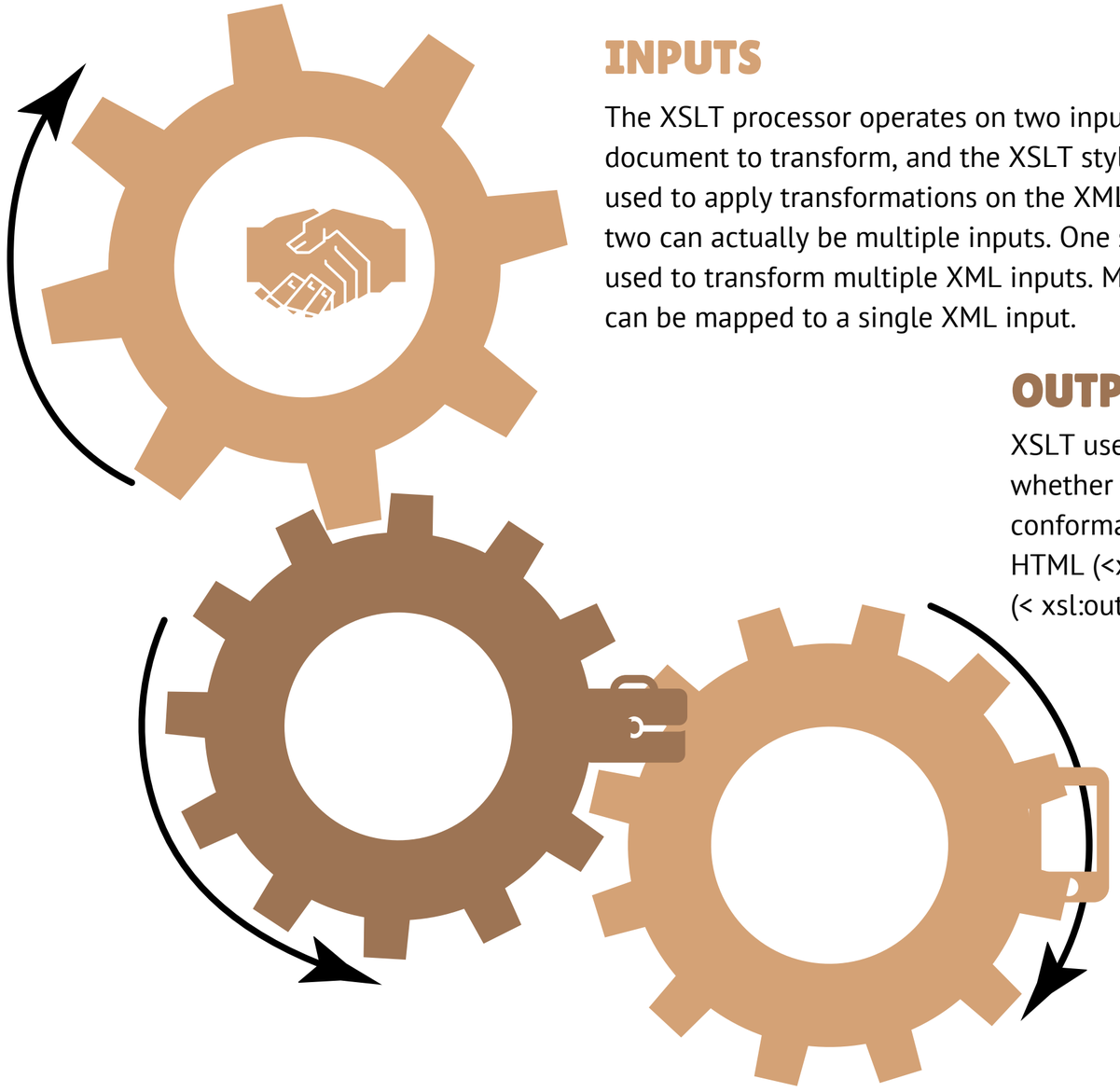
XSLT PROCESSING MODELS

TREES



INPUTS

The XSLT processor operates on two inputs: the XML document to transform, and the XSLT stylesheet that is used to apply transformations on the XML. Each of these two can actually be multiple inputs. One stylesheet can be used to transform multiple XML inputs. Multiple stylesheets can be mapped to a single XML input.

