Rexsel: A Compact XSLT

Hugh Field-Richards

June 2024

Contents

Preface

Rexsel is a simplified (compact) version of XSLT. It does not use XML to write the code, rather a simplified, easy to read language. It has been successfully used on the Paloose[1] site, and the Rexsel[2] site. In both cases generating all the necessary XSLT transform files. This is done when the site is built and uploaded to the server, rather than being performed on each browser request.

It was conceived after many years of writing XSLT templates for translating XML into various forms: XML, XHTML, Swift code, and even LaTeX. While the underlying structure is virtually identical to XSLT, it is hoped that the result will help users produce more concise, readable stylesheets independent of XML.

```
stylesheet {
    version "1.0"

     proc helloWorld {
        element "div" {
            attribute "class"  "simpleBox"
            text "Hello World!"
        }
    }
}
```

Rexsel was developed on a Mac using Xcode and Swift as the underlying language, however the actual compiler kernel is "OS neutral". The compiler (*RexselKernel*[3]) is written as a Swift Package, and there is a command line application (*CRexsel*[4]), also a Swift Package, that uses this *RexselKernel* package.

Both can be built on any system that supports Swift (Windows may be able to support Swift in the near future and Rexsel will be tested on it). Currently it has been tested on a MacOS system and a Parallels based Ubuntu Linux, both running on MacOS 14.4.1 Sonora on a M2 Mac Mini.

The *CRexsel* application also provides an "uncompiler" procedure that takes existing XSLT files and translates them to a basic *Rexsel* format (ignoring XML comments). This should make migrating XSLT stylesheets into *Rexsel* much easier.

There is a MacOS only application, using the same *RexselKernel* package, that provides an editor and real-time compile function.

[1] `https://www.paloose.org`
[2] `https://www.rexsel.org`
[3] Currently at `https://hsfr@bitbucket.org/hsfr/rexselkernel.git`
[4] Currently at `https://hsfr@bitbucket.org/hsfr/crexsel.git`

Chapter 1

Introduction

Rexsel provides a clean, easily understandable means of writing XSLT scripts without having the extra "baggage" of XML syntax. It allows the actual mechanism of the script to be grasped more easily. It does nothing other than provide a different means of producing XSLT, in particular it does not replace supporting languages used by XSLT such as XQuery.

To illustrate the language, consider a simple XSLT stylesheet that defines a (recursive) procedure that outputs, or returns, a string after the last occurrence of a specified delimiter within a given string.

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <template name="substring-after-last">
        <param name="string"/>
        <param name="delimiter"/>

        <choose>
            <when test="contains($string, $delimiter)">
                <call-template name="substring-after-last">
                    <with-param name="string" select="substring-after($string, $delimiter)"/>
                    <with-param name="delimiter" select="$delimiter"/>
                </call-template>
            </when>
            <otherwise>
                <value-of select="$string"/>
            </otherwise>
        </choose>
    </template>
</stylesheet>
```

Even with a good understanding of XML and XSLT, it can be difficult to see the operation of this function. One particular aspect is the use of the keyword '`template`' which is used to define both functions and matches, albeit with different attributes: '`name`' and '`match`' respectively.

Rexsel attempts to make this whole process more transparent, and thus the XSLT proc above can be written as:

```
stylesheet {
   version "1.0"

    proc substring-after-last {
      parameter string
      parameter delimiter
      choose {
         when "contains($string, $delimiter)" {
            call substring-after-last {
               with string "substring-after($string, $delimiter)"
               with delimiter "$delimiter"
            }
         }
         otherwise {
            value "$string"
         }
      }
   }
}
```

Chapter 2

XSLT/Rexsel Comparisons

Most of the *Rexsel* syntax uses familiar keywords that have a direct analogue within XSLT. However, there are certain restrictions and enhancements which, while not onerous, must be borne in mind when writing *Rexsel* stylesheets.

1. Templates in XSLT can be either a ***named*** or ***pattern match*** template. The former is effectively a proc that can be called from elsewhere in the stylesheet, while *match* is a means of matching the input against a pattern. *Rexsel* explicitly distinguishes between these two types by using different keywords: '`function`' (**??**) and '`match`' (6.18).

   Having a distinction between '`function`' and '`match`' ensures that there is never confusion and fulfils the requirement that the '`name`' and '`match`' attributes should never occur together in a `<xsl:template>` tag.

   As an example, the '`match`' below matches the input against the XPath "`p`" within the scope (mode) of "inline-text".

```
match using "p" scope "inline-text" {
    variable extra "'normalPara'"

    element "para" {
        apply-templates scope "inline-text"
    }
}
```

   This would translate to

```
<template  match="p" mode="inline-text">
    <variable  name="extra" select="'normalPara'"/>

    <element  name="para">
        <apply-templates  mode="inline-text"/>
    </element>
</template>
```

   Procedures are simpler:

```
 proc common.substring-after-last {
    parameter string
    parameter delimiter

    ...
}
```

would translate to

```
<template name="common.substring-after-last">
   <param  name="string"/>
   <param  name="delimiter"/>

   ...
</template>
```

This proc could be called as

```
call common.substring-after-last {
   with string "Some text"
   with delimiter ","
}
```

As an aside, note the use of names and string above. In general items that will appear in the output after XSLT processing is held within quote marks, while items such as proc names that are used by the XSLT processor do not.

2. Within '`match`' templates there is concept of ***mode*** which defines a ***context*** or ***scope*** for that match (see example above). *Rexsel* uses the '`scope`' keyword which, I believe, more accurately describes the concept.

3. When describing text strings it is important to "escape" control characters such as '\'.

Required	Entered
'\'	'\\'
'<'	'<'
'>'	'>'
'"'	'\"'

4. When not a simple piece of text, such as in '`attribute`', text must be enclosed using the '`text`' keyword. For example

```
<xsl:template name="begin-documentation-section">
     \nwbegindocs{<xsl:value-of select="position() - 1"/>}
</xsl:template>
```

would have to be written as

```
proc begin-documentation-section {
   text "\nwbegindocs{"
   value "position() - 1 "
   text "}"
}
```

or as

```
proc begin-documentation-section {
   value "concat( '\nwbegindocs{', position() - 1, '}' )"
}
```

5. *Rexsel* does not allow "naked" non-XSLT elements. For example if the XSLT contained

```
<xsl:template match="example">
```

```
  <div style="border: solid red">
     <xsl:apply-imports/>
  </div>
</xsl:template>
```

this would have to be written in *Rexsel* as

```
match using "example" {
   element "div" {
      attribute "style" "border: solid red"
      apply-imports
   }
}
```

While these restrictions may feel restrictive, I believe that they encourage a more structured, and thus more resilient, code to be produced.

Compiling *Rexsel* Stylesheets

The *Rexsel* compiler can be used either as a command line program (Chapter 10), or as an Apple application[1]. The compiler itself is relatively mature, offering checks on duplicate and undefined variables, and a listing of symbols used within the program (see Chapter 5).

The MacOS App provides an editing window with three panels (Figure 2.1): an *Rexsel* source panel for inputing the code, an XSLT panel that displays the code produced by the compiler (in real time), and an error/symbols panel that lists errors and the symbol table. However the editor panel is currently relative crude and needs enhancing to be fully operational. Currently this application is used by me to support the development of the compiler. The command line application is the ideal way to run *Rexsel*.

[1] This is currently not supported on the App Store and it may never be. It is in very early stages of development and is work in progress.

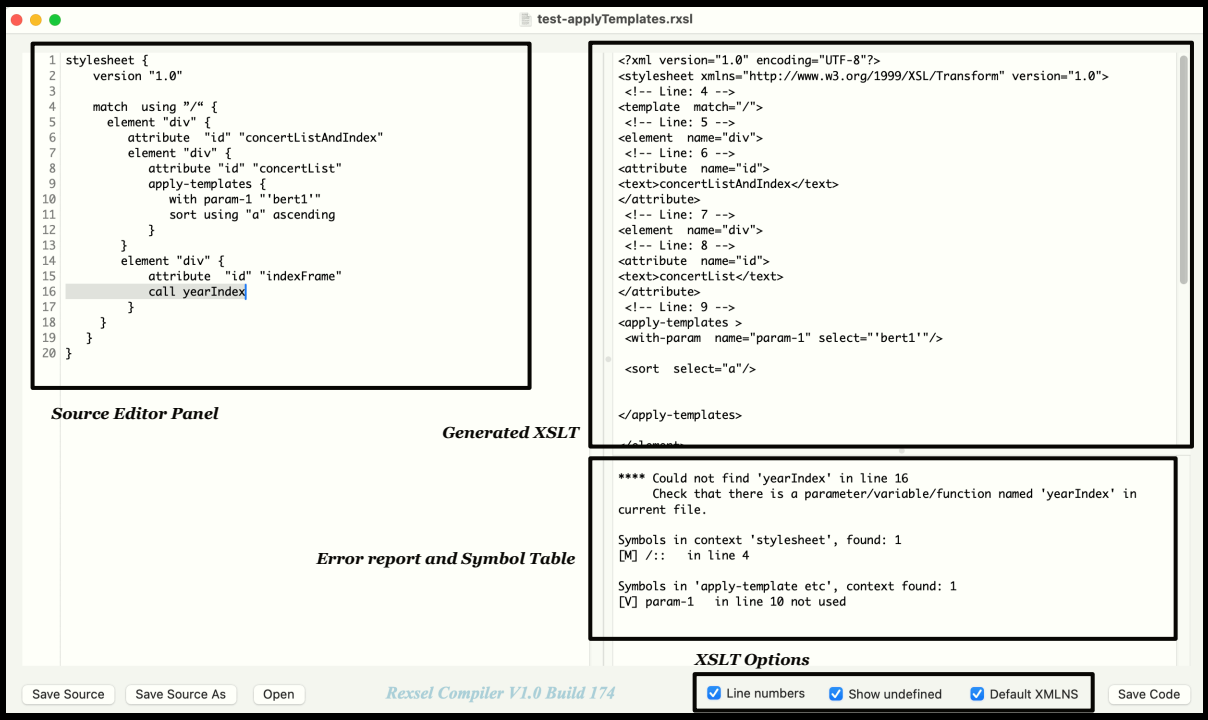

Figure 2.1: MacOS Rexsel Application Editing Window

Chapter 3

Further Work

Although the current state of *Rexsel* language and compiler is reasonably mature there are several areas which could benefit from further work.

- Analysing XPath expressions. This is a fairly advanced process and may never be implemented.
- Checking on '`scope`' (mode) definitions to ensure that there is a corresponding definition of the name in a '`match`' statement.
- Checking if parameters in a procedure exist when the call is made.
- Improving the editor for the MacOS application which is currently very crude. The text editor panel needs: undo/redo, syntax aware colouring, etc.

Chapter 4

Running the `crexsel` Command

The command line using `crexsel` is the recommended and publicly available way to run Rexsel. It can also, therefore, be used in conjunction with a build system, such as Apache Ant.

Synopsis

```
crexsel ( [<inputs> ...] [--uncompile] [--symbols] [--errors] [--undefined]
        [--lineNumbers] [--noXmlnsPrefix] [--verbose] [--debug] ) | ( -(h|help)] )
```

Description

The `crexsel` command invokes the *Rexsel* compiler on a list of input files. Each input file should have the extension "`rxsl`". The XSL output is sent to an output file in the same directory with the same name as the input but with the "`xsl`" extension.

The options that `crexsel` understands:

-h | **--help**	Show the full help. All other options on the command tail are ignored.
-l | **--lineNumbers**	Show source line number comments interleaved in the output XSL.
-n | **--noXmlnsPrefix**	Do not use `xsl:` prefix for XSL tags. The default name space for the document is considered to be '`http://www.w3.org/1999/XSL/Transform`'
-s | **--symbols**	Output the symbol table as a comment at the head of the XSL output. If the verbose flag is set this is repeated on the console.

-u | **--undefined** Output the undefined variables with the errors.

-e | **--errors** Show the error list as a comment at the head of the XSL output. If the verbose flag is set the output is also sent to the console.

-v | **--verbose** Output all messages on the console.

-d | **--debug** Major debugging messages — avoid.

--uncompile Invoke the translation of existing XSLT files into *Rexsel* files.

4.1 `crexsel` Arguments

4.1.1 Help

```
USAGE: crexsel [<inputs> ...] [--uncompile] [--symbols] [--errors] [--undefined]
               [--lineNumbers] [--noXmlnsPrefix] [--verbose] [--debug]

ARGUMENTS:
  <inputs>                Set of Rexsel files to compile to XSLT

OPTIONS:
  --uncompile             Transform XSLT file into Rexsel
  -s, --symbols           Generate symbol table in the code
  -e, --errors            Show error list in output
  -u, --undefined         Show undefined errors (ignore external defines)
  -l, --lineNumbers       Insert Rexsel source line numbers in XSL
  -n, --noXmlnsPrefix     No xsl namespace prefix
  -v, --verbose           Verbose console messages
  -d, --debug             Debug on
  -h, --help              Show help information.
```

4.1.2 Input Files

```
~ % crexsel *.rxsl
~ %
```

Will translate all files matching `*.rxsl` in the current folder, and the output files will be to `*.xsl`. The operation is carried out silently with no output to the console.

4.1.3 Output Line Numbers (-l | --lineNumbers)

Normally the output XSL has no information to link individual lines back to the *Rexsel* source. For example

```
stylesheet {
   version "1.0"

   xmlns "aw" "http://www.hsfr.org.uk/Schema/AntWeave"

   output {
      method text
      version "1.0"
```

```
      omit-xml-declaration yes
      encoding "UTF-8"
   }

   parameter rootChunk
   parameter startCommentString "'//'"
   parameter endCommentString "''"
```

Compiling this using

```
crexsel tangle.rxsl --lineNumbers
```

will produce an output showing the line numbers of the original code that produced the XSL output line.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Produced using Rexsel compiler 1.0 Build 172 on 08/03/2024 01:54:42 -->
<xsl:stylesheet
      xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
      xmlns:aw="http://www.hsfr.org.uk/Schema/AntWeave" version="1.0">
 <!-- Line: 6 -->
<xsl:output  method="text" version="1.0" omit-xml-declaration="yes" encoding="UTF-8"/>

 <!-- Line: 13 -->
<xsl:param  name="rootChunk"/>

 <!-- Line: 14 -->
<xsl:param  name="startCommentString" select="'//'"/>

 <!-- Line: 15 -->
<xsl:param  name="endCommentString" select="''"/>
```

