XLST 2.0 - Quick Reference Sheet

Stylesheet structure

```
<xsl:transform                                  or xsl:stylesheet [3.6]
  id = ID
  default-validation = "preserve"|"strip"
  input-type-annotations =
    "preserve"|"strip"|"unspecified" >
    xsl:import*,
    (declaration|xsl:variable|xsl:param)*
</xsl:transform>

<xsl:include                                    [decl:3.10.2]
  href = anyURI />

<xsl:import                                     [3.10.3]
  href = anyURI />

<xsl:import-schema                              [decl:3.14]
  namespace = anyURI
  schema-location = anyURI >
    xs:schema?
</xsl:import-schema>
```

Template Rules and Functions

```
<xsl:template                                   [decl:6.1]
  match = pattern
  priority = decimal
  mode = (qname|"#default")+|"#all"
  name = QName
  as = sequence-type >
    xsl:param*, sequence-constructor*
</xsl:template>

<xsl:apply-templates                            [inst:6.3]
  select = expression
  mode = qname|"#default"|"#current" >
    (xsl:sort|xsl:with-param)*
</xsl:apply-templates>

<xsl:call-template                              [inst:10.1]
  name = QName >
    xsl:with-param*
</xsl:call-template>

<xsl:function                                   [decl:10.3]
  name = QName
  override = "yes"|"no"
  as = sequence-type >
    xsl:param*, sequence-constructor*
</xsl:function>

<xsl:apply-imports>                             override rules [inst:6.7]
    xsl:with-param*
</xsl:apply-imports>

<xsl:next-match>                                call to overridden [inst:6.7]
    (xsl:with-param|xsl:fallback)*
</xsl:next-match>
```

Node Creation

```
<xsl:element                                    [inst:11.2]
  name = {string}
  namespace = {string}
  inherit-namespaces = "yes"|"no"
  use-attribute-sets = QNames
  type = QName
  validation = "strict"|"lax"|"preserve"|"strip" >
    sequence-constructor
</xsl:element>

<xsl:attribute                                  [inst:11.3]
  name = {string}
  namespace = {string}
  select = expression
  separator = {string}
  type = QName
  validation = "strict"|"lax"|"preserve"|"strip" >
    sequence-constructor
</xsl:attribute>

<xsl:comment                                    [inst:11.8]
  select = expression >
    sequence-constructor
</xsl:comment>

<xsl:document                                   [inst:11.5]
  validation = "strict"|"lax"|"preserve"|"strip"
  type = QName >
    sequence-constructor
</xsl:document>

<xsl:namespace                                  [inst:11.7]
  name = {string}
  select = expression >
    sequence-constructor
</xsl:namespace>

<xsl:processing-instruction                     [inst:11.6]
  name = {string}
  select = expression >
    sequence-constructor
</xsl:processing-instruction>

<xsl:text>                                      [inst:11.4.2]
  disable-output-escaping = "yes"|"no" >
    character data
</xsl:text>

<xsl:value-of                                   [inst:11.4.3]
  select = expression
  separator = {string}
  disable-output-escaping = "yes"|"no" >
    sequence-constructor
</xsl:value-of>

<xsl:sequence                                   [inst:11.10]
  select = expression
  as = sequence-type >
    sequence-constructor
</xsl:sequence>
```

Variables and parameters

```
<xsl:variable                                   [9.1]
  name = QName
  select = expression
  as = sequence-type >
    sequence-constructor
</xsl:variable>

<xsl:param                                      global/actual parameters [9.2]
  name = QName
  select = expression
  as = sequence-type
  required = "yes"|"no"
  tunnel = "yes"|"no" >
    sequence-constructor
</xsl:param>

<xsl:with-param                                 formal parameters [10.1.1]
  name = QName
  select = expression
  as = sequence-type
  tunnel = "yes"|"no" >
    sequence-constructor
</xsl:with-param>
```

Source copy

```
<xsl:copy                                       Shallow copy [inst:11.9.1]
  copy-namespaces = "yes"|"no"
  inherit-namespaces = "yes"|"no"
  use-attribute-sets = QNames
  type = QName
  validation = "strict"|"lax"|"preserve"|"strip" >
    sequence-constructor
</xsl:copy>

<xsl:copy-of                                    Deep copy [inst:11.9.2]
  select = expression
  copy-namespaces = "yes"|"no"
  type = QName
  validation = "strict"|"lax"|"preserve"|"strip" />
```

Condition and iteration

```
<xsl:if                                         simple if-then [inst:8.1]
  test = expression >
    sequence-constructor
</xsl:if>

<xsl:choose>                                    case-like [inst:8.2]
    xsl:when+, xsl:otherwise?
</xsl:choose>

<xsl:when                                       [8.2]
  test = expression >
    sequence-constructor
</xsl:when>

<xsl:otherwise>                                 [8.2]
    sequence-constructor
</xsl:otherwise>

<xsl:for-each                                   Repetition [inst:7]
  select = expression >
    xsl:sort*, sequence-constructor*
</xsl:for-each>
```