OUTPUT

XSLT uses the <xsl:output> element to determine whether the output produced by the transformation is conformant XML (<xsl:output method="xml"/>), valid HTML (<xsl:output method="html"/>), or unverified text (< xsl:output method="text"/>)

TEMPORARY

XSLT 2.0 eliminates result tree fragments and replaces them with a more powerful feature: temporary trees. Once you create a temporary tree in an xsl:variable, xsl:param, or xsl:with-param element, you can do anything with it that you can do with a source tree.

XSLT STYLESHEET MODULES

STYLESHEET MODULE

A stylesheet module is either a ***standard stylesheet module*** or a ***simplified stylesheet module***:

A standard stylesheet module: this is a tree, or part of a tree, consisting of an xsl:stylesheet or xsl:transform element together with its descendant nodes and associated attributes and namespaces.]

A simplified stylesheet module: This is a tree, or part of a tree, consisting of a literal result element together with its descendant nodes and associated attributes and namespaces. This element is not itself in the XSLT namespace, but it must have an xsl:version attribute, which implies that it must have a namespace node that declares a binding for the XSLT namespace.

STYLESHEET MODULE

Both forms of stylesheet module (standard and simplified) can exist either as an entire XML document, or embedded as part of another XML document, typically but not necessarily a source document that is to be processed using the stylesheet.

A standalone stylesheet module: This is a stylesheet module that comprises the whole of an XML document.
An embedded stylesheet module: This is a stylesheet module that is embedded within another XML document, typically the source document that is being transformed.

KINDS OF STYLESHEET MODULE

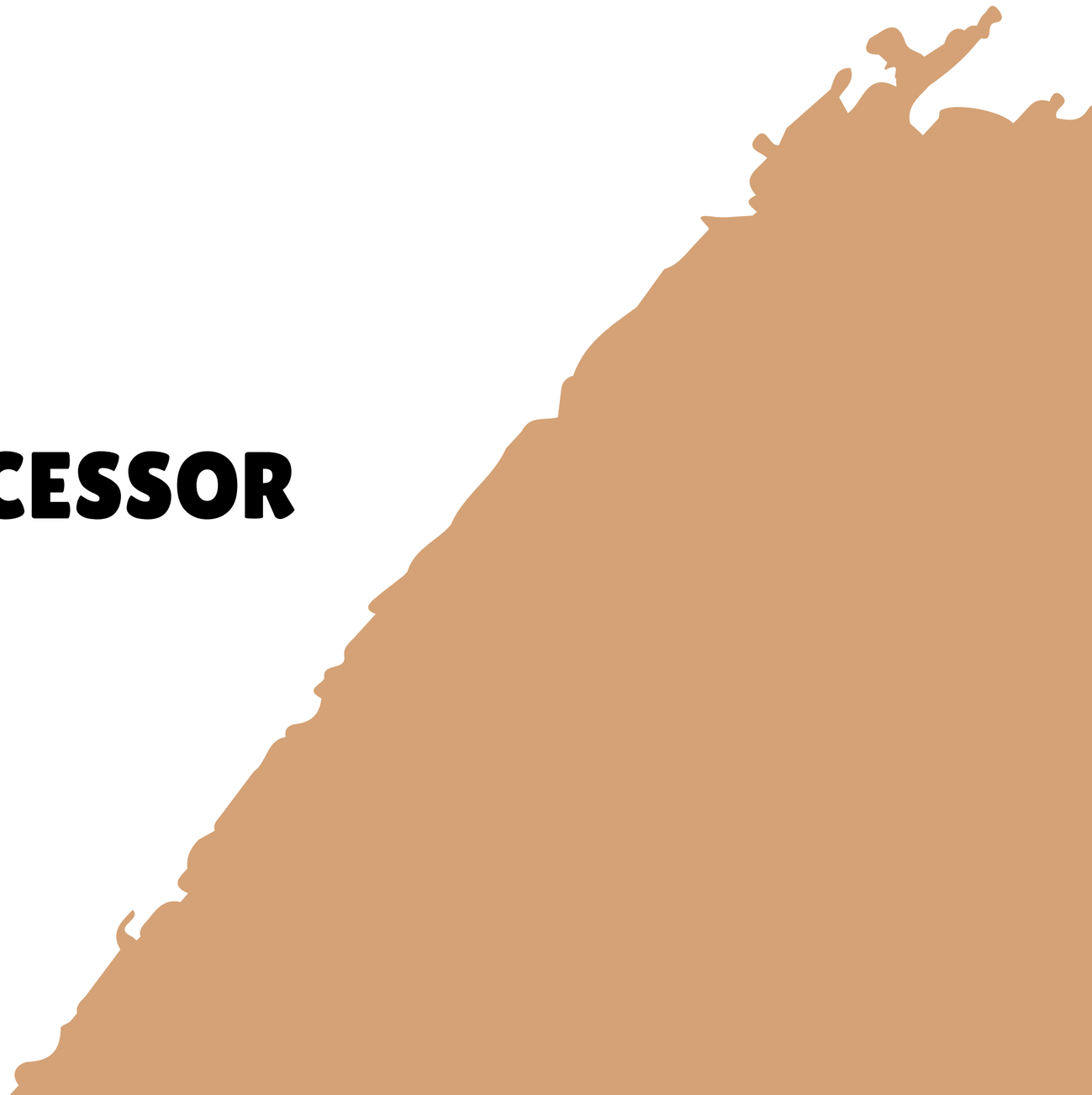
- standalone standard stylesheet modules
- standalone simplified stylesheet modules
- embedded simplified stylesheet modules
- embedded standard stylesheet modules