4.1.4 Use Default XSL Namespace (-n | --noXmlnsPrefix)

Most of the examples within this note use the explicit namespace prefix for the XSL elements. However it is possible to inhibit this and make a default namespace. For example

```
crexsel tangle.rxsl --noXmlnsPrefix
```

would produce

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Produced using Rexsel compiler 1.0 Build 172 on 08/03/2024 01:54:42 -->
<stylesheet
   xmlns="http://www.w3.org/1999/XSL/Transform"
   xmlns:aw="http://www.hsfr.org.uk/Schema/AntWeave" version="1.0">
   <output  method="text" version="1.0" omit-xml-declaration="yes" encoding="UTF-8"/>

   <param  name="rootChunk"/>
   <param  name="startCommentString" select="'//'"/>
   <param  name="endCommentString" select="''"/>
```

instead of

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Produced using Rexsel compiler 1.0 Build 172 on 08/03/2024 01:54:42 -->
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:aw="http://www.hsfr.org.uk/Schema/AntWeave" version="1.0">
   <xsl:output  method="text" version="1.0" omit-xml-declaration="yes" encoding="UTF-8"/>

   <xsl:param  name="rootChunk"/>
   <xsl:param  name="startCommentString" select="'//'"/>
   <xsl:param  name="endCommentString" select="''"/>
```

4.1.5 Output Symbol Table (-s | --symbols)

It is possible to generate a list of symbols (proc names etc.) used throughout a stylesheet. The symbol output is shown as a list of "blocks" or ***contexts*** within which the symbols used within that immediate context are shown. See Chapter 5 for a complete description of the format.

```
crexsel tangle.rxsl --symbols
```

will output the symbol table as a comment at the head of the XSL output (see Section 5):

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Produced using Rexsel compiler 1.0 Build 172 on 08/03/2024 02:53:05 -->
<!--
Symbols in context "stylesheet", found: 6
[F] process-code         in line 47 not used
[M] /::                  in line 20
[M] node() | @*::        in line 89
[P] endCommentString     in line 15 not used
[P] rootChunk            in line 13 not used
[P] startCommentString   in line 14 not used

Symbols in context "process-code", found: 3
[V] chunkNode      in line 49 not used
[V] isFirstChunk   in line 48 not used
[V] name           in line 50 not used
 -->
<xsl:stylesheet
   xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
   xmlns:aw="http://www.hsfr.org.uk/Schema/AntWeave" version="1.0">
 <xsl:output  method="text" version="1.0" omit-xml-declaration="yes" encoding="UTF-8"/>

 <xsl:param  name="rootChunk"/>
```

4.1.6 Show Errors (-e | --errors)

Normally the `crexsel` command does not report errors. Use this flag if showing errors (except for the undefined errors, see the '`undefined`' option below).

```
~ % crexsel tangle.rxsl --verbose --errors
Processing tangle.rxsl -> tangle.xsl
No Errors
Finished
~ %
```

Adding an error would give

```
hsfr@bramley ~ % crexsel tangle.rxsl --verbose --errors
Processing tangle.rxsl -> tangle.xsl
**** (105) Unexpected symbol "vale" found in "otherwise" in line 28
     Check spelling, illegal keyword, or missing quote?
Finished
hsfr@bramley ~ %
```

4.1.7 Show Undefined Errors (-u | --undefined)

When checking the source it is useful to distinguish between errors that are specific to the current source file and ignore those errors that are consequential from referencing variables in included files. *Rexsel* makes no attempt to resolve these in the error reports.

In order to see undefined errors the '`errors`' flag must be set. Having a keyword error and an undefined variable would give

```
hsfr@bramley RexselTests % crexsel tangle.rxsl --verbose --errors --undefined
Processing tangle.rxsl -> tangle.xsl
**** (105) Unexpected symbol "vale" found in "otherwise" in line 28
     Check spelling, illegal keyword, or missing quote?
**** (116) Could not find "rotName" in line 39
     Check "rotName" is defined in current block.Finished
hsfr@bramley ~ %
```

Ignoring the '`undefined`' option would only give the "unexpected symbol" error.

```
hsfr@bramley RexselTests % crexsel tangle.rxsl --verbose --errors --undefined
Processing tangle.rxsl -> tangle.xsl

**** Found unexpected symbol "vale" in line 26
     Check spelling or missing quote.
Finished
hsfr@bramley ~ %
```

Sometimes there are only undefined errors which may be as a result of references to included files which are not resolved in *Rexsel*. In this case the compiler will display no errors status but with a proviso: "check undefined".

```
hsfr@bramley ~ % crexsel tangle.rxsl --verbose --errors
Processing tangle.rxsl -> tangle.xsl
No Errors (check undefined)
Finished
hsfr@bramley ~ %
```

Running this again with the '`undefined`' option shows the undefined error.

```
hsfr@bramley RexselTests % crexsel -i tangle.rxsl --verbose --errors --undefined
Processing tangle.rxsl -> tangle.xsl
**** (116) Could not find "rotName" in line 39
     Check "rotName" is defined in current block.
Finished
hsfr@bramley RexselTests %
```

4.1.8 Verbose Errors on the Console (-v | --verbose)

Normally the errors and status reporting is output with the XSL as a set of comments. Using the '`verbose`' flag will send all errors and status messages to the console. Taking the extended example in Section 5 the following will output the symbol table to the console (or confirmation of error status).

```
hsfr@bramley ~ % crexsel tangle.rxsl --verbose --symbols
Processing tangle.rxsl -> tangle.xsl

Symbols in context "stylesheet", found: 6
[F] process-code         in line 47  used in line(s) 36, 69
[M] /::                  in line 20
[M] node() | @*::        in line 89
[P] endCommentString     in line 15  used in line(s) 57
[P] rootChunk            in line 13  used in line(s) 22, 25, 29
[P] startCommentString   in line 14  used in line(s) 54

Symbols in context "forEach://aw:chunk[(@name = '*' or @name = $rootChunk) and
                                                        @type = 'code']", found: 2
[V] firstChunkPosition   in line 34  used in line(s) 37
[V] rootName             in line 23  used in line(s) 39
```

```
Symbols in context "process-code", found: 3
[V] chunkNode      in line 49  used in line(s) 52
[V] isFirstChunk   in line 48  used in line(s) 53
[V] name           in line 50  used in line(s) 57

Symbols in context "when:name() = 'aw:include'", found: 1
[V] chunkRef   in line 67  used in line(s) 71, 72

Finished
hsfr@bramley ~ %
```

4.1.9 Translate XSLT Files into *Rexsel*. (--uncompile)

To assist the production of *Rexsel* scripts from existing XSLT scripts, the `crexsel` offers an "uncompile" proc that translates XSLT into *Rexsel*. The uncompiler is certainly not foolproof, especially when xml entities are used, and the code would require checking afterwards to ensure accuracy; it also does not translate embedded comments. However it gives a good starting point when translating legacy XSLT scripts, converting most scripts accurately.

For example

```
hsfr@bramley widgets % crexsel buildSiteWidget-latex.xsl --uncompile -v
Processing buildSiteWidget-latex.xsl to buildSiteWidget-latex.rxsl
Finished
hsfr@bramley widgets %
```

The '`verbose`' flag is the only valid option here. Normally the whole process runs silently.

Chapter 5

Symbol Table

The symbol table stores the defined variables (constants), parameters, functions, and matches that have been declared in the stylesheet. They are output in a simple format as required. The format of the output is

```
Symbols in context "<context hame>" found: <number of symbols found in that context>
[<type of symbol 1>] <symbol name 1> in line <line where symbol declared 1> <where used>
[<type of symbol 2>] <symbol name 2> in line <line where symbol declared 2> <where used>
```

where a context is defined as the locality within a procedure etc. that provides the scope of each variable.

The symbol type can be *proc* (F), *variable/constant* (V), *attribute-set* (A), *parameter* (P), or *match* (M). Where the symbol type is associated with a *match* the XPath expression and the scope are used together in the format '`xpath::scope`'. For example if the source is

```
match using "node() | @*" priority "-1" scope "inline-text" {
```

the symbol table entry would be, for example

```
[M] node() | @*/::inline-text          in line 20
```

A variable used within a particular scope has access to defined variables and parameters inherited from its parent scope and then up to the top level. By definition the top level (within *stylesheet*) is global and all enclosed variables/parameters have access to those declared global.

For example the symbol table for the extended example `tangle.rxsl` would be

```
Symbols in context "stylesheet", found: 6
[F] process-code         in line 47  used in line(s) 36, 69
[M] /::                  in line 20
[M] node() | @*::        in line 89
[P] endCommentString     in line 15  used in line(s) 57
[P] rootChunk            in line 13  used in line(s) 22, 25, 29
[P] startCommentString   in line 14  used in line(s) 54

Symbols in context "forEach://aw:chunk[(@name = '*' or @name = $rootChunk) and @type = 'code']", found: 2
[V] firstChunkPosition   in line 34  used in line(s) 37
[V] rootName             in line 23  used in line(s) 39

Symbols in context "process-code", found: 3
[V] chunkNode      in line 49  used in line(s) 52
[V] isFirstChunk   in line 48  used in line(s) 53
[V] name           in line 50  used in line(s) 57
```

```
Symbols in context "when:name() = 'aw:include'", found: 1
[V] chunkRef   in line 67  used in line(s) 71, 72
```

To show a more contrived example to illustrate more complex contexts

```
stylesheet {
    version "1.0"

    parameter param-1

    variable globalConstant-0 "$globalConstant-2"

    variable globalConstant-1 {
        variable localConstant-2 "'Inner String 1'"
        value "$localConstant-2âĂİ
    }

    variable globalConstant-2 {
        variable localConstant-5 "'xxx'"
        variable localConstant-6 {
            variable localConstant-7 "'yyy"
            value "$globalConstant-1"

            variable localConstant-8 {
                variable localConstant-9 "$localConstant-7"
                value "$localConstant-9"
                value "$localConstant-2"
            }
        }
    }
}
```

Compiling this with

```
crexsel tangle.rxsl --errors --symbols
```

gives

```
**** (116) Could not find "localConstant-2" in line 22
     Check "localConstant-2" is defined in current block/context.

Symbols in context "stylesheet", found: 4
[P] param-1            in line 4 not used
[V] globalConstant-0   in line 6 not used
[V] globalConstant-1   in line 8  used in line(s) 17
[V] globalConstant-2   in line 13  used in line(s) 6

Symbols in context "globalConstant-1", found: 1
[V] localConstant-2   in line 9  used in line(s) 10

Symbols in context "globalConstant-2", found: 2
[V] localConstant-5   in line 14 not used
[V] localConstant-6   in line 15 not used

Symbols in context "localConstant-6", found: 2
[V] localConstant-7   in line 16  used in line(s) 20
[V] localConstant-8   in line 19 not used

Symbols in context "localConstant-8", found: 1
[V] localConstant-9   in line 20  used in line(s) 21
```

This shows that *localConstant-2* in the *globalConstant-1* context cannot be accessed in the *globalConstant-2* context.

Chapter 6

Language Description

All of *Rexsel* keywords map, more or less, to an equivalent XSLT construction. Where it is possible the same keyword is related directly to the XSLT element. The examples that follow use the default XML namespace for XSLT, thus not requiring the '`xsl:`' prefix with the XSLT elements.

The generalised form of a *Rexsel* statement is

```
keyword options { block elements }
```

where keywords are valid instructions (*stylesheet*, *variable*, etc.); options are an optional list of names or expressions, either single or in pairs; and block elements are a list of valid instructions enclosed by brackets. If there are no block elements required then the brackets can be ignored.

Instructions can obviously be nested

```
keyword options {
   keyword options { block elements }
   keyword options { block elements }
}
```

6.1 Comments

Comments are ignored by the *Rexsel* compiler and are identical to comments in C, Swift, etc. Single-line comments begin with two forward-slashes (//). Multi-line comments are not currently supported, but will be added in future versions.

6.2 XSLT to *Rexsel* Map Equivalents

XSLT	*Rexsel*
<apply-imports>	apply-imports (6.3)
<apply-templates>	apply-templates using "XPath" (6.4)
<attribute name="attrib-name">	attribute "attrib-name" (6.5)
<attribute-set name="attrib-set-name">	attribute-set "attrib-set-name" {...} (6.6)
<call-template name="function-name">	call fnName { with "param-name" "value" } (6.7)
<choose>	choose {...} (6.8)
<comment>	comment "..." (6.9)
<copy>	copy {...} (6.10)
<copy-of>	copy-of "..." (6.11)
<decimal-format>	decimal-format {...} (6.12)
<element>	element "..." {...} (6.13)
<for-each>	foreach "..." {...} (6.14)
<if>	if "..." {...} (6.15)
<import>	import "..." (6.16)
<key>	key using "xpath-expr" {...} (6.17)
<message>	message terminate "yes"|"no" {...} (6.19)
<namespace-alias>	namespace-alias map-from "..." to "..." (6.20)
<number>	number {...} (6.21)
<otherwise>	otherwise {...} (6.8)
<output>	output {...} (6.22)
<param>	parameter name {...} (6.23)
<preserve-space>	preserve-space "..." (6.24)
<processing-instruction>	processing-instruction "..." {...} (6.26)
<sort>	sort {...} (6.27)
<strip-space>	strip-space "..." (6.28)
<stylesheet>	stylesheet {...} (6.29)
<stylesheet version="1.0">	stylesheet { version "1.0" ...} (6.33)
<stylesheet xmlns:prefix="URI">	stylesheet { xmlns "prefix" "URI" ...} (6.34)
<template name="fn-name">	proc fnName {...} (6.25)
<template match="xpath-expr">	match using "xpath-expr" {...} (6.18)
<text>	text "..." (6.30)
<value-of>	value "..." (6.31)
<variable>	variable name {...} (6.32)
<when>	when "..." {...} (6.8)
<with-param>	with name {...} (6.7)

<fallback> is not currently supported.

6.3 apply-imports

Import precedence requires that template rules in main stylesheets have higher precedence than template rules in imported stylesheets. Sometimes, however, it is useful to be able to force the processor to use a template rule from the (lower precedence) imported stylesheet rather than an equivalent rule in the main stylesheet.

Syntax

<apply-imports> ::= "`apply-imports`"

Options

None

Elements

None

Examples

'`apply-imports`' statements are standalone and do not have an associated value or block.

```
match using "//list" {
    apply-imports
}
```

giving

```
<template  match="//list">
    <apply-imports/>
</template>
```

Errors

'`apply-templates`' can only be used within a '`match`' or a '`function`' statement.