XLST 2.0 - Quick Reference Sheet

```
<xsl:for-each-group                                    [inst:14.3]
  select = expression
  group-by = expression
  group-adjacent = expression
  group-starting-with = pattern
  group-ending-with = pattern
  collation = anyURI >
    xsl:sort*, sequence-constructor*
</xsl:for-each-group>

<xsl:analyze-string                    Regular expr [inst:15.1]
  select = expression
  regex = {string}
  flags = {string} >
    xsl:matching-substring?,
    xsl:non-matching-substring?, xsl:fallback*
</xsl:analyze-string>

<xsl:matching-substring>                               [15.1]
    sequence-constructor
</xsl:matching-substring>

<xsl:non-matching-substring>                           [15.1]
    sequence-constructor
</xsl:non-matching-substring>
```

Sorting, searching and numbering

```
<xsl:sort                          defining sorting order [13.1]
  select = expression
  lang = {string}
  data-type = {string}
  order = {string}
  case-order = {string}
  collation = {string}
  stable = "yes"|"no" >
    sequence-constructor
</xsl:sort>

<xsl:perform-sort                                      [inst:13.2]
  select = expression >
    xsl:sort+, sequence-constructor*
</xsl:perform-sort>

<xsl:key                                               [decl:16.3.1]
  name = QName
  match = pattern
  use = expression
  collation = anyURI >
    sequence-constructor
</xsl:key>

<xsl:number                                            [inst:12]
  value = expression
  select = expression
  level = "single"|"multiple"|"any"
  count = pattern
  from = pattern
  format = {string}
  lang = {string}
  letter-value = {string}
  ordinal = {string}
  grouping-separator = {string}
  grouping-size = {string} />
```

```
<xsl:decimal-format                                    [decl:16.4.1]
  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 />
```

Output control

```
<xsl:output                                            [decl:20]
  name = QName
  method = "xml"|"xhml"|"html"|"text"|
    qname-but-not-cname
  byte-order-mark = "yes"|"no"
  cdata-section-elements = QNames
  doctype-public = string
  doctype-system = string
  encoding = string
  escape-uri-attributes = "yes"|"no"
  include-content-type = "yes"|"no"
  indent = "yes"|"no"
  media-type = string
  normalization-form = NMTOKEN
  omit-xml-declaration = "yes"|"no"
  standalone = "yes"|"no"|"omit"
  undeclare-prefixes = "yes"|"no"
  use-character-maps = QNames
  version = NMTOKEN />

<xsl:result-document                                   [inst:19.1]
  format = {string}
  href = {string}
  type = QName
  validation = "strict"|"lax"|"preserve"|"strip"
  method = {string}
  byte-order-mark = {string}
  cdata-section-elements = {string}
  doctype-public = {string}
  doctype-system = {string}
  encoding = {string}
  escape-uri-attributes = {string}
  include-content-type = {string}
  indent = {string}
  media-type = {string}
  normalization-form = {string}
  omit-xml-declaration = {string}
  standalone = {string}
  undeclare-prefixes = {string}
  use-character-maps = QNames
  output-version = {string} >
    sequence-constructor
</xsl:result-document>

<xsl:character-map                                     [decl:20.1]
  name = QName
  use-character-maps = QNames >
    xsl:output-character*
</xsl:character-map>

<xsl:output-character                                  [20.1]
  character = char
  string = string />
```

Miscellaneous

```
<xsl:message                                   tracing [inst:17]
  select = expression
  terminate = {string} >
    sequence-constructor
</xsl:message>

<xsl:attribute-set                                     [decl:10.2]
  name = QName
  use-attribute-sets = QNames >
    (xsl:attribute)*
</xsl:attribute-set>

<xsl:strip-space                                       [decl:4.4]
  elements =
    (qname|"*"|"prefix:*"|"*:localname")+|"#all" />

<xsl:preserve-space                                    [decl:4.4]
  elements =
    (qname|"*"|"prefix:*"|"*:localname")+|"#all" />

<xsl:fallback>                                         [inst:18.2.3]
    sequence-constructor
</xsl:fallback>

<xsl:namespace-alias                                   [decl:11.1.4]
  stylesheet-prefix = prefix-or-default
  result-prefix = prefix-or-default />
```

Attributes for any element

```
default-collation = uri-list
exclude-result-prefixes = (NCName)*|"#all"
extension-element-prefixes = prefix-list
use-when = expression
xpath-default-namespace = anyURI
version = decimal
```

All declarations

xsl:attribute-set, xsl:character-map, xsl:decimal-format, xsl:function, xsl:import-schema, xsl:include, xsl:key, xsl:namespace-alias, xsl:output, xsl:preserve-space, xsl:strip-space, xsl:template, xsl:variable

All instructions

xsl:analyze-string, xsl:apply-imports, xsl:apply-templates, xsl:attribute, xsl:call-template, xsl:choose, xsl:comment, xsl:copy, xsl:copy-of, xsl:document, xsl:element, xsl:fallback, xsl:for-each, xsl:for-each-group, xsl:if, xsl:message, xsl:namespace, xsl:next-match, xsl:number, xsl:perform-sort, xsl:processing-instruction, xsl:result-document, xsl:sequence, xsl:text, xsl:value-of, xsl:variable

Conventions used in this sheet

bold	required attribute	*italic*	element content
{...}	attribute value template	,	sequence
?	optional	|	alternative
*	repetition 0 or more	+	repetition 1 or more