XSLT EXTERNAL CONTEXT

KINDS OF STYLESHEET MODULE

The results of some expressions and instructions in a stylesheet may depend on information provided contextually. This context information is divided into two categories: the static context, which is known during static analysis of the stylesheet, and the dynamic context, which is not known until the stylesheet is evaluated. Although information in the static context is known at analysis time, it is sometimes used during stylesheet evaluation.

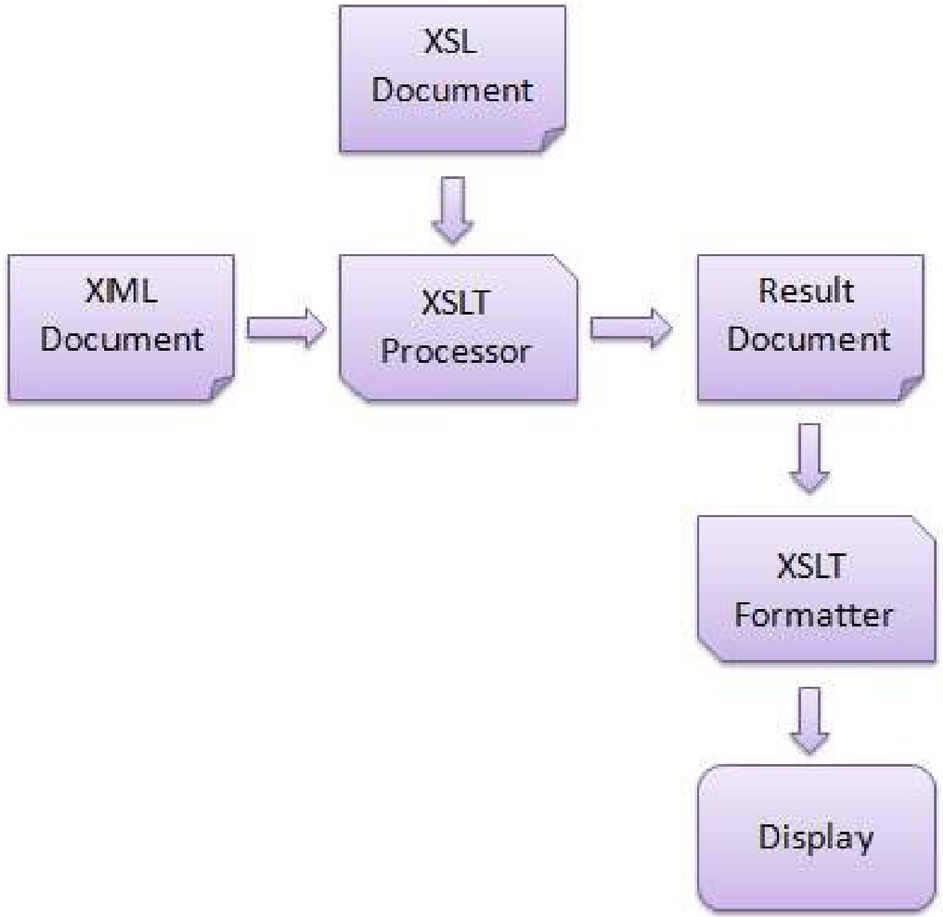
THE XSLT PROCESSOR



THE XSLT PROCESSOR

XSLT Processor takes the XSLT stylesheet and applies the transformation rules on the target XML document and then it generates a formatted document in the form of XML, HTML, or text format. This formatted document is then utilized by XSLT formatter to generate the actual output which is to be displayed to the end-user.