```
stylesheet {
    version "1.0"

    apply-imports
}
```

would give

```
**** (139) Found unexpected reserved word "apply-imports" in "stylesheet" in line 1
     Check spelling.
```

6.4 apply-templates

The *apply-templates* keyword instructs the XSLT to recursively process any the children of the source element.

Syntax

<apply-templates> ::= "`apply-templates`"
("`using`" <xpath expression>)? ("`scope`" <name>)?
("{" (<with>)* (<sort>)? "}")?

Options

using (optional) an XPath expression indicating the node and children to match.

scope (optional) the scope (mode) of this template, matching that defined in an '`match`' statement.

Elements

with (optional) a parameter carried into the invoked match templates.

sort (optional) the definition of how the match templates should be applied (6.27).

Examples

```
match using "//list" {
    apply-templates {
        with param-1 "'List Title'"
        sort using "a" descending
    }
}
```

This says that when any list is detected then the children of '`//list`' are processed with a parameter *param-1* with a value of "`'List Title'`" (a simple string), and the results of applying the templates sorted with the <a> element in descending order.

```
<template  match="/">
    <apply-templates>
        <with-param  name="param-1" select="'List Title'"/>
        <sort select="a" order="descending"/>
    </apply-templates>
</template>
```

Note that if the '`decending`' instruction had been '`ascending`' or omitted then this is the default value and the '`order`' attribute would be also omitted in the output XSLT.

As an aside the above can also be written as

```
match using "//list" {
    apply-templates {
        with param-1 {
```

```
            text "List Title"
        }
        sort using "a" descending
    }
}
```

with the same effect.

Errors

A simple misspelling

```
stylesheet {
    version "1.0"

    match using "node() | @*" priority "-1" scope "inline-text" {
        copy {
            apply-templates usin "@*"
            apply-templates
        }
    }

}
```

will give the error

```
**** (106) Unexpected symbol "usin" instead of "'using' or 'scope'" in "apply-templates" in line 6
     Check spelling, illegal keyword, or missing bracket/quote?
```

An illegal or misspelled keyword on the following line

```
stylesheet {
    version "1.0"

    match using "node() | @*" priority "-1" scope "inline-text" {
        copy {
            apply-templates using "@*"
            apply-template
        }
    }
}
```

will give a similar error

```
**** (106) Unexpected symbol "apply-template" instead of "'using' or 'scope'" in "apply-templates" in line 7
     Check spelling, illegal keyword, or missing bracket/quote?
```

6.5 attribute

attributes are associated with elements and attribute lists. Note that the first construction here is translated to a block in the output XSLT. This is not a direct reflection of XSLT which only allows a block or a text content.

Syntax

<attribute> ::= "`attribute`" <quote> <name> <quote>
("`namespace`" <URI>)?
(<quote> <alphanumeric string> <quote> | "{" (<block template>)+ "}")

Options

name		a QName defining the name of this attribute.
namespace	(optional)	a namespace associated with this attribute.
alphanumeric string		an optional string defining a simple text value for the named attribute

Elements

One or more of the block elements defined in Section **??**. If any elements are defined then there should not be an string value defined and *vice versa*.

Examples

A simple element definition with two defined attributes

```
element "script" {
    attribute "src" {
       value "concat( $inScriptsDir, $inScript )"
    }
    attribute "type"  "text/javascript"
}
```

would compile to

```
<element name="script">
    <attribute name="src">
        <value-of select="concat( $inScriptsDir, $inScript )"/>
    </attribute>
    <attribute name="type">
        <text>text/javascript</text>
    </attribute>
</element>
```

This shows the two forms of the '`attribute`' statement.

In the above, note that if it is required to have an XPath expression (for example with a variable) then the '`value`' command must be used. For example

```
attribute "attrib-1" {
   value "$var-1"
}
```

would give the following XSLT

```
<xsl:attribute name="attrib-1">
    <xsl:value-of select="$var-1"/>
</xsl:attribute>
```

Adding an explicit namespace is done with the *namespace* option:

```
 element "div" {
    attribute "attrib-1" namespace "http://www.w3.org/1999/xhtml" "value-1"
 }
```

giving

```
<element name="div">
   <attribute name="attrib-1" namespace="http://www.w3.org/1999/xhtml">
      <text>value-1</text>
   </attribute>
</element>
```

Errors

Having an attribute value and a block

```
stylesheet {
    version "1.0"

     proc tryout {
        element "div" {
            attribute "attrib-1"  "'value-1'" {
                value "'value-2'"
            }
        }
    }
}
```

gives the error

```
**** (128) A variable/parameter cannot have default and enclosed templates in line 6
     Remove either default/select or enclosed templates.
```

Having an attribute value and a block, as well as a misspelled keyword

```
stylesheet {
    version "1.0"

     proc tryout {
       element "div" {
           attribute "attrib-1"  "'value-1'" {
              valu "'value-2'"
           }
       }
   }
}
```

gives the errors (136 is a consequential error)

```
**** (128) A variable/parameter cannot have default and enclosed templates in line 6
     Remove either default/select or enclosed templates.
**** (105) Unexpected symbol "valu" found in "attribute" in line 7
     Check spelling, missing expression, bracket or quote?
**** (136) Unmatched brackets in 1
     Too many open brackets?
```

6.6 attribute-set

An *attribute-set* creates a named set of attributes, which can be used within other elements my referencing the name.

Syntax

<attribute-set> ::= “`attribute-set`” <quote> <name> <quote>
(“`use-attribute-sets`” <name list>)?
“{”
(<attribute>)+
“}”

Options

name		a QName defining the name of this attribute
use-attribute-sets	(optional)	a string containing a whitelist separated list of other sets.

Elements

One or more attribute definitions.

Examples

A trivial example for having common attributes which might occur throughout an HTML file.

```
stylesheet {
   version "1.10"

    attribute-set "block-style" {
        attribute "font-size" "12pt"
        attribute "font-weight" "bold"
    }

    attribute-set "block-margin" {
        attribute "leading" "12pt"
        attribute "margin-top" "24pt"
    }

    match using "/" {
        element "block" {
            attribute "id" "start"
            use-attribute-sets "block-style block-margin"

            text "Some block text"
        }
    }

}
```

would compile to

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.10">
    <xsl:attribute-set name="block-style">
        <xsl:attribute name="font-size">
            <xsl:text>12pt</xsl:text>
        </xsl:attribute>
        <xsl:attribute name="font-weight">
            <xsl:text>bold</xsl:text>
        </xsl:attribute>
    </xsl:attribute-set>

    <xsl:attribute-set name="block-margin">
        <xsl:attribute name="leading">
            <xsl:text>12pt</xsl:text>
        </xsl:attribute>
        <xsl:attribute name="margin-top">
            <xsl:text>24pt</xsl:text>
        </xsl:attribute>
    </xsl:attribute-set>

    <xsl:template match="/">
        <xsl:element name="block" use-attribute-sets="block-style block-margin">
            <xsl:attribute name="id">
                <xsl:text>start</xsl:text>
            </xsl:attribute>
            <xsl:text>Some block text</xsl:text>
        </xsl:element>
    </xsl:template>

</xsl:stylesheet>
```

To show how the above works, consider a one line XML file:

```
<?xml version="1.0" encoding="UTF-8"?>

<test-attribute/>
```

Running the XSLT we get

```
<?xml version="1.0" encoding="UTF-8"?>
<block
   font-size="12pt" font-weight="bold"
   leading="12pt" margin-top="24pt"
   id="start">Some block text</block>
```

Errors

Forgetting the leading block bracket:

```
attribute-set "block-style"
     attribute "font-size" "12pt"
     attribute "font-weight" "bold"
}
```

will give

```
**** (104) Missing "{" in line 5
     Insert bracket.
```

Forgetting the name of the attribute set

```
stylesheet {
   version "1.10"

    attribute-set {
```

```
            attribute "font-size" "12pt"
            attribute "font-weight" "bold"
        }

        attribute-set "block-margin" {
            attribute "leading" "12pt"
            attribute "margin-top" "24pt"
        }

        match using "/" {
            element "block" {
                attribute "id" "start"
                use-attribute-sets "block-style block-margin"

                text "Some block text"
            }
        }

    }
```

will give two errors

```
    **** (115) Expected name after "attribute-set" in line 4
         Syntax: attribute-set <qname>  <block>.
    **** (116) Could not find "block-style" in line 15
         Check "block-style" is defined in current block.
```

6.7 call/with

To invoke a named proc the *call* is used. A name must be given together with a set of optional parameters (‘`with`’) passed through to the function.

Syntax

<call> ::= “`call`” <quote> <name> <quote> (“{” (<with>)+ “}”)?

Options

name a name defining the name of the procedure to call.

Elements

Zero or more ‘`with`’ parameter definitions within a block.

Examples

For example the following is a recursive routine *common.substring-after-last* that includes a call to itself.

```
proc common.substring-after-last {
  parameter string
  parameter delimiter
  choose {
     when "contains($string, $delimiter)" {
       call common.substring-after-last {
          with string "substring-after($string, $delimiter)"
          with delimiter "$delimiter"
       }
     }
     otherwise {
        value "$string"
     }
  }
}
```

which translates to a named XSLT template

```
<template name="common.substring-after-last">
    <param name="string"/>
    <param name="delimiter"/>

    <choose>
        <when test="contains($string, $delimiter)">
            <call-template name="common.substring-after-last">
                <with-param name="string" select="substring-after($string, $delimiter)"/>
                <with-param name="delimiter" select="$delimiter"/>
            </call-template>
        </when>
        <otherwise>
            <value-of select="$string"/>
```

```
        </otherwise>
    </choose>
</template>
```

Errors

Taking the above example, spelling the name incorrectly

```
call common.substring-after-las {
   with string "substring-after($string, $delimiter)"
   with delimiter "$delimiter"
}
```

will produce the error

```
**** (116) Could not find "common.substring-after-las" in line 9
     Check "common.substring-after-las" is defined in current block.
```

6.8 choose/when/otherwise

choose provides a multi choice statement equivalent to a switch statement in languages such as Swift or C. Like these languages it provides for default conditions when all the suggested alternatives fail.

Switch cases use the '`when`' keyword with a test and an enclosed block if that test is true. If none of the cases is true then, either the choose statement exits, or the '`otherwise`' block is run.

Syntax

<choose> ::= "`choose`" "{"
(<when>)+
(<otherwise>)?
"}"

<when> ::= "`when`" <quote> <xpath> <quote> "{"
(<block template>)+
"}"

<otherwise> ::= "`otherwise`" "{"
(<block template>)+
"}"

Options

None

Required Elements

One or more '`when`' blocks.

Examples

As an example the following sets a variable *gPageTitle* dependent on a condition

'`string-length(//page:meta/page:pageTitle) > 0`'

and gives a default value. In this case it is equivalent to a '`if ... then ... else ...`' statement.

```
variable gPageTitle {
    choose {
        when "string-length(//page:meta/page:pageTitle) > 0" {
            value "//page:meta/page:pageTitle"
        }
        otherwise {
            value "'Paloose'"
        }
```

```
        }
    }
```

will result in

```
<xsl:variable  name="gPageTitle">
     <xsl:choose>
          <xsl:when test="string-length(//page:meta/page:pageTitle) > 0">
               <xsl:value-of  select="//page:meta/page:pageTitle"/>
          </xsl:when>
          <xsl:otherwise>
               <xsl:value-of  select="'Paloose'"/>
          </xsl:otherwise>
      </xsl:choose>
</xsl:variable>
```

A slightly larger, multi choice, example might be

```
choose {
    when "@type = 'dir'" {
        element "span" {
            attribute "class" "code-dir"
            apply-templates scope "inline-text"
        }
    }
    when "@type = 'var'" {
        element "span" {
            attribute "class" "code-var"
            apply-templates scope "inline-text"
        }
    }
    when "@type = 'tag'" {
        element "span" {
            attribute "class" "code-dir"
            text "&lt;"
            apply-templates scope "inline-text"
            text "&gt;"
        }
    }
    otherwise {
        apply-templates scope "inline-text"
    }
}
```

compiled to

```
<xsl:choose>
    <xsl:when test="@type = 'dir'">
        <xsl:element name="span">
            <xsl:attribute name="class">
                <xsl:text>code-dir</xsl:text>
            </xsl:attribute>
            <xsl:apply-templates mode="inline-text"/>
        </xsl:element>
    </xsl:when>
    <xsl:when test="@type = 'var'">
        <xsl:element name="span">
            <xsl:attribute name="class">
                <xsl:text>code-var</xsl:text>
            </xsl:attribute>
            <xsl:apply-templates mode="inline-text"/>
        </xsl:element>
    </xsl:when>
    <xsl:when test="@type = 'tag'">
        <xsl:element name="span">
            <xsl:attribute name="class">
                <xsl:text>code-dir</xsl:text>
            </xsl:attribute>
            <xsl:text>&lt;</xsl:text>
```

```
            <xsl:apply-templates mode="inline-text"/>
            <xsl:text>&gt;</xsl:text>
        </xsl:element>
    </xsl:when>
    <xsl:otherwise>
        <xsl:apply-templates mode="inline-text"/>
    </xsl:otherwise>

</xsl:choose>
```

Errors

The most common error is misspelling

```
variable gPageTitle {
    choose {
        when "string-length(//page:meta/page:pageTitle) > 0" {
            value "//page:meta/page:pageTitle"
        }
        otherwie {
            value "'Paloose'"
        }
    }
}
```

will produce the error (together with potential consequential errors)

```
**** (105) Unexpected symbol "otherwie" found in "choose" in line 11
     Check spelling, illegal keyword, or missing quote?
```

6.9 comment

A ***comment*** statement (not to be confused with a *Rexsel* comment) injects an XML comment to the output XSLT document.

Syntax

<comment> ::= "`comment`" <quote> <alphanumeric string> <quote>

Options

None

Required Elements

None

Examples

For example

```
proc func-1 {
   comment 'function click(event) {
         var mark = event.target;
         while ((mark.className != "b") && (mark.nodeName != "BODY")) {
             mark = mark.parentNode
         }
      }'
}
```

compiles to

```
 <template name="func-1">
     <comment><![CDATA[function click(event) {
         var mark = event.target;
         while ((mark.className != "b") && (mark.nodeName != "BODY")) {
             mark = mark.parentNode
         }
      }]]></comment>

</template>
```

Note that the type of enclosing quotes is quite important here. In the example above the comment text in the example above is enclosed with single quotes. This is because of the use of double quote occurring within the text. If required you can use escaped quotes for the same effect.

```
proc func-1 {
   comment "
      function click(event) {

         var mark = event.target;
         while ((mark.className != \"b\") && (mark.nodeName != \"BODY\")) {
```

```
            mark = mark.parentNode
        }
    }"
}
```

Errors

The primary error is forgetting the correct quotes. To take the above example and replace the enclosing single quotes with double quotes.

```
proc func-1 {
   comment "function click(event) {
         var mark = event.target;
         while ((mark.className != "b") && (mark.nodeName != "BODY")) {
             mark = mark.parentNode
         }
     }"
}
```

This would give the errors

```
**** (105) Unexpected symbol "b" found in "proc" in line 7
     Check spelling, illegal keyword, or missing quote?
**** (136) Unmatched brackets in 1
     Too many open brackets?
```

Obviously curable by escaping the internal quotes or replacing the enclosing quotes.

