

# XSLT – QuickReference

## Basic elements

### Root

```
<xsl:stylesheet
  id = id
  extension-element-prefixes = tokens
  exclude-result-prefixes = tokens
  version = number
  xpath-default-namespace = uri
  default-validation = "strict" | "lax" | "preserve" | "strip">
  <!-- Inhalt: (xsl:import*, other-declarations) -->
</xsl:stylesheet>
```

### Output

```
<xsl:output
  name = qname
  method = "xml" | "html" | "xhtml" | "text" | qname-but-not-ncname
  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
  normalize-unicode = "yes" | "no"
  omit-xml-declaration = "yes" | "no"
  standalone = "yes" | "no"
  undeclare-namespaces = "yes" | "no"
  use-character-maps = qnames
  version = nmtoken />
```

### Whitespace

Deletes whitespace from the elements in the source document which are named in the token list.

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

Preserves whitespace in the listed elements.

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

### Legend

[ element1] element2 ]  
Either element1 or element2 from the element group  
[\*]  
Repeated use of group possible  
{ }  
Optional declaration

## Templates

### Definition

Defines a template which can be addressed by its name in the name attribute or which is automatically selected by the XPath pattern in the match attribute. An additional name in the mode attribute can be used to create templates with the same name but different modes i.e. different processing under the same name or for the same pattern.

```
<xsl:template
  match = pattern
  name = qname
  priority = number
  mode = tokens
  as = sequence-type>
  <!-- Inhalt: (xsl:param*, sequence-constructor) -->
</xsl:template>
```

### Applying templates

Applies suitable templates which are selected by the XPath patterns in the select attribute and – if existing – the value in the mode attribute.

```
<xsl:apply-templates
  select = node-sequence-Ausdruck
  mode = token>
  <!-- Inhalt: (xsl:sort | xsl:with-param)* -->
</xsl:apply-templates>
```

Calls a template by using the qualified name and applies it to the selected node list.

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

Use of the imported templates instead of the already selected template.

```
<xsl:apply-imports>
  <!-- Inhalt: xsl:with-param* -->
</xsl:apply-imports>
```

Additional use of the imported templates following the priorities in the import tree.

```
<xsl:next-match>
  <!-- Inhalt: (xsl:with-param | xsl:fallback)* -->
</xsl:next-match>
```

### Output

Outputs the PCDATA text in the result document with – if necessary – parsing of internal entities.

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

Outputs the result of the XPath pattern (e.g. the content of a simple text node or the result of a calculation).

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

## XML Output

### Elements

Creates an XML element with the name specified in the name attribute.

```
<xsl:element
  name = { qname }
  namespace = { uri-reference }
  use-attribute-sets = qnames
  validation = "strict" | "lax" | "preserve" | "strip"
  type = qname>
  <!-- Inhalt: sequence-constructor -->
</xsl:element>
```

### Attributes

Creates an XML attribute with the name specified in the name attribute.

```
<xsl:attribute
  name = { qname }
  namespace = { uri-reference }
  validation = "strict" | "lax" | "preserve" | "strip"
  type = qname
  disable-output-escaping = "yes" | "no">
  <!-- Inhalt: sequence-constructor -->
</xsl:attribute>
```

Copies attributes which are stored in global attribute sets into the element using a whitespace separated list of attribute set names.

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

### Comments

Creates an XML comment containing the text node as output.

```
<xsl:comment>
  <!-- Inhalt: sequence-constructor -->
</xsl:comment>
```

## Control Statements

### Conditional processing

Encloses a conditional statement consisting of one or more when statements (cases) and additionally an optional default case which is defined in the otherwise element.

```
<xsl:choose>
  <!-- Content: (xsl:when+, xsl:otherwise?) -->
  <xsl:when
    test = expression>
    <!-- Content: sequence-constructor -->
  </xsl:when>
  <xsl:otherwise>
    <!-- Content: sequence-constructor -->
  </xsl:otherwise>
</xsl:choose>
```

Executes statements if the condition in the test attribute evaluates to true.

```
<xsl:if
  test = expression>
  <!-- Content: sequence-constructor -->
</xsl:if>
```

### Loops

Creates an iteration for all nodes which are selected by the XPath expressions in the select attribute and executes the inner statements.

```
<xsl:for-each
  select = sequence-expression>
  <!-- Content: (xsl:sort*, sequence-constructor) -->
</xsl:for-each>
```



## Dynamic Values

### Variables

Creates a variable with the name being defined in the name attribute. The value is either defined in the select attribute or by the executed statements in the sequence constructor. The scope of the variable is either the element in which it was declared or the stylesheet that is declared as a child element of xsl:stylesheet.

```
<xsl:variable
  name = qname
  select = expression
  as = sequence-type>
<!-- Inhalt: sequence-constructor -->
</xsl:variable>
```