THE XSLT PROCESSOR



WHAT KIND OF LANGUAGE IS XSLT

Extensible Stylesheet Language Transformations (XSLT) is an XML-based language used, in conjunction with specialized processing software, for the transformation of XML documents.

BASIC XSLT CONCEPTS

XSLT CONCEPT

- Template Rules
- Built-in templates
- Sequence constructors
- Variables and expressions
- Parameters
- Context

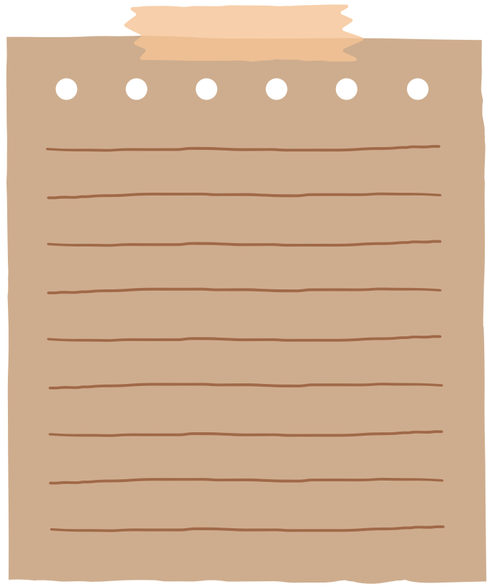
TEMPLATE RULES

An XSL style sheet consists of one or more set of rules that are called templates. A template contains rules to apply when a specified node is matched.

Declaration

Following is the syntax declaration of **<xsl:template>** element.

```
<xsl:template
  name = Qname
  match = Pattern
  priority = number
  mode = QName >
</xsl:template>
```



ATTRIBUTES

1	**Name** Name of the element on which template is to be applied.
2	**match** Pattern which signifies the element(s) on which template is to be applied.
3	**priority** Priority number of a template. Matching template with low priority is not considered in from in front of high priority template.
4	**mode** Allows element to be processed multiple times to produce a different result each time.

DEMO

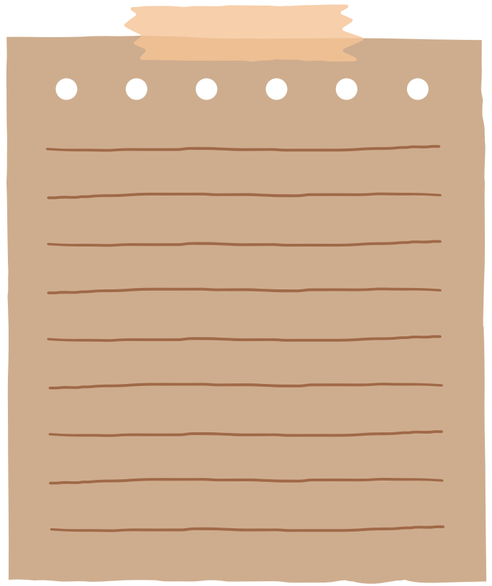
This template rule has a pattern that identifies <student> elements and produces an output in a tabular format.

BUILT-IN TEMPLATE RULES

Built-in template rule for element and document nodes

This template processes the document node and any of its children. This processing ensures that recursive processing will continue, even if no template is declared for a given element:

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

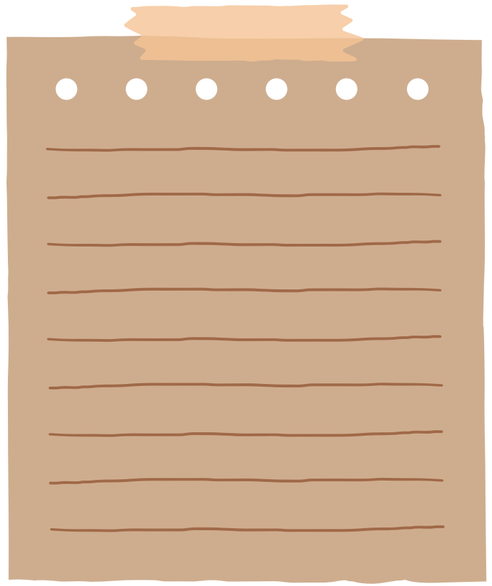


BUILT-IN TEMPLATE RULES

Built-in template rule for modes

This template ensures that element and document nodes are processed, regardless of any mode that might be in effect.