6.10 copy

The *copy* transfers a shallow copy (the node and any associated namespace node) of the current node to the output document. It does not copy any children or attributes of the current node.

Syntax

<copy> ::= "copy" ("use-attribute-sets" <URI>)? "{"
(<block template>)+
"}")

Options

use-attribute-sets a string containing a whitelist separated list of other sets.

Elements

One or more block statements.

Examples

For example

```
match using "node() | @*" priority "-1" scope "inline-text" {
    copy {
        apply-templates using "@*"
        apply-templates
    }
}
```

will result in

```
<template match="node() | @*" mode="inline-text" priority="-1">
    <copy>
        <apply-templates select="@*"/>
        <apply-templates/>
    </copy>
</template>
```

which will apply templates for the attributes and descendants of the current node (similar to 'copy-of').

Errors

Forgetting the open bracket

```
stylesheet {
    version "1.0"
```

```
        match using "node() | @*" priority "-1" scope "inline-text" {
            copy
                apply-templates using "@*"
                apply-templates
            }
        }
    }
```

gives the error

```
**** (105) Unexpected symbol "apply-templates" found in "copy" in line 6
     Check spelling, illegal keyword, or missing bracket/quote?
```

6.11 copy-of

The *copy-of* makes a deep copy including all descendant nodes specified by the associated XPath expression.

Syntax

<copy-of> ::= "`copy-of`" <quote> <xpath expression> <quote>

Options

None.

Elements

None.

Examples

For example

```
match using "pcms:scripts" scope "buildPagesList.properties" {
    copy-of "."
}
```

will result in

```
<xsl:template match="pcms:scripts" mode="buildPagesList.properties">
    <xsl:copy-of select="."/>
</xsl:template>
```

which copies all the descendant nodes and attributes of the node '`pcms:scripts`'.

Errors

The common error is forgetting a quote round the expression:

```
stylesheet {
    version "1.0"

    // Pass through (written so that can be expanded later more easily).

        match using "scripts" scope "buildPagesList.properties" {
                copy-of ".
        }
}
```

which will give

```
**** (105) Unexpected symbol ".
        }

}
" found in "copy-of" in line 11
     Check spelling, illegal keyword, or missing bracket/quote?
**** (136) Unmatched brackets in 1
     Too many open brackets?
```

6.12 decimal-format

decimal-format is a root level statement that defines the characters and symbols that are to be used in converting numbers into strings. The various names within the block are identical to the XSLT attributes.

Syntax

```
<decimal-format>   ::=   “decimal-format” <name> “{”
                         (
                              ( “decimalSeparator” <quote> <alpha character> <quote> )? |
                              ( “groupingSeparator” <quote> <alpha character> <quote> )? |
                              ( “infinity” <quote> <alphanumeric string> <quote> )? |
                              ( “minusSign” <quote> <alphanumeric character> <quote> )? |
                              ( “notNumber” <quote> <alphanumeric string> <quote> )? |
                              ( “percent” <quote> <alphanumeric character> <quote> )? |
                              ( “perMille” <quote> <alphanumeric character> <quote> )? |
                              ( “zeroDigit” <quote> <alphanumeric character> <quote> )? |
                              ( “digit” <quote> <alphanumeric character> <quote> )? |
                              ( “patternSeparator” <quote> <alphanumeric character> <quote> )?
                         )+
                         “}”
```

Elements

The range of values within an ‘`output`’ block is identical to the XSLT definition.

decimal-separator	Specifies the thousands separator character. The default is (,).
grouping-separator	Specifies the thousands separator character. The default is (,).
infinity	Specifies the string used to represent infinity. The default is the string “Infinity”.
NaN	Specifies the string used when the value is not a number. The default is the string “NaN”.
percent	Specifies the percentage character. The default is (%).
perMille	Specifies the per thousand character. The default is (‰).
zeroDigit	Specifies the digit zero character. The default is (0).
digit	Specifies the character used in the format pattern to stand for a digit. The default is (#).
patternSeparator	Specifies the character separating positive and negative subpatterns in a format pattern. The default is the semi-colon (;).

Options

Name is always required.

Elements

At least one of the included elements.

Examples

A simple example to define the format of numbers: the thousands separator and decimal separator.

```
stylesheet {
    version "1.0"

    decimal-format decName {
       decimal-separator ","
       grouping-separator "."
    }
}
```

which outputs the following

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet
      xmlns="http://www.w3.org/1999/XSL/Transform"
      version="1.0">

   <decimal-format
      name="decName"
      decimal-separator=","
      grouping-separator="."/>
</stylesheet>
```

Errors

Forgetting the name:

```
stylesheet {
    version "1.0"

    decimal-format {
       decimal-separator "."
       grouping-separator "."
    }
}
```

will give

```
**** (115) Expected name after "decimal-format" in line 4
     Syntax: decimal-format <qname>  <block>.
```

Having an empty expression:

```
stylesheet {
    version "1.0"
```

```
    decimal-format {
       decimal-separator "."
       grouping-separator ""
    }
}
```

will give

```
**** (140) Null string instead of "valid separator" in "grouping-separator" in line 6
     Check string expression.
```

Leaving out the expected expression completely:

```
stylesheet {
    version "1.0"

    decimal-format {
       decimal-separator
       grouping-separator ","
    }
}
```

will give

```
**** (105) Unexpected symbol "grouping-separator" found in "decimal-separator" in line 6
     Check spelling, illegal keyword, or missing bracket/quote?
```

Similarly with the last entry:

```
stylesheet {
    version "1.0"

    decimal-format {
       decimal-separator "."
       grouping-separator
    }
}
```

will give

```
**** (105) Unexpected symbol "}" found in "grouping-separator" in line 7
     Check spelling, illegal keyword, or missing bracket/quote?
```

6.13 element

An *element* phrase defines an XML element in the output stream. It has a name and a block containing other keywords.

Syntax

<element> ::= “element” <quote> <name> <quote>
(“namespace” <quote> <URI> <quote>)?
(“use-attribuite-sets” <quote> <name list> <quote>)? “{”
(<block template>)*
“}”

Options

name the name of the element.

namespace the XML namespace to be defined for this element (optional).

Elements

Zero or more of the block elements.

Examples

For example the scrap

```
stylesheet {
    version "1.0"

    match using "/" {

        element "body" {
            attribute "name" "topOfPage"

            element "a" {
                attribute "name" "topOfPage"
            }
            element "div" {
                attribute "id" "bodyFrame"
                apply-templates
            }
        }
    }
}
```

would compile to

```
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <template match="/">
        <element name="body">
            <attribute name="name">
                <text>topOfPage</text>
            </attribute>
```

```
            <element name="a">
                <attribute name="name">
                    <text>topOfPage</text>
                </attribute>
            </element>

            <element name="div">
                <attribute name="id">
                    <text>bodyFrame</text>
                </attribute>
                <apply-templates/>
            </element>
        </element>
    </template>
</stylesheet>
```

There are two optional parameters that can specify name-spaces and attribute set for the element and its enclosed block. So for example the above might be extended

```
stylesheet {
    version âĂ1.0âĂİ

    match using "/" {

        element "body" namespace "http://www.w3.org/1999/xhtml" {
            attribute "name" "topOfPage"

            element "a" namespace "http://www.w3.org/1999/xhtml" {
                attribute "name" "topOfPage"
            }
            element "div" namespace "http://www.w3.org/1999/xhtml" {
                attribute "id" "bodyFrame"
                apply-templates
            }
        }
    }
}
```

and compile to

```
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <template  match="/">
        <element  name="body" namespace="http://www.w3.org/1999/xhtml">
            <attribute  name="name">
                <text>topOfPage</text>
            </attribute>

            <element  name="a" namespace="http://www.w3.org/1999/xhtml">
                <attribute  name="name">
                    <text>topOfPage</text>
                </attribute>
            </element>

            <element  name="div" namespace="http://www.w3.org/1999/xhtml">
                <attribute  name="id">
                    <text>bodyFrame</text>
                </attribute>
                <apply-templates />
            </element>
        </element>
    </template>
</stylesheet>
```

Errors

No opening bracket:

```
stylesheet {
    version "1.0"

    match using "/" {
       element "div"
            attribute "id" "bodyFrame"
            apply-templates
        }
    }
}
```

will give

```
**** (105) Unexpected symbol "attribute" found in "element" in line 6
     Check spelling, illegal keyword, or missing bracket/quote?
```

Leaving out the namespace expression:

```
stylesheet {
    version "1.0"

    match using "/" {
       element "div" namespace {
            attribute "id" "bodyFrame"
            apply-templates
        }
    }
}
```

will give

```
**** (106) Unexpected symbol "{" instead of "expression" in "element" in line 5
     Check spelling, illegal keyword, or missing bracket/quote?
**** (136) Unmatched brackets in 1
     Too many open brackets?
```

6.14 foreach

'`foreach`' allows simple loops based on an XPath expression.

Syntax

<foreach> ::= "`foreach`" <quote> <xpath> <quote>
"{"
(<sort>)?
(<block template>)+
"}"

Required Options

None

Required Elements

Although there are no required elements within the block, it is meaningless to have an empty loop block and does not return any values (string characters).

Examples

For example consider the following

```
foreach "*" {
    choose {
            when "name() = 'aw:include'" {
            variable chunkRef "@ref"
            call process-code {
                with chunkNode "//aw:chunk[@name = $chunkRef]"
                with name "$chunkRef"
            }
        }
        otherwise {
            value disable-output-escaping "."
        }
    }
}
```

which translates to

```
<for-each select="*">
    <choose>
        <when test="name() = 'aw:include'">
            <variable name="chunkRef" select="@ref"/>

            <call-template name="process-code">
                <with-param name="chunkNode" select="//aw:chunk[@name = $chunkRef]"/>
                <with-param name="name" select="$chunkRef"/>
            </call-template>
        </when>
        <otherwise>
            <value-of select="." disable-output-escaping="yes"/>
```

```
        </otherwise>
    </choose>
</for-each>
```

Errors

Forgetting the XPath expression

```
foreach {
    choose {
            when "name() = 'aw:include'" {
            variable chunkRef "@ref"
            call process-code {
                with chunkNode "//aw:chunk[@name = $chunkRef]"
                with name "$chunkRef"
            }
        }
        otherwise {
            value disable-output-escaping "."
        }
    }
}
```

will give the error

```
**** (105) Unexpected symbol "{" found in "foreach" in line 6
     Check spelling, missing expression, bracket or quote?
```

Similarly forgetting the open block bracket

```
foreach "*"
    choose {
            when "name() = 'aw:include'" {
            variable chunkRef "@ref"
            call process-code {
                with chunkNode "//aw:chunk[@name = $chunkRef]"
                with name "$chunkRef"
            }
        }
        otherwise {
            value disable-output-escaping "."
        }
    }
}
```

reports

```
**** (105) Unexpected symbol "{" found in "foreach" in line 6
     Check spelling, missing expression, bracket or quote?
```

the '`sort`' statement has to be the first in the block

```
foreach "*" {
    choose {
            when "name() = 'aw:include'" {
            variable chunkRef "@ref"
            call process-code {
                with chunkNode "//aw:chunk[@name = $chunkRef]"
                with name "$chunkRef"
            }
        }
        otherwise {
            value disable-output-escaping "."
        }
```

```
        }
        sort ascending
    }
```

reports

```
    **** (141) Sort in "foreach:*" in line 19 must follow declaration.
         Check order.
```

6.15 if

The ***if*** statement provides conditional construction that allows a set of templates to be processed based on an XPath expression.

It does not provide an '`else`' function, however this is simply done using the '`choose ... when ... otherwise ...`' statement.

Syntax

<if> ::= "`if`" <quote> <xpath> <quote> "{"
(<block template>)+
"}"

Options

None

Elements

Although there are no required elements within the condition block, it is meaningless to have a condition that is null and does not return any values (string characters).

Examples

The following example outputs the necessary `<div>` element to produce a breadcrumb panel when the '`id`' attribute matches the variable *gPage*.

```
stylesheet {
    version "1.0"

     proc buildBody.htmlBody {

        if "//breadcrumbs//list:item[@id = $gPage]" {
            element "div" {
                attribute "id" "breadcrumbPanel"

            }
        }

    }
}
```

will result in

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <template name="buildBody.htmlBody">
        <if test="//breadcrumbs//list:item[@id = $gPage]">
            <element name="div">
                <attribute name="id">
                    <text>breadcrumbPanel</text>
```

```
                </attribute>
            </element>
        </if>
    </template>
</stylesheet>
```

Errors

Leaving out the test

```
stylesheet {
    version "1.0"

     proc buildBody.htmlBody {

        if  {
            element "div" {
                attribute "id" "breadcrumbPanel"

            }
        }

    }
}
```

will produce the error

```
**** (122) Missing test expression in line 6
     Insert test.
```

6.16 import/include

The *import* only occurs at the root level of the stylesheet and used to import the contents of another stylesheet into the currently declared sheet.

Syntax

<import> ::= ("`import`" | "`include`") <quote> <uri> <quote>

Required Options

<uri> The URI of the include file.

Required Elements

None

Examples

Using an example in the W3C specification[1] one stylesheet could import sheet `doc.xsl` and modify the treatment of example <`elements`> as follows:

```
stylesheet {
   version "1.0"

   import "doc.xsl"

    match using "example" scope "example-1" {
       element "div" {
           attribute "style" "border: solid red"
           apply-imports
       }
    }
}
```

would give the following XSLT.

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">
   <import  href="doc.xsl"/>

   <template  match="example" mode="example-1">
      <element  name="div">
         <attribute  name="style">
            <text>border: solid red</text>
         </attribute>
         <apply-imports/>
      </element>
   </template>

</stylesheet>
```

[1] https://www.w3.org/TR/xslt-10/#apply-imports

Errors

The only error here (until URI checking is added) is a missing URI reference.

```
stylesheet {
    version "1.0"

    import

    match using "example" scope "example-1" {
       element "div" {
           attribute "style" "border: solid red"
           apply-imports
       }
    }

}
```

would report:

```
**** (126) Missing URI, found 'match' in line 4
     Insert URI.
```

6.17 key

To quote from the W3C standard: "*Keys provide a way to work with documents that contain an implicit cross-reference structure. The ID, IDREF and IDREFS attribute types in XML provide a mechanism to allow XML documents to make their cross-reference explicit.*" [2]

Syntax

<key> ::= "`key`" <quote> <name> <quote>
"`using`" <quote> <xpath> <quote>
"`keyNodes`" <quote> <xpath> <quote>

Examples

Consider a simple list of composers:

```
<composers>
    <composer>
        <firstnames>Johann Sebastian</firstnames>
        <lastname>Bach</lastname>
    </composer>
    <composer>
        <firstnames>Leopold</firstnames>
        <lastname>Mozart</lastname>
    </composer>
    <composer>
        <firstnames>Carl Philipp Emanuel</firstnames>
        <lastname>Bach</lastname>
    </composer>
    <composer>
        <firstnames>Johann Christian</firstnames>
        <lastname>Bach</lastname>
    </composer>
    <composer>
        <firstnames>Wolfgang Amadeus</firstnames>
        <lastname>Mozart</lastname>
    </composer>
</composers>
```

Say we wished to extract all the members of a single family and list them in a table. Using a key construction we can use the following *Rexsel* script which searches all <`composer`> tags and returns a list of nodes matching '`lastname`'.