### Parameters

Creates a parameter for a template (if declared within xsl:template) or the stylesheet (if directly declared within xsl:stylesheet). A default value can be stored in the select attribute or by the execution of the sequence constructor.

```
<xsl:param
  name = qname
  select = Ausdruck
  as = sequence-type
  required = "yes" | "no">
<!-- Inhalt: sequence-constructor -->
</xsl:param>
```

Sets the value of a parameter when using xsl:apply templates or xsl:call template. The value is either stored in the select attribute or generated by the execution of the statements in the sequence constructor.

```
<xsl:with-param
  name = qname
  select = Ausdruck>
<!-- Inhalt: sequence-constructor -->
</xsl:with-param>
```

## Copying

Copies the name of the context node into the result document.

```
<xsl:copy
  copy-namespaces = "yes" | "no"
  use-attribute-sets = qnames
  validation = "strict" | "lax" | "preserve" | "strip"
  type = qname>
<!-- Content: sequence-constructor -->
</xsl:copy>
```

Copies the selected tree into the result document.

```
<xsl:copy-of
  select = Ausdruck
  copy-namespaces = "yes" | "no"
  validation = "strict" | "lax" | "preserve" | "strip"
  type = qname />
```

## Grouping

Groups the selected node set by the expression in the group-by attribute. Advanced grouping via start and end expressions is possible by using the attributes group-starting-with and group-ending-with. The function current-grouping-key() gets the present value of the grouping key which is evaluated by group-by. The function current-group() points to the context group and is used within the select attributes of xsl:apply templates or xsl:for-each.

```
<xsl:for-each-group
  select = expression
  group-by = expression
  group-adjacent = expression
  group-starting-with = pattern
  group-ending-with = pattern
  collation = { uri }>
<!-- Content: (xsl:sort*, sequence-constructor) -->
</xsl:for-each-group>
```

## Sorting

Sorts the selected nodes within xsl:apply templates and xsl:for-each.

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

## Regular Expressions

Tests the regular expression in the regex attribute within the scope of the selected nodes.

```
<xsl:analyze-string
  select = Ausdruck
  regex = { string }
  flags = { string }>
<!-- Content: (xsl:matching-substring?,
xsl:non-matching-substring?, xsl:fallback*) -->
```

Contains the statements for the matching nodes.

```
<xsl:matching-substring>
<!-- Content: sequence-constructor -->
</xsl:matching-substring>
```

Contains the statements for the non-matching nodes.

```
<xsl:non-matching-substring>
<!-- Content: sequence-constructor -->
</xsl:non-matching-substring>
```

## Modular XSLT files

### Including files

External templates can be included before and after internal xsl:template elements. Therefore, external templates can override internal ones or be overridden by the internal templates depending on the position of the xsl:include element.

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

### Importing files

The import statement has to be located before all internal xsl:template elements. Therefore, imported templates can only be overridden and they cannot override internal ones.

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

### XML Schema

Imports an XML Schema document so that names of data types and global elements defined within the XML Schema can be referenced from the XSLT document.

```
<xsl:import-schema
  namespace = { uri-reference }
  schema-location = { uri-reference } />
```

## Functions

Declares a user-defined function which can be used in XPath expression (select or test attributes).

```
<xsl:function
  name = qname
  as = sequence-type
  override = „yes“ | „no“>
<!-- Content: (xsl:param*, sequence-constructor) -->
</xsl:function>
```

## Keys

Generates a unique key which can be referenced from the key() function in XPath expressions.

```
<xsl:key
  name = qname
  match = pattern
  use = Ausdruck
  as = qname
  collation = { uri } />
```



## Documentation

XSLT 1.0  
XSL Transformations (XSLT) Version 1.0, W3C Recommendation.  
Siehe <http://www.w3.org/TR/xslt>  
XSLT 2.0  
XSL Transformations (XSLT) Version 2.0, W3C Recommendation.  
Siehe <http://www.w3.org/TR/xslt2>  
XPath 1.0  
XML Path Language (XPath) Version 1.0, W3C Recommendation.  
Siehe <http://www.w3.org/TR/xpath>

XPath 2.0  
XML Path Language (XPath) 2.0, W3C Recommendation.  
Siehe <http://www.w3.org/TR/xpath20/>  
XPath-/XQuery-Funktionen  
XQuery 1.0 and XPath 2.0 Functions and Operators, W3C Recommendation.  
Siehe <http://www.w3.org/TR/xpath-functions/>