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

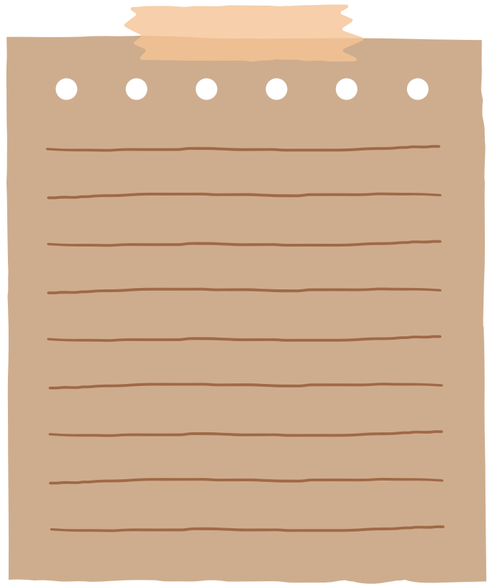


BUILT-IN TEMPLATE RULES

Built-in template rule for text and attribute nodes

This template copies the text of all text and attribute nodes to the output tree. Be aware that you have to actually select the text and attribute nodes for this rule to be invoked:

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



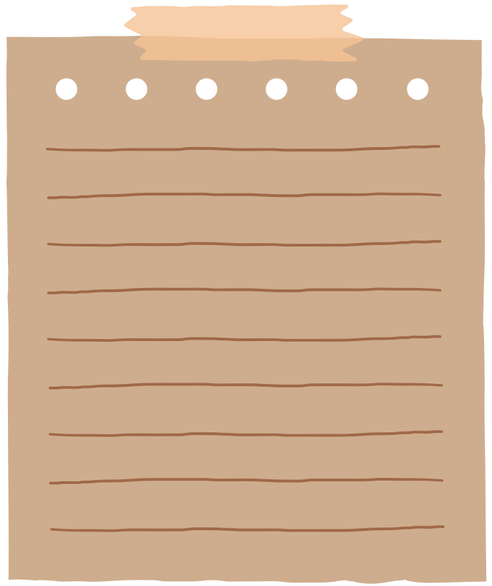
BUILT-IN TEMPLATE RULES

Built-in template rule for comment and processing instruction nodes
This template does nothing:

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

Built-in template rule for namespace nodes
This template also does nothing:

```
<xsl:template match="namespace()"/>
```



SEQUENCE CONSTRUCTOR

xsl:sequence

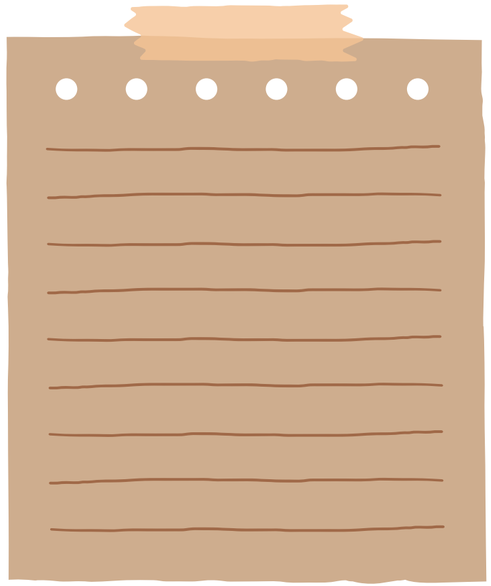
Used to construct arbitrary sequences. It may select any sequence of nodes and/or atomic values, and essentially adds these to the result sequence. Available in XSLT 2.0 and later versions. Available in all Saxon editions.

Attributes

select?

expression

Specifies the input. Mandatory attribute in XSLT 2.0, but in XSLT 3.0 (and implemented since Saxon 9.5) the input may be specified either by a select attribute, or by the enclosed sequence constructor.



SEQUENCE CONSTRUCTOR

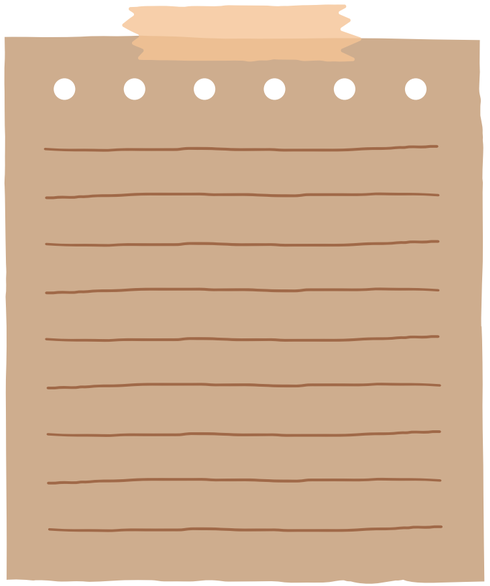
Example 1⚓
Returning a result from a function:

```
<xsl:function name="f:increment" as="xs:integer">
<xsl:param name="in" as="xs:integer"/>
<xsl:sequence select="$in + 1"/>
</xsl:function>
```