```
stylesheet {
    version "1.0"
    xmlns "xml" "http://www.w3.org/XML/1998/namespace"

    key "surname-search" using "composer" keyNodes "lastname"

    match  using "/" {
        element "html" {
            element "body" {
                element "h2" { text "Composers" }
                element "table" {
                    element "tr" {
                        element "th" { text "First Names" }
                        element "th" { text "Last Name" }
                        element "th" { text "Marks" }
                    }
```

[2] https://www.w3.org/TR/xslt-10/#element-key

```
                    foreach "key('surname-search', 'Bach')" {
                        element "tr" {
                            element "td" { value "firstnames" }
                            element "td" { value "lastname" }
                        }
                    }
                }
            }
        }
    }
}
```

The '`foreach`' XPath expression '`key('surname-search', 'Bach')`' uses the list of nodes produced by the key that matches against '`lastname`', in this case the string "`Bach`". This will produce the XML

```
<html>
   <body>
      <h2>Composers</h2>
      <table>
         <tr>
            <th>First Names</th>
            <th>Last Name</th>
            <th>Marks</th>
         </tr>
         <tr>
            <td>Johann Sebastian</td>
            <td>Bach</td>
         </tr>
         <tr>
            <td>Carl Philipp Emanuel</td>
            <td>Bach</td>
         </tr>
         <tr>
            <td>Johann Christian</td>
            <td>Bach</td>
         </tr>
      </table>
   </body>
</html>
```

Not using the key statement would require the loop XPath to be replaced with

```
foreach "//composer/lastname[. = 'Bach']/parent::*" {
    element "tr" {
        element "td" { value "firstnames" }
        element "td" { value "lastname" }
    }
}
```

which is more complex.

Errors

Leaving out the name

```
key  using "composer" keyNodes "surname"
```

gives

```
**** (106) Unexpected symbol "using" instead of "expression after 'match'" in "key" in line 7
     Check spelling, missing expression, bracket or quote?
```

Similarly forgetting either of the expressions

```
key "surname-search" using keyNodes "surname"
```

typically gives

```
**** (106) Unexpected symbol "keyNodes" instead of "expression after 'using'" in "key" in line 5
     Check spelling, missing expression, bracket or quote?
```

Not having unique key names

```
key "surname-search" using "composer" keyNodes "surname"
key "surname-search" using "composer" keyNodes "surname"
```

gives

```
*** (110) "surname-search" symbol in 5 already declared in line 6
     Remove duplicate or check spelling.
```

6.18 match

A *match* instruction allows a template that produces output that is invoked by comparing the input against an XPath expression. It is equivalent to an XSLT match template. Having a distinct difference between '`function`' and '`match`' ensures that there is never confusion and fulfils the requirement that the '`name`' and '`match`' attributes should never occur together in a `<xsl:template>`.

Syntax

```
<match>   ::=   "match"
                    "using" <quote> <xpath expression><quote>
                    ( "scope" <name> )?
                    ( "priority" <quote> <number> <quote> )?
                "{"
                    ( <parameter> )*
                    ( <block template> )+
                "}"
```

Options

using an XPath expression indicating the node and children to match.

scope the scope (mode) of this template, matching that defined in an '`apply-remplates`' statement.

priority the numeric priority this template to determine which match is processed when more than one template of the same pattern are present.

Required Options

'`using`' is always required.

Required Elements

Although there are no required elements within the match block, the only time an empty block is used is to absorb matches not done by other matches defined earlier. For example, if it is required to ensure there is no output when there are elements in the input stream that should be ignored.

Examples

For example

```
stylesheet {
    version "1.0"

    match  using "//concertList"  scope "lists" priority "6" {
      element "div" {
```

```
            attribute  "id" "concertListAndIndex"

            element "div" {
               attribute "id" "concertList"
               apply-templates scope "lists"
            }
         }
      }
   }
```

Using the command line with errors and symbols selected, this compiles to

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Produced using Rexsel compiler 1.0 Build 189 on 20/03/2024 12:29:46 -->
<!-- No Errors -->
<!--
Symbols in context "stylesheet" in line 1, found: 1 symbol
[M] //concertList::lists           in line 4
 -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="//concertList" mode="lists" priority="6">
        <xsl:element name="div">
            <xsl:attribute name="id">
                <xsl:text>concertListAndIndex</xsl:text>
            </xsl:attribute>

            <xsl:element name="div">
                <xsl:attribute name="id">
                    <xsl:text>concertList</xsl:text>
                </xsl:attribute>

                <xsl:apply-templates mode="lists"/>
            </xsl:element>
        </xsl:element>
    </xsl:template>
</xsl:stylesheet>
```

Errors

The '`using`' option is always required so

```
match scope "lists" {
   element "div" {
      attribute  "id" "concertListAndIndex"

   }
}
```

will give the error

```
**** (118) "using" required in element "match" in line 4
     Insert element.
```

Leaving out an expression for '`scope`' or other option

```
match  using âÃİ//concertListâÃ  scope  {
  element "div" {
     attribute  "id" "concertListAndIndex"

     element "span" {
        attribute "class" "code-dir"
        text "&lt;"
        apply-templates scope "inline-text"
        text "&gt;"
     }
```

```
  }
}
```

will give the error

```
**** (130) Missing using/scope/priority expression in line 4
     Insert expression.
```

6.19 message

The ***message*** statement outputs a text message to the console that is running the XSLT. It also allows termination of the stylesheet — useful when debugging.

Syntax

<message> ::= "message" ("terminate" ("yes" | "no"))?
(<quote> <xpath> <quote> | "{" (<block template>)+ "}")

Examples

For example, a case where there are three messages of different kinds.

```
stylesheet {
    version "1.0"

    match  using âĂİ/" {
       message "A simple message"
       message terminate yes {
          text "A block message that will terminate"
       }
       message terminate no "A simple message"
    }
}
```

This compiles to

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <xsl:template match="/">
        <xsl:message>
            <xsl:text>A simple message</xsl:text>
        </xsl:message>
        <xsl:message terminate="yes">
            <xsl:text>A block message that will terminate</xsl:text>
        </xsl:message>
        <xsl:message>
            <xsl:text>A simple message</xsl:text>
        </xsl:message>
    </xsl:template>
</xsl:stylesheet>
```

When this is run the console will display (dependent on the XSLT runtime application).

```
A simple message
A block message that will terminate
```

Showing that the final message is not output.

Errors

Simple text and block together.

```
stylesheet {
    version "1.0"

    match  using âĂİ/" {
```

```
            message "Oops" {
               text "A block message"
            }

         }
      }
```

will give[3]

```
**** (128) A variable/parameter cannot have default and enclosed templates in line 5
     Remove either default/select or enclosed templates.
```

[3]This is an error derived from the '`variable`', etc. syntax.

6.20 namespace-alias

The ***namespace-alias*** statement provides a means of mapping a namespace used in a stylesheet onto another, different, namespace in the output tree. It is provided in *Rexsel* for completeness but would never normally be used, for the reason below.

Syntax

<namespace-alias> ::= "namespace-alias"
"map-from" <quote> <name> <quote>
"to" <quote> <name> <quote>

Examples

It is rarely used except in the cases where a new stylesheet is generated from within another. Within *Rexsel* this is not used since it does not support "naked" elements. For example, consider the XSLT for producing another XSLT stylesheet.

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" xmlns:nxsl="nxsl.xsl">

    <xsl:param name="inputName">name</xsl:param>

    <xsl:param name="inputValue">value</xsl:param>

    <xsl:namespace-alias stylesheet-prefix="nxsl" result-prefix="xsl"/>

    <xsl:template match="/">
        <nxsl:stylesheet version="1.0">
            <nxsl:variable name="{$inputName}">
                <nxsl:value-of select="{$inputValue}"/>
            </nxsl:variable>
        </nxsl:stylesheet>
    </xsl:template>

</xsl:stylesheet>
```

This would produce a new stylesheet

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:variable name="name">
        <xsl:value-of select="value"/>
    </xsl:variable>
</xsl:stylesheet>
```

As indicated this would be impossible to directly encode into *Rexsel*. However the following would give the same effect.

```
stylesheet {
    version "1.0"

    parameter inputName "name"
    parameter inputValue "value"

    match  using "/" {
        element "xsl:stylesheet" {
            attribute "version" "1.0"
            element "xsl:variable" {
                attribute "name" { value "$inputName" }
                element "xsl:value-of" {
```

```
                    attribute "select" { value "$inputValue" }
                }
            }
        }
    }
}
```

As would the slightly more correct version.

```
stylesheet {
    version "1.0"

    parameter inputName "name"
    parameter inputValue "value"

    match  using "/" {
        element "stylesheet" namespace "http://www.w3.org/1999/XSL/Transform" {
            attribute "version" "1.0"
            element "variable" namespace "http://www.w3.org/1999/XSL/Transform" {
                attribute "name" { value "$inputName" }
                element "value-of" namespace "http://www.w3.org/1999/XSL/Transform" {
                    attribute "select" { value "$inputValue" }
                }
            }
        }
    }
}
```

In all both cases the new stylesheet would be.

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <variable name="name">
        <value-of select="blank"/>
    </variable>
</stylesheet>
```

Errors

No name for ‘`map-from`’:

```
stylesheet {
    version "1.0"

    xmlns "pxsl" "https://paloose.org.uk/stylesheets.xsl"

    namespace-alias map-from to "xsl"

    match  using "/" {
    }
}
```

will give

```
**** (106) Unexpected symbol "to" instead of "'map-from' name" in line 6
     Check spelling, missing expression, bracket or quote?
**** (119) Missing item map expression in line 6
     Insert item.
```

Similarly with neither present:

```
stylesheet {
    version "1.0"
```

```
        xmlns "pxsl" "https://paloose.org.uk/stylesheets.xsl"

        namespace-alias map-from to

        match  using "/" {
        }
    }
```

will give

```
    **** (106) Unexpected symbol "to" instead of "'map-from' name" in line 6
         Check spelling, missing expression, bracket or quote?
    **** (106) Unexpected symbol "match" instead of "'to' name" in line 6
         Check spelling, missing expression, bracket or quote?
    **** (119) Missing item map expression in line 6
         Insert item.
```

6.21 number

The ***number*** element count items sequentially and inserts a formatted number into the result tree. It is not convenient to give all the various conditions that this element allows, so for a more expansive description see the W3C document[4].

Syntax

```
<number>  ::=  "number" "{"
                   ( "level" ( "single" | "multiple" | "any" ) |
                     "count" <quote> <pattern> <quote> |
                     "from" <quote> <pattern> <quote> |
                     "value" <quote> <number expression> <quote> |
                     "letter-value" ( "alphabetic" | "traditional" )? |
                     "lang" <quote> <language identifier> <quote> |
                     "grouping-separator" <quote> <grouping character> <quote> |
                     "grouping-size" <quote> <digits> <quote> )*
               "}"
```

Elements

The range of values within an '`number`' block is identical to the XSLT definition. This is a very general list and the formal specification should be consulted for a complete description.

count	An XPath that defines what in the source tree should be numbered sequentially.
level	Defines how levels of the source tree should be considered in generating sequential numbers.
from	Specifies where the numbering should start.
value	Applies a given format to a number.
format	Defines the format of the generated number: "`1`", "`01`", etc.
lang	The language specifier.
letter-value	Provides clarification for languages that have more than one numbering system that uses letters.
grouping-separator	Specifies the character to be placed between groups of digits (eg thousands). The default is '`,`'.
grouping-size	The size of the groups of digits that make up a number. Defaults to 3 representing thousands.

[4] https://www.w3.org/TR/xslt-10/#element-number

Required Options

None

Required Elements

None

Examples

Using an example based on the W3C description, consider outputting a new list with the number of an item inserted as a Roman digit.

```
stylesheet {
    version "1.0"

    match using "items" {
       element "newList" {
          foreach "item" {
            sort using "."
            element "p" {
               number {
                   value "position()"
                   format "i. "
               }
               value "."
            }
         }
      }
   }
}
```

compiles to

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <template  match="items">
        <element  name="newList">
            <for-each select="item">
                <sort  select="."/>
                <element  name="p">
                    <number  value="position()" format="i. "/>
                    <value-of  select="."/>
                </element>
            </for-each>
        </element>
    </template>
</stylesheet>
```

Running this against the XML scrap

```
<?xml version="1.0" encoding="UTF-8"?>
<items>
    <item>First item</item>
    <item>Second item</item>
    <item>Third item</item>
    <item>Fourth item</item>
</items>
```

would give the result which would sorted alhabetically, so the item "`Fourth item`" would be placed second in the list.

