```

SORTING ELEMENT

xsl:sort § 10

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

STYLESHEET STRUCTURE ELEMENTS

xsl:import § 2.6.2

```
<xsl:import
  href = uri-reference />
```

xsl:include § 2.6.1

```
<xsl:include
  href = uri-reference />
```

xsl:stylesheet § 2.2

```
<xsl:stylesheet
  id = id
  extension-element-prefixes = tokens
  exclude-result-prefixes = tokens
  version = number>
  <!-- Content: (xsl:import*, top-level-elements)   -->
</xsl:stylesheet>
```

xsl:transform § 2.2

```
<xsl:transform
  id = id
  extension-element-prefixes = tokens
  exclude-result-prefixes = tokens
  version = number>
  <!-- Content: (xsl:import*, top-level-elements) -->
</xsl:transform>
```

TEMPLATE RULE ELEMENTS

xsl:apply-imports § 5.6

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

xsl:apply-templates § 5.4

```
<xsl:apply-templates
  select = node-set-expression
  mode = qname>
  <!-- Content: (xsl:sort | xsl:with-param)* -->
</xsl:apply-templates>
```

xsl:template § 5.3

```
<xsl:template
  match = pattern
  name = qname
  priority = number
  mode = qname>
  <!-- Content: (xsl:param*, template) -->
</xsl:template>
```

VARIABLE/PARAMETER ELEMENTS

xsl:copy-of § 11.3

```
<xsl:copy-of
  select = expression />
```

xsl:param § 11

```
<xsl:param
  name = qname
  select = expression>
  <!-- Content: template -->
</xsl:param>
```

xsl:variable § 11

```
<xsl:variable
  name = qname
  select = expression>
  <!-- Content: template -->
</xsl:variable>
```

xsl:with-param § 11.6

```
<xsl:with-param
  name = qname
  select = expression>
  <!-- Content: template -->
</xsl:with-param>
```

FUNCTIONS

node-set **current**() § 12.4

The current function returns a node-set that has the current node as its only member.

note-set **document**(object, node-set?) § 12.1

The document function allows access to XML documents other than the main source document.

boolean **element-available**(string) § 15

The element-available function returns true if and only if the expanded-name is the name of an instruction. If the expanded-name has a namespace URI equal to the XSLT namespace URI, then it refers to an element defined by XSLT.

string **format-number**(number, string, string?) § 12.3

The format-number function converts its first argument to a string using the format pattern string specified by the second argument and the decimal-format named by the third argument, or the default decimal-format, if there is no third argument. The format pattern string is in the syntax specified by the JDK 1.1 DecimalFormat class.

boolean **function-available**(string) § 15

The function-available function returns true if and only if the expanded-name is the name of a function in the function library.

string **generate-id**(node-set?) § 12.4

The generate-id function returns a string that uniquely identifies the node in the argument node-set that is first in document order.

node-set **key**(string, object) § 12.2

The key function does for keys what the id function does for IDs.

object **system-property**(string) § 12.4

The system-property function returns an object representing the value of the system property identified by the name. If there is no such system property, the empty string should be returned.

string **unparsed-entity-uri**(string) § 12.4

The unparsed-entity-uri returns the URI of the unparsed entity with the specified name in the same document as the context node.

NOTATION

`|`	separator for alternative values
`,`	separator for consecutive values
`?`	zero-or-more repetitions
`*`	zero-or-more repetitions
`+`	one-or-more repetitions
`#PCDATA`	parsable character data
`boolean-expression`	expression returning a Boolean
`char`	represents a single character
`expression`	XPath production expression
`id`	XML name used as unique identifier within the document, special attribute type
`ncname`	non-colon-name - XML Name without colon (see also qname)
`nmtoken`	name token – mixture of XML name characters
`node-set-expression`	expression returning a node-set
`number`	represents a number
`number-expression`	expression retuning a number
`pattern`	XPath pattern
`prefix`	XML namespace prefix
`qname`	qualified name – XML name with local part and optional XML namespace prefix, separated by a colon
`string`	represents a string
`string-expression`	expression returning a string
`token`	attribute type
`uri-reference`	Universal Resource Identifier reference
`XML name`	XML name is a string beginning with a letter or one of a few punctuation characters, and continuing with letters, digits, hyphens, underscores, colons, or full stops, together known as name characters.

Quick Reference

XSL Transformations (XSLT)

Version 1.0

W3C Recommendation
16 November 1999

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

Table of Contents:

Elements

Functions

deepX Ltd.

Dublin, Ireland

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