Example 2⚓
Copying atomic values into a tree:

```
<e>
<xsl:sequence select="1 to 5"/>
<br/>
<xsl:sequence select="6 to 10"/>
</e>
```

This produces the output <e>1 2 3 4 5
6 7 8 9 10</e>.



VARIABLES AND EXPRESSIONS

Used to declare a variable and give it a value. If it appears at the top level (immediately within xsl:stylesheet) it declares a global variable, otherwise it declares a local variable that is visible only within the stylesheet element containing the xsl:variable declaration. The value of a variable can be referenced within an expression using the syntax $name.

Type	Description
name	eqname Defines the name of the variable.
select?	expression The value of the variable may be defined either by an expression within the optional select attribute, or by the contents of the xsl:variable element. In the latter case the result is a temporary tree. A temporary tree can be used like a source document, for example it can be accessed using path expressions and processed using template rules.
as?	sequence-type Defines the required type of the variable. The supplied value of the variable will be converted to this type if required.
static?	boolean The value yes may be specified only for global variables (not local variables), and indicates that the variable is a static variable (that is, its value is known during static analysis of the stylesheet). The default is no.

Type	Description
visibility?	"public" | "private" | "final" | "abstract" New in XSLT 3.0. Allowed only for global variables (not local variables). Determines the potential visibility of the component corresponding to this variable; the default is private.
saxon:as?	sequence-type Allows additional type information to be supplied using Saxon extension syntax.
saxon:assignable?	boolean May be set on global variables. Setting the value to yes ensures that the variable is actually evaluated, which is useful if the select expression calls extension functions with side-effects; without this, a variable that is never referenced may never be evaluated.

Notes on the Saxon implementation

In standard XSLT, variables once declared cannot be updated. Saxon however provides a saxon:assign extension element to circumvent this restriction. The extension attribute saxon:assignable must be set to yes on the xsl:variable in order to use this feature.

PARAMETERS

Used to define a formal parameter to a template, the stylesheet, a function, or an iteration.

Type	Description
name	eqname The name of the parameter.
select?	expression The default value of the parameter may be defined either by a select attribute, or by the contents of the xsl:param element, in the same way as for xsl:variable. The default value is ignored if an actual parameter is supplied with the same name.
as?	sequence-type Defines the required type of the parameter. The supplied value of the parameter will be converted to this type if required. If the parameter is omitted, the default value must conform to the type. Note that if no default is specified, the default is a zero-length string, which may conflict with the required type.
required?	boolean Not allowed for function parameters, which are always required. If the parameter is required, no default value may be specified. Failure to supply a value for a required parameter gives a run-time error (the specification says that in the case of xsl:call-template, it should be a static error).

Type	Description
tunnel?	boolean The value yes may be specified only for template parameters, and indicates that the parameter is a tunnel parameter. Tunnel parameters are automatically passed on by the called template to any further templates that it calls, and so on recursively. The default is no.
static?	boolean The value yes may be specified only for stylesheet parameters, and indicates that the parameter is a static parameter (that is, its value is known during static analysis of the stylesheet). The default is no.
saxon:as?	sequence-type Allows additional type information to be supplied using Saxon extension syntax.
saxon:assignable?	boolean May be set on global variables. Setting the value to yes ensures that the variable is actually evaluated, which is useful if the select expression calls extension functions with side-effects; without this, a variable that is never referenced may never be evaluated.

CONTEXT

Used to declare the initial context item for a template: whether the template requires a context item, and if so, what its expected type is

Type	Description
as?	item-type The required type of the context item; the default is item().
use?	"required" | "optional" | "absent" Specifies whether a template requires a context item; the default is optional.

Notes on the Saxon implementation

Implemented since Saxon 9.7. The initial implementation is functionally complete, but the context item information is not used for optimization or for static type checking. The information is used for assessing streamability.

THANKS FOR WATCHING

Thynk Unlimited

Presentations are communication tools that can be used as demonstrations.