```
<?xml version="1.0" encoding="UTF-8"?>
<newList>
    <p>i. First item</p>
    <p>ii. Fourth item</p>
    <p>iii. Second item</p>
    <p>iv. Third item</p>
</newList>
```

Errors

Most of the errors here are self-explanatory. For example

```
stylesheet {
    version "1.0"

    number {
       value "position()"
       formt "1. "
    }
}
```

giving

```
**** (105) Unexpected symbol "formt" found in "number" in line 6
     Check spelling, missing expression, bracket or quote?
```

6.22 output

output is a root level instruction (within the stylesheet element) that controls the characteristics of the output document. Only one declaration of this element is allowed.

Syntax

```
<output>  ::=  "output" "{"
                   ( "method" ( "xml" | "html" | "text" ) |
                     "version" <version number> |
                     "encoding" ( "UTF-8" | "UTF-16" | ··· ) |
                     "omit-xml-declaration" ( "yes" | "no" )? |
                     "indent" ( ( "yes" | "no" ) )? |
                     "doctype-system" <quote> <DTD> <quote> |
                     "doctype-public" <quote> <DTD> <quote> |
                     "cdata-section-elements" <quote> <cdata list> <quote> )*
               "}"
```

Elements

The range of values within an '`output`' block is identical to the XSLT definition.

method	Specifies the output format (currently only '`xml`', '`html`' or '`text`').
version	Specifies the value of the version attribute of the XML or HTML declaration in the output document. This attribute should only be used when method is '`xml`' or '`html`'.
encoding	Specifies the text encoding to be used: UTF-8, etc.
omit-xml-declaration.	Specifies whether the XSLT processor should output an XML declaration; permissible values are '`yes`' or '`no`'.
standalone	Specifies whether the XSLT processor should output a standalone document declaration; permissible values are '`yes`' or '`no`'.
indent	Specifies whether the XSLT processor may add additional whitespace when outputting the result tree; permissible values are '`yes`' or '`no`'.

doctype-public	Specifies the public identifier to be used in the document type declaration.
doctype-system	Specifies the system identifier to be used in the document type declaration.
cdata-section-elements	Specifies a list of the names of elements whose text node children should be output using CDATA sections.
media-type	Specifies the media type (MIME content type) of the data that results from outputting the result tree.

Required Options

None

Required Elements

None

Examples

Setting a set of output selections

```
stylesheet {
    version "1.0"

    output {
       doctype-public "-//W3C//DTD XHTML 1.0 Transitional//EN"
       doctype-system "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
       method xml
       version "1.1"
       encoding "UTF-16"
    }
}
```

compiles to

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">

    <output
       doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
       doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
       method="xml"
       version="1.1"
       encoding="UTF-16"/>

</stylesheet>
```

Errors

Most of the errors here are self-explanatory. For example

```
stylesheet {
    version "1.0"

    output {
       method txt
    }
}
```

giving

```
**** (108) Illegal value for "method", found "txt" instead of "xml", "html" or "text" in line 5
     Supply correct value.
```

6.23 parameter

A ***parameter*** can occur either at the root level with the stylesheet or as a definition of what parameters are used by a function. Parameters can be declared with a default value either as a simple string or a block.

Syntax

<parameter> ::= “parameter” <name>
((<quote> <xpath expression> <quote>) |
(“{” (<block template>)+ “}”)
)?

Examples

```
stylesheet {
    version "1.0"

   parameter param-1
   parameter param-2 "'default-value'"
   parameter param-3 {
      text "default-value"
   }
}
```

Note that *param-2* and *param-3* are notionally identical.

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet
    xmlns="http://www.w3.org/1999/XSL/Transform"
    version="1.0">

    <param name="param-1"/>
    <param name="param-2" select="'default-value'"/>
    <param name="param-3">
       <text>default-value</text>
    </param>

</stylesheet>
```

6.24 preserve-space

A ‘`preserve-space`’ statement defines the elements in the source document whose whitespace should be preserved. It is a list of whitespace separated names.

Syntax

<preserve-space> ::= “`preserve-space`” <quote> <name list> <quote>

Options

None

Elements

None

Examples

It is possible to be selective and only include certain namespaces. For example preserving spaces in namespace ‘`a`’ and ‘`dc`’.

```
stylesheet {
    version "1.0"

    xmlns "a"       "http://relaxng.org/ns/annotation/1.0"
    xmlns "dc"      "http://purl.org/dc/elements/1.1/"
    xmlns "xhtml"   "http://www.w3.org/1999/xhtml"

    preserve-space "a dc"

}
```

Note that *Rexsel* does not currently check to see whether the namespace list has valid entries for the declared namespaces.

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet
    xmlns="http://www.w3.org/1999/XSL/Transform"
    xmlns:a="http://relaxng.org/ns/annotation/1.0"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
    version="1.0">
 <preserve-space elements="a dc"/>

</stylesheet>
```

Errors

Leaving out the namespace list

```
stylesheet {
    version "1.0"

    xmlns "a"       "http://relaxng.org/ns/annotation/1.0"
    xmlns "dc"      "http://purl.org/dc/elements/1.1/"
    xmlns "xhtml"   "http://www.w3.org/1999/xhtml"

    preserve-space

     proc fn {

    }

}
```

gives the error

```
*** (106) Unexpected symbol "function" instead of "namesspace list" in "preserve-space" in line 10
     Check spelling, missing expression, bracket or quote?
```

6.25 proc

Procedures (or functions, note to be confused with `<xsl:function>` are scraps of templates that can be called from other parts of the stylesheet. The keyword takes no options other than the procedure name. Parameters are declared within the block that follows the procedure declaration.

Syntax

```
<proc>   ::=   "proc" <quote> <name> <quote>
               "{"
                   ( "parameter" <name> <quote> <xpath> <quote> )*
                   ( <block template> )+
               "}"
```

Options

None

Required Elements

Although there are no required elements within the procedure block, it is meaningless to have a procedure that is null and does not return any values (string characters).

Examples

Consider a procedure that returns a substring after the last occurrence of a character. The following would be an example of such a (recursive) procedure.

```
proc substring-after-last {
   parameter string
   parameter delimiter
   choose {
      when "contains($string, $delimiter)" {
        call substring-after-last {
           with string "substring-after($string, $delimiter)"
           with delimiter "$delimiter"
        }
     }
     otherwise {
        value "$string"
     }
   }
}
```

which translates to the rather more expansive

```
<template name="substring-after-last">
    <param name="string"/>
    <param name="delimiter"/>

    <choose>
```

```
        <when test="contains($string, $delimiter)">
            <call-template name="substring-after-last">
                <with-param name="string" select="substring-after($string, $delimiter)"/>
                <with-param name="delimiter" select="$delimiter"/>
            </call-template>
        </when>
        <otherwise>
            <value-of select="$string"/>
        </otherwise>
    </choose>
</template>
```

To use this within a variable to extract the final component in a filename and path.

```
variable idString {
    call substring-after-last {
        with string "substring-after($filename, '/')"
        with delimiter "'/'"
    }
}
```

Errors

Forgetting the proc name

```
proc {
   parameter string
   parameter delimiter

   ...
}
```

will give the error

```
**** (115) Expected name after "proc" in line 4
     Syntax: proc <qname>  <block>.
```

Leaving out the opening bracket

```
proc substring-after-last
   parameter string
   parameter delimiter

   ...
}
```

will give the error

```
**** (105) Unexpected symbol in "proc" in line 4
     Check spelling, illegal keyword, or missing quote?
```

The other main error that can occur is having the parameter declared in any other place than the start of the procedures's block statements. So for example

```
proc substring-after-last {
   parameter string

   choose {
      when "contains($string, $delimiter)" {
        call substring-after-last {
           with string "substring-after($string, $delimiter)"
           with delimiter "$delimiter"
        }
```

```
      }
      otherwise {
         value "$string"
      }
   }
   parameter delimiter
}
```

will give the error report

```
**** (131) Parameter "delimiter" in "proc:substring-after-last" in line 19 must follow declaration.
     Check order.
```

6.26 processing-instruction

The ***processing-instruction*** statement injects a processing instruction into the output document where the statement is placed.

Syntax

<processing-instruction> ::= “processing-instruction” <quote> <name> <quote>
“{”
(<block template>)+
“}”

Examples

For example, a stylesheet that iterates through a set of <loop> elements in a file and output the result together with a processing instruction that generates an index number of the elements.

```
stylesheet {
    version "1.0"

    match  using "/root" {
        element "list" {
            foreach "loop" {
                element "div" {
                    processing-instruction "loop" {
                        value "concat( '[', position(), ']' )"
                    }
                    value "concat( 'Entry [', ., ':', position(), ']' )"
                }
            }
        }
    }
}
```

This compiles to

```
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">
    <template match="/root">
        <element name="list">
            <for-each select="loop">
                <element name="div">
                    <processing-instruction name="loop">
                        <value-of select="concat('[', position(), ']')"/>
                    </processing-instruction>
                    <value-of select="concat('Entry [', ., ':', position(), ']')"/>
                </element>
            </for-each>
        </element>
    </template>
</stylesheet>
```

To see the result of the above, consider a scrap of XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
    <loop>A</loop>
    <loop>B</loop>
    <loop>C</loop>
    <loop>D</loop>
</root>
```

Running the script would produce

```
<?xml version="1.0" encoding="UTF-8"?>
<list>
    <div><?loop [1]?>Entry [A:1]</div>
    <div><?loop [2]?>Entry [B:2]</div>
    <div><?loop [3]?>Entry [C:3]</div>
    <div><?loop [4]?>Entry [D:4]</div>
</list>
```

Errors

No opening bracket:

```
element "div" {
    processing-instruction "loop"
        value "concat( '[', position(), ']' )"
    }
    value "concat( 'Entry [', ., ':', position(), ']' )"
```

will give

```
**** (106) Unexpected symbol "value" instead of "{" in "processing-instruction" in line 8
     Check spelling, missing expression, bracket or quote?
**** (139) Found unexpected reserved word "value" in "processing-instruction" in line 8
     Check spelling.
```

Leaving out the name

```
processing-instruction  {
    value "concat( '[', position(), ']' )"
}
```

will give

```
**** (115) Expected name after "processing-instruction" in line 8
     Syntax: processing-instruction <qname>  <block>.
```

6.27 sort

A *sort* instruction is used in conjunction with '`apply-templates`' and '`foreach`' to determine the order in which they should process the selected nodes.

Syntax

```
<sort>  ::=  "sort"
               (
                 ( "using" <quote> <xpath expression> <quote> )?
                 ( "ascending" | "descending" )?
                 ( "upper-first" | "lower-first" )?
                 ( "lang" <quote> <language identifier> <quote> )?
                 ( "text-sort" | "number-sort" )?
               )
```

Options

using	an XPath expression indicating the node and children to match.
ascending**/**descending	specifies the processing order.
upper-first**/**lower-first*	specifies whether upper or lower case take precedence in the ordering.
lang	specifies the language to be used for the sort.
text-sort**/**number-sort	specifies whether ordering should be done alphabetically or numerically.

* denotes the default value.

Note that the options '`text-sort`' and '`number-sort`' are to avoid conflict with the reserved word '`text`' and '`number`'.

Elements

None

Examples

For example

```
stylesheet {
   version "1.0"

    proc output-variable-index {
      foreach "//aw:definitions/aw:define" {
         sort using "@var" upper-first

      }
   }

}
```

This compiles to

```
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0">

    <template name="output-variable-index">
        <for-each select="//aw:definitions/aw:define">
            <sort select="@var" case-order="upper-first"/>
        </for-each>
    </template>

</stylesheet>
```

Errors

‘`sort`’ statements should occur directly after the item they are refering to. For example

```
stylesheet {
   version "1.0"

    proc output-variable-index {
      foreach "//aw:definitions/aw:define" {
         value "'Some text'"
         sort using "@var" upper-first
      }
   }
}
```

would give an error

```
**** (141) Sort in "foreach://aw:definitions/aw:define" in line 7 must follow declaration.
     Check order.
```

6.28 strip-space

strip-space defines the elements in the source document whose whitespace should be removed. It is a list of whitespace separated names.

Syntax

<strip-space> ::= "`strip-space`" <quote> <name list> <quote>

Options

None

Required Elements

None

Examples

It is possible to be selective and only include certain namespaces. For example stripping spaces in namespace '`a`' and '`dc`'.

```
stylesheet {
    version "1.0"

    xmlns "a"       "http://relaxng.org/ns/annotation/1.0"
    xmlns "dc"      "http://purl.org/dc/elements/1.1/"
    xmlns "xhtml"   "http://www.w3.org/1999/xhtml"

    preserve-space "a dc"

}
```

Note that *Rexsel* does not currently check to see whether the namespace list has valid entries for the declared namespaces.

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet
    xmlns="http://www.w3.org/1999/XSL/Transform"
    xmlns:a="http://relaxng.org/ns/annotation/1.0"
    xmlns:dc="http://purl.org/dc/elements/1.1/"
    xmlns:xhtml="http://www.w3.org/1999/xhtml"
    version="1.0">
 <preserve-space elements="a dc"/>

</stylesheet>
```

Errors

Leaving out the namespace list

```
stylesheet {
    version "1.0"

    xmlns "a"       "http://relaxng.org/ns/annotation/1.0"
    xmlns "dc"      "http://purl.org/dc/elements/1.1/"
    xmlns "xhtml"   "http://www.w3.org/1999/xhtml"

    preserve-space

     proc fn {

    }

}
```

gives the error

```
*** (106) Unexpected symbol "function" instead of "namesspace list" in "preserve-space" in line 10
     Check spelling, missing expression, bracket or quote?
```

6.29 stylesheet

The *stylesheet* keyword encloses the entire group of templates. There should be only one stylesheet definition for each file. Any stylesheet statements that occur after the first declaration are ignored.

Syntax

<stylesheet> ::= "`stylesheet`" "{"
"`version`" <version number>
("`id`" <name>)?
("`exclude-result-prefixes`" <name list>)?
(<name space list> (6.34))?
(<output> (6.22))?
(<import> |
<include> |
<variable> (6.32) |
<parameter> (6.23) |
<key> (6.17) |
<matcher> |
<proc>)*
"}"

Required Options

None

Required Elements

version The version number of the XSLT required by this stylesheet.

Examples

For example the skeleton of a Rexsel stylesheet is

```
stylesheet {
        version "1.0"
}
```

would give the following XSLT. The '`xmlns:xsl`' namespace is automatically added to the output.

```
<!-- Produced using Rexsel compiler 1.0 Build 197 -->
<!-- No Errors -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0"/>
```

Errors

Note that the *version* declaration is required and will generate an error if not there.

```
stylesheet {
}
```

would give the following error in the XSLT output.

```
**** (118) "version" required in element "stylesheet" in line 1
     Insert element.
```

Any statements that are invalid within a stylesheet declaration are reported as an error.

```
stylesheet {
    version "1.0"

    element "div" {
    }
}
```

would return

```
**** (139) Found unexpected reserved word "element" in "stylesheet" in line 1
     Check spelling.
```

6.30 text

A ***text*** statement sends the enclosed string directly to the output stream.

Syntax

<text> ::= "`text`" ("`disable-output-escaping`")?
<quote> <alphanumeric string> <quote>

Options

disable-output-escaping	specifies whether special characters are escaped when written to the output. For example, if present, "`<div>`" would be output as "`&lt;div&gt;`".

Elements

None

Examples

For example from a stylesheet used when processing LaTeX files (note the use of '\\' to produce '\' because a single backspace is an escape character).

```
stylesheet {
    version "1.0"

     proc nwmargintag {
        text disable-output-escaping "\\nwmargintag{"
        text "}"
    }

}
```

will compile to

```
<template name="nwmargintag">
    <text>\nwmargintag{</text>
    <text>}</text>
</template>
```

Errors

Misspelling the option

```
stylesheet {
    version "1.0"

     proc nwmargintag {
        text disable-ouput-escaping "\\nwmargintag{"
        text "}"
```

```
    }

}
```

gives the error (there may also be other consequential errors here)

```
*** (105) Unexpected symbol "disable-ouput-escaping" found in "text" in line 5
     Check spelling, missing expression, bracket or quote?
```

In the unlikely case of forgetting the text expression

```
stylesheet {
    version "1.0"

     proc nwmargintag {
        text disable-output-escaping
        text "}"
    }

}
```

which is reported as

```
**** (105) Unexpected symbol "text" found in "text" in line 6
     Check spelling, missing expression, bracket or quote?
```

6.31 value

A ***value*** statement evaluates and returns an XPath string.

Syntax

<value> ::= "`value`" ("`disable-output-escaping`")?
<quote> <xpath expression> <quote>

Options

disable-output-escaping	specifies whether special characters are escaped when written to the output. For example, if present, "`<tag>`" would be output as "`&lt;tag&gt;`".

Elements

None

Examples

```
variable var-1 {
    value "'Some text'"
}

variable var-2 {
    value "$var-1"
}
```

will result in *var-2* being set to the string "Some text".

```
<variable name="var-1">
    <value-of select="'Some text'"/>
</variable>

<variable name="var-2">
    <value-of select="$var-1"/>
</variable>
```

Note that there is no enclosing block and an error will thrown if one is given. Also remember that the above could be written as

```
constant var-1 {
    value "'Some text'"
}

constant var-2 {
    value "$var-1"
}
```

6.32 variable/constant

A *variable* can occur either at the root level with the stylesheet or within any valid block. Variables can be declared with a single value or a block.

Note that the term *variable* in the XSLT is a misnomer for those who are used to more conventional computer languages. Once a variable has been set within a particular scope it cannot be reset.

Because of the XSLT concept of variables being set only once, *Rexsel* offers an alternative keyword, '`constant`', to better reflect the actual operational type, see the example below.

Syntax

<variable> ::= ("`variable`" | "`constant`") <name>
(<quote> <xpath> <quote>) | ("{" (<block template>)+ "}")

Examples

Consider a set of variables at a global level.

```
stylesheet {
    version "1.0"

    variable var-1 "'Some text'"
    variable var-2 {
       text "Some text"
    }
    variable var-3 {
       value "$var-1"
    }
}
```

The variables *var-1*, *var-2* and *var-3* would all be set the same value "Some text". The same declarations using the alternative approach using '`constant`'.

```
stylesheet {
    version "1.0"

    constant var-1 "'Some text'"
    constant var-2 {
       text "Some text"
    }
    constant var-3 {
       value "$var-1"
    }
}
```

giving identical results. It is left to users to decide if the approach is useful. These compile to

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet
     xmlns="http://www.w3.org/1999/XSL/Transform"
     version="1.0">

   <variable  name="var-1" select="'Some text'"/>
   <variable  name="var-2">
      <text>Some text</text>
   </variable>
```

```
   <variable  name="var-3">
      <value-of  select="$var-1"/>
   </variable>

</stylesheet>
```

6.33 version

The *version* statement is a non-optional statement that defines the XSLT version to be used. It should be placed at the root level within the stylesheet statement. For example

Rexsel

```
stylesheet {
        version "1.0"
}
```

which gives

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet xmlns="http://www.w3.org/1999/XSL/Transform" version="1.0"/>
```

6.34 xmlns

The *xmlns* keyword allows definition of the various namespaces used with the templates.

Syntax

<name space list>	::=	<name space def> (<newline> <name space def>)*
<name space def>	::=	<prefix> <reference>
<prefix>	::=	<quote> <name> <quote>
<reference>	::=	<quote> <uri> <quote>

Required Options

<prefix>	The id of the namespace that is used within the stylesheet.
<reference>	The unique reference of the namespace, usually a URL reference.

Required Elements

None

Examples

Definitions are given as a list within the stylesheet definition

```
stylesheet {
    version "1.0"

    xmlns "a"       "http://relaxng.org/ns/annotation/1.0"
    xmlns "dc"      "http://purl.org/dc/elements/1.1/"
    xmlns "xhtml"   "http://www.w3.org/1999/xhtml"

}
```

The above "folds" the declared namespace into the `<stylesheet>` element.

```
<?xml version="1.0" encoding="UTF-8"?>
<stylesheet
   xmlns="http://www.w3.org/1999/XSL/Transform"
   xmlns:a="http://relaxng.org/ns/annotation/1.0"
   xmlns:dc="http://purl.org/dc/elements/1.1/"
   xmlns:xhtml="http://www.w3.org/1999/xhtml"
   version="1.0"/>
```

Errors

Missing URI Definition

The only error here (until URI checking is added) is a missing URI reference. Note that currently two definitions of the same namespace are allowed. This will be addressed.

```
stylesheet {
    version "1.0"

    xmlns "a"       "http://relaxng.org/ns/annotation/1.0"
    xmlns "dc"
    xmlns "xhtml"   "http://www.w3.org/1999/xhtml"

}
```

would report:

```
*** (106) Unexpected symbol "xmlns" instead of "Namespace pair [prefix] [ref]" in "xmlns" in line 5
     Check spelling, illegal keyword, or missing quote?
```

Chapter 7

Errors Summary

The following is a list of errors that the compiler detects. The error number is not a contiguous set with some numbers ignored, there are historical reasons for this.

In some case the line number associated with the error is not necessarily the line number of the actual error, rather the start line of the block in which the error has occurred.

Note also the line number presented in the error messages may not be accurate for the fragment shown because it is taken from a larger piece of source code.

101 — Fatal Error

An unrecoverable error of indeterminate type. Should (!) never happen.

102 — Source File Does Not Exist

File that *Rexsel* is trying to read does not exist. Check that both path and file exists (check correct file extension).

103 — Cannot Read From File

In general this is a permissions problem.

104 — Missing "{" Bracket

Mostly, the only time that this is detected is after the '`stylesheet`' keyword at the start of the file. For example, the simplistic source

```
stylesheet
```

would give the error

```
**** (104) Missing "{" in line 1
     Insert bracket.
```

105 — Found Unexpected Symbol

An unexpected symbol was found in a keyword. A general "catch all" for something the compiler does not understand. Usually caused by keyword mis-spelling, and illegal keyword for that syntax or a missing quote.

106 — Found Unexpected Symbol Instead Of

Similar to the previous error, but gives what should be in the syntax at that point. For example while entering a XML namespace at the start

```
stylesheet {

   version "1.0"

xmlns
```

would give the error (no closing bracket also)

```
**** (106) Unexpected symbol found instead of "Namespace pair [prefix] [ref]" in "xmlns" in line 5
     Check spelling, illegal keyword, or missing quote?
```

107 — Found Unexpected Expression

Found a quoted expression when expecting a keyword or bracket.

108 — Unknown Value

Produced when a particular value of option is expected. For example the output '`method`' expects the values '`xml`', '`html`' or '`text`', anything else will cause this error. For example

```
stylesheet {
    version "1.0"

   output {
      method doc
   }
}
```

would give the error

```
**** (108) Illegal value for "method", found 'doc' instead of 'xml', 'html' or 'text' in line 5
     Supply correct value.
```

109 — Already Declared In

When a symbol has already been declared. For example, only one declaration of '`version`' can be declared

```
stylesheet {
   version "1.0"
   version "1.0"
}
```

would give the error

```
**** (109) "version" in line 3 is already declared in line 2
     Remove duplicate.
```

110 — Duplicate Symbol

Similar to Error 109 but used when building the symbol table. For example

```
stylesheet {
    version "1.0"

    variable smallText "'Some text'"

    variable smallText "'Some more text'"
}
```

would give the error

```
**** (110) "smallText" symbol in 6 already declared in line 4
     Remove duplicate or check spelling.
```

115 — Expected Name

If there is a missing name in a declaration etc. For example

```
stylesheet {
    version "1.0"

     proc  {
        parameter inScript
        parameter inScriptsDir

        element "script" {
            attribute "src" {
               value "concat($inScriptsDir, $inScript)"
            }
            attribute "type"  "text/javascript"
        }
    }
}
```

would give the error

```
**** (115) Expected name after "function" in line 4
     Syntax:  proc <qname> <block>.
```

116 — Could Not Find Variable

If an XPath variable is used within a block has not been declared. For example

```
stylesheet {
    version "1.0"
```

```
    proc join {
       parameter param-1
       parameter param-2

       value "concat($param-1, $param-3)"
    }
}
```

would give the error

```
**** (116) Could not find "param-3" in line 8
     Check "param-3" is defined in current block.
```

118 — Required Element

A missing keyword that is required by the syntax. For example the source

```
stylesheet {

}
```

would give the error

```
**** (118) "version" required in element "stylesheet" in line 1
     Insert element.
```

121 — Missing Variable Value

Unlike parameters that can have no default value or (empty) block, variables (constants) must have either the value or block, both of which return a value to assign to the variable. For example

```
stylesheet {
    version "1.0"

    constant valid-param-1
}
```

would give the error

```
**** (129) A variable must have either default or enclosed templates in line 4
     Supply either default/select or enclosed templates.
```

This also detects an empty block so that

```
stylesheet {
    version "1.0"

    constant valid-param-1 {
    }

}
```

would give the error

```
**** (121) Missing value for "valid-param-1" in line 4
     Insert variable/constant value or block.
```

122 — Missing Test Expression

'`when`' and '`if`' conditions require an associated test XPath expression. So for example

```
when {
   call common.substring-after-last {
      with string "substring-after($string, $delimiter)"
      with delimiter "$delimiter"
   }
}
```

would give the error

```
**** (122) Missing test expression in line 9
     Insert test.
```

128 — Cannot Have Both Default And Block

In a variable or parameter declaration it is not valid to have a value declared as well as a block. For example the following

```
stylesheet {
    version "1.0"

        variable smallText "'Some text'" {
           text "Some other text"
        }

}
```

would give the error

```
**** (128) A variable/parameter cannot have default and enclosed templates in line 4
     Remove either default/select or enclosed templates.
```

130 — Missing Match Options

When any expressions are missing from the '`match`' options ('`using`', '`scope`' or '`priority`'. For example

```
stylesheet {
    version "1.0"

    match  using "/" scope priority "-1" {
      element "div" {
         attribute  "id" "concertListAndIndex"

      }
   }
}
```

would give the error

```
**** (130) Missing using/scope/priority expression in line 4
     Insert expression.
```

131 — Parameter Must Be First

When declared in a '`function`' or '`match`' block, parameters should be declared as the first child in that block. For example

```
 proc addHeader {
    value "concat( '!!! ', $string )"
    parameter string
}
```

would give the error

```
**** (131) Parameter "string" in "function:addHeader" in line 6 must follow declaration.
     Check order.
```

136 — Unmatched Brackets

This is a very broad error condition and reports the matching of the "{" and "}" brackets throughout the stylesheet. Currently it only works at a global level. For example while entering a basic sitemap source file

```
stylesheet {
    version "1.0"
```

would give the error

```
***** (136) Unmatched brackets in 1
      Too many open brackets?
```

As an aside if there are any keywords, brackets or expressions declared after the closing stylesheet bracket they are ignored.

139 — Found Reserved Word

If a reserved word is used where a QName is expected, for example a proc or a variable name is expected, this will cause an error.

```
stylesheet {
    version "1.0"

     proc text {

    }
}
```

would give the error

```
**** (139) Found unexpected reserved word "text" in "function" in line 4
     Check spelling.
```

140 — invalid expression

Found invalid or null string where there should be expression.

141 — sort statement must follow appropriate statement

‘`sort`’ statement must immediately follow declaration.

Chapter 8

Installing the `crexsel` Command

Installing CRexsel is a relatively simple task. The Package consists of a small set of files.

CRexsel/Package.swift

```
CRexsel ------- Sources ------- crexsel.swift
          |               +--- Resources ------- xsl2rexsel.rxsl
          |                                  +--- xsl2rexsel.xsl
          +--- Package.swift
          +--- README.md
          +--- build.sh
```

where

`crexsel.swift`	the main crexsel program.
`xsl2rexsel.rxsl`	the uncompiler stylesheet written in Rexsel (see Chapter 12).
`xsl2rexsel.xsl`	the compiled stylesheet for the uncompiler.
`Package.swift`	the Package manifest (see Chapter 10).
`README.md`	the Package readme file (not complete at time of writing this).
`build.sh`	a simple installer script can be be modified to cater for local conditions.

Running Installation Script

This script is designed to build the CRexsel package and compile the uncompiler. It should work on both MacOS and Linux systems, however it may be necessary to be run under superuser manually.

CRexsel/Package.swift

```
swift build
cp -v .build/debug/crexsel /usr/local/bin
```

```
crexsel Sources/Resources/xsl2rexsel.rxsl -e
cp -v Sources/Resources/xsl2rexsel.xsl /usr/local/bin
```

It is designed to run only on Unix based systems at present. After the Package is built the binary application is sent to the '`/usr/local/bin`' directory.

Before running the installer script ensure that command '`/usr/bin/xsltproc`' app is installed on both (it usually is on most MacOS and Linux systems).

Next the uncompiler script is built from the Rexsel script '`xsl2rexsel.rxsl`' and sent to '`xsl2rexsel.xsl`' in the same folder (it is already there but this ensures that the latest version is used). Finally this XSL stylesheet is sent to the directory '`/usr/local/bin`'.

Chapter 9

Installing Swift on Linux

On MacOS based systems Swift is always present, but Linux based systems may require Swift to be installed. The following is a typical installation process for an Ubuntu system (on a Parallels emulator running on MacOS). It has also been tested successfully on a Fedora based system, although in both cases there is no guarantee that local conditions will prove so amenable as that described below.

Installing Dependencies

Some dependencies need to be installed before Swift is downloaded. It is worth noting that some flavours of Linux do not need this stage.

CRexsel/Package.swift

```
~$ sudo apt-get install binutils \
>           git \
>           gnupg2 \
>           libc6-dev \
>           libcurl4-openssl-dev \
>           libedit2 \
>           libgcc-9-dev \
>           libpython3.8 \
>           libsqlite3-0 \
>           libstdc++-9-dev \
>           libxml2-dev \
>           libz3-dev \
>           pkg-config \
>           tzdata \
>           unzip \
>           zlib1g-dev
[sudo] password for xxxx: ********
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
...
Setting up libgcc-9-dev:arm64 (9.5.0-1ubuntu1~22.04) ...
Setting up libstdc++-9-dev:arm64 (9.5.0-1ubuntu1~22.04) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.7) ...
~$
```

The results of running may different depending on the installation. Next load the Swift from the repository at `https://www.swift.org/download/?ref=itsfoss.com`. Download the appropriate flavour of Swift as well as the signature file. After downloading the `Home/Downloads` folder should have the following.

```
~$ ls -la Downloads/
total 914168
drwxr-xr-x  3 xxxx xxxx      4096 May  9 16:29 .
drwxr-x--- 15 xxxx xxxx      4096 May  2 16:49 ..
-rw-rw-r--  1 xxxx xxxx 592835466 May  2 16:57 swift-5.10-RELEASE-ubuntu22.04-aarch64.tar.gz
-rw-rw-r--  1 xxxx xxxx       819 May  9 16:29 swift-5.10-RELEASE-ubuntu22.04-aarch64.tar.gz.sig
~$
```

If this is the first time downloading Swift packages, import the PGP keys into the keyring. Go to the `Home/Downloads` folder and run the following.

```
~/Downloads$ wget -q -O - https://swift.org/keys/all-keys.asc — gpg --import -
gpg: directory '/home/hsfr/.gnupg' created
gpg: keybox '/home/hsfr/.gnupg/pubring.kbx' created
gpg: /home/hsfr/.gnupg/trustdb.gpg: trustdb created
gpg: key D441....B37AD: public key "Swift Automatic Signing Key #1 <swift-infrastructure@swift.org>" imported
gpg: key 9F59....A56D5F: public key "Swift 2.2 Release Signing Key <swift-infrastructure@swift.org>" imported
...
gpg: key 925C....3D1561: "Swift 5.x Release Signing Key <swift-infrastructure@swift.org>" 1 new signature
gpg: key F167....9CE069: "Swift Automatic Signing Key #4 <swift-infrastructure@forums.swift.org>" 1 new signature
gpg: Total number processed: 11
gpg:               imported: 8
gpg:         new signatures: 3
~/Downloads$
```

Refresh the keys to download new key revocation certificates, if any are available.

```
~/Downloads$ gpg --keyserver hkp://keyserver.ubuntu.com --refresh-keys Swift
gpg: refreshing 8 keys from hkp://keyserver.ubuntu.com
gpg: key F167....9CE069: "Swift Automatic Signing Key #4 <swift-infrastructure@forums.swift.org>" not changed
gpg: key FAF6....C16FEA: "Swift Automatic Signing Key #3 <swift-infrastructure@swift.org>" not changed
gpg: key 925C....3D1561: "Swift 5.x Release Signing Key <swift-infrastructure@swift.org>" not changed
gpg: key 7638....B2B08C4: "Swift Automatic Signing Key #2 <swift-infrastructure@swift.org>" not changed
gpg: key EF54....1E1B235: "Swift 4.x Release Signing Key <swift-infrastructure@swift.org>" not changed
gpg: key 63BC....1D306C6: "Swift 3.x Release Signing Key <swift-infrastructure@swift.org>" not changed
gpg: key 9F59....1A56D5F: "Swift 2.2 Release Signing Key <swift-infrastructure@swift.org>" not changed
gpg: key D441....12B37AD: "Swift Automatic Signing Key #1 <swift-infrastructure@swift.org>" not changed
gpg: Total number processed: 8
gpg:              unchanged: 8
~/Downloads$
```

Now extract the tar file.

```
~/Downloads$ tar xvf swift-5.10-RELEASE-ubuntu22.04-aarch64.tar.gz
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/bin/
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/bin/clang
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/bin/clang++
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/bin/clang-15
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/bin/clang-cache
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/bin/clang-cl
...
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/share/swift/
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/share/swift/compatibility-symbols
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/share/swift/diagnostics/
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/share/swift/diagnostics/en.db
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/share/swift/diagnostics/en.strings
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/share/swift/features.json
swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/share/swift/LICENSE.txt
~/Downloads$
```

Once the files have been extracted, set up the path to the binaries. There is a `usr/bin` directory in the extracted directory. First of all move this to a new location in the /usr/local/swift directory.

```
~/Downloads$ sudo mkdir /usr/local/swift
[sudo] password for hsfr: ********
~/Downloads$ sudo mv swift-5.10-RELEASE-ubuntu22.04-aarch64/usr/ /usr/local/swift
~/Downloads$
```

Add this to the PATH environment

```
~/Downloads$ echo 'export PATH=/usr/local/swift/usr/bin:$PATH' >> ~/.bashrc
~/Downloads$
```

At this point the Swift environment should be loaded so test using

```
~/Documents/SoftwareDevelopment/Public_Packages/CRexsel$ swift --version
Swift version 5.10 (swift-5.10-RELEASE)
Target: aarch64-unknown-linux-gnu
~/Documents/SoftwareDevelopment/Public_Packages/CRexsel$ swift

Welcome to Swift!

Subcommands:

  swift build      Build Swift packages
  swift package    Create and work on packages
  swift run        Run a program from a package
  swift test       Run package tests
  swift repl       Experiment with Swift code interactively

  Use `swift --version` for Swift version information.

  Use `swift --help` for descriptions of available options and flags.

  Use `swift help <subcommand>` for more information about a subcommand.

~/Documents/SoftwareDevelopment/Public_Packages/CRexsel$
```

Before the uncompile option of CRexsel can be run the xsltproc command must be installed (on some Linux distrobutions it may already be present).

```
~/Documents/SoftwareDevelopment/Public_Packages/CRexsel$ sudo apt-get install xsltproc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  xsltproc
0 upgraded, 1 newly installed, 0 to remove and 3 not upgraded.
Need to get 14.8 kB of archives.
After this operation, 159 kB of additional disk space will be used.

...

Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
~/Documents/SoftwareDevelopment/Public_Packages/CRexsel$
```

This should be sufficient to install and run CRexsel (see Chapter 9).

Chapter 10

Interfacing to the Rexsel Kernel

Accessing the *RexselKernel* compiler is very simple. The best way on both Linux and MacOS is using a Swift Package. The example below is taken from the `crexsel` command line application. The *RexselKernel* is included as a dependency from the BitBucket repository. If you are going to build a command line app then the *ArgumentParser* Package would also need to be added.

CRexsel/Package.swift

```
let package = Package(
    name: "crexsel",
    products: [
        .executable(name: "crexsel", targets: ["crexsel"]),
    ],
    dependencies: [
        .package(url: "https://bitbucket.org/hsfr/rexselkernel.git", from: "1.0.34"),
        .package(url: "https://github.com/apple/swift-argument-parser", from: "1.0.0"),
    ],
    targets: [
        .executableTarget(
            name: "crexsel",
            dependencies: [
                .product( name: "RexselKernel", package: "rexselkernel" ),
                .product( name: "ArgumentParser", package: "swift-argument-parser" ),
            ],
            resources: [
                .process("Resources")
            ]
        )
    ]
)
```

The outline of the application is very simple, first the various Packages need to be installed.

CRexsel/Sources/crexsel.swift

```
import Foundation

import RexselKernel
import ArgumentParser

@main
struct CRexsel: ParsableCommand {
```

```
    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    // -*-*-*-*-*-*-*-*-* COMMAND ENTRY POINT *-*-*-*-*-*-*-*-*
    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

    public func run() {
         // Define the command arguments using ArgumentParser
         // Set up the various flags etc from the command arguments
         // Run command
    }
}
```

The actual interface to the compiler is relatively simple.

```
// Declare an instance of the compiler.
let theCompiler = RexselKernel.sharedInstance

// Read the contents of a file (set up in "inputFile") into the compiler
theCompiler.source.readIntoCompilerStringFromFile( inputFile )

// Run the compiler with the various flags. Any left out will revert to a default
// (usually silent) value.
let runResults = theCompiler.run( showUndefined: showUndefinedErrors,
                                  lineNumbers: showLineNumbers,
                                  defaultNameSpace: useDefaultXSLNamespace,
                                  verbose: showFullMessages,
                                  debugOn: showDebugMessages )

// Extract the compiled XSL, together with the symbol table and error report.
let xslString = runResults.codeListing
let symbolListingString = runResults.symbolTable
let errors = thisCompiler.rexselErrorList
```

To see how the arguments are specified and utilised, and how the Uncompiler is invoked, look at the *crexsel* package. Note that the uncompiler relies on the command '`/usr/bin/xsltproc`' which must be present on both the Linux and MacOS systems.

Chapter 11

Extended Example

The following is an extended example badsed on the templates used to produce code from within the AntWeave[1] literate programming processor.

```
stylesheet {
    version "1.0"

    xmlns "aw" "http://www.hsfr.org.uk/Schema/AntWeave"

    output {
       method text
       version "1.0"
       omit-xml-declaration yes
       encoding "UTF-8"
    }

    parameter rootChunk
    parameter startCommentString "'//'"
    parameter endCommentString "''"

//-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
//-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

    match using "/" {
        // Find the first code chunk to process (and all others with same name)
        foreach "//aw:chunk[(@name = '*' or @name = $rootChunk) and @type = 'code']" {
            variable rootName {
                choose {
                    when "string-length($rootChunk) = 0" {
                        value "'*'"
                    }
                    otherwise {
                        value "$rootChunk"
                    }
                }
             }

            variable firstChunkPosition "position()"

            call process-code {
                with isFirstChunk "$firstChunkPosition = 1"
                with chunkNode "."
                with name "$rootName"
            }
        }
    }

//-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
```

[1] https://www.antweave.hsfr.org.uk/home.html

```
//-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

    proc process-code {
        parameter isFirstChunk
        parameter chunkNode
        parameter name

        foreach "$chunkNode" {
            if "not($isFirstChunk)" {
                value "concat( $startCommentString , ' File:' )"
                value "concat( @file, ' Line:' )"
                value "concat( @line, ' chunk:' )"
                value "concat( $name, $endCommentString, '&#x0A;' )"
            }

            // For each of the code lines in this chunk -->
            foreach "aw:code" {
                // For each of the tokens and separator elements in this line
                foreach "*" {
                    choose {
                            when "name() = 'aw:include'" {
                            // An include statement which defines a reference to
                            // another chunk which must be inserted here
                            variable chunkRef "@ref"
                            // Recurse until all included lines are output
                            call process-code {
                                // A set of nodes that are the named chunks
                                with chunkNode "//aw:chunk[@name = $chunkRef]"
                                with name "$chunkRef"
                            }
                        }
                        otherwise {
                            // Ordinary text so just output it
                            value disable-output-escaping "."
                        }
                    }
                }
                text "&#x0A;"
            }
        }
    }

//-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
//-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

     match using "node() | @*" priority "-1" { }

}
```

Compiling this with the command

```
hsfr@bramley ~ % crexsel tangle.rxsl --errors --symbols --verbose
hsfr@bramley ~ %
```

produces the following output in the file `tangle.xsl` (the output below has been compacted to remove excess lines and also indented, and the version nuimber may be different to the latest version.) :

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Produced using Rexsel compiler 1.0 Build 172 on 09/03/2024 12:14:25 -->
<!-- No Errors
-->
<!--
Symbols in context 'stylesheet', found: 6
[F] process-code         in line 47  used in line(s) 36, 69
[M] /::                  in line 20
[M] node() | @*::        in line 89
[P] endCommentString     in line 15  used in line(s) 57
[P] rootChunk            in line 13  used in line(s) 22, 25, 29
```

```
[P] startCommentString   in line 14  used in line(s) 54

Symbols in context 'forEach://aw:chunk[(@name = '*' or @name = $rootChunk) and @type = 'code']', found: 2
[V] firstChunkPosition   in line 34  used in line(s) 37
[V] rootName             in line 23  used in line(s) 39

Symbols in context 'process-code', found: 3
[V] chunkNode      in line 49  used in line(s) 52
[V] isFirstChunk   in line 48  used in line(s) 53
[V] name           in line 50  used in line(s) 57

Symbols in context 'when:name() = 'aw:include'', found: 1
[V] chunkRef   in line 67  used in line(s) 71, 72
 -->
<xsl:stylesheet
         xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
         xmlns:aw="http://www.hsfr.org.uk/Schema/AntWeave"
         version="1.0">
    <xsl:output method="text" version="1.0" omit-xml-declaration="yes" encoding="UTF-8"/>
    <xsl:param name="rootChunk"/>
    <xsl:param name="startCommentString" select="'//'"/>
    <xsl:param name="endCommentString" select="''"/>

    <xsl:template match="/">
        <xsl:for-each select="//aw:chunk[(@name = '*' or @name = $rootChunk) and @type = 'code']">
            <xsl:variable name="rootName">
                <xsl:choose>
                    <xsl:when test="string-length($rootChunk) = 0">
                        <xsl:value-of select="'*'"/>
                    </xsl:when>
                    <xsl:otherwise>
                        <xsl:value-of select="$rootChunk"/>
                    </xsl:otherwise>
                </xsl:choose>
            </xsl:variable>
            <xsl:variable name="firstChunkPosition" select="position()"/>
            <xsl:call-template name="process-code">
                <xsl:with-param name="isFirstChunk" select="$firstChunkPosition = 1"/>
                <xsl:with-param name="chunkNode" select="."/>
                <xsl:with-param name="name" select="$rootName"/>
            </xsl:call-template>
        </xsl:for-each>
    </xsl:template>

    <xsl:template name="process-code">
        <xsl:param name="isFirstChunk"/>
        <xsl:param name="chunkNode"/>
        <xsl:param name="name"/>

        <xsl:for-each select="$chunkNode">
            <xsl:if test="not($isFirstChunk)">
                <xsl:value-of select="concat($startCommentString, ' File:')"/>
                <xsl:value-of select="concat(@file, ' Line:')"/>
                <xsl:value-of select="concat(@line, ' chunk:')"/>
                <xsl:value-of select="concat($name, $endCommentString, '&#x0A;')"/>
            </xsl:if>
            <xsl:for-each select="aw:code">
                <xsl:for-each select="*">
                    <xsl:choose>
                        <xsl:when test="name() = 'aw:include'">
                            <xsl:variable name="chunkRef" select="@ref"/>
                            <xsl:call-template name="process-code">
                                <xsl:with-param name="chunkNode" select="//aw:chunk[@name = $chunkRef]"/>
                                <xsl:with-param name="name" select="$chunkRef"/>
                            </xsl:call-template>
                        </xsl:when>
                        <xsl:otherwise>
                            <xsl:value-of select="." disable-output-escaping="yes"/>
                        </xsl:otherwise>
                    </xsl:choose>
                </xsl:for-each>
```

```
                <xsl:text>&#x0A;</xsl:text>
            </xsl:for-each>
        </xsl:for-each>
    </xsl:template>

    <xsl:template match="node() | @*" priority="-1"/>

</xsl:stylesheet>
```

Chapter 12

“Uncompiler” Example

In order to translate existing XSLT files into *Rexsel* the following script (in *Rexsel*) is used, and provides another extended example of the language.

```
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
//
// XSL to Rexsel Translator
//
// This is a simple translator which takes syntactically correct XSLT
// and outputs a valid Rexsel stylesheet. No error detection on the
// inputted XSLT is carried out. Once the Rexsel is produced the Rexsel
// compiler should detect any errors in the original XSLT. However it
// may be necessary to update/delete some namespace definitions. Also
// XML-based comments are ignored.
//
// This is the original Rexsel file to produce the uncompiler and should
// be used if any modifications are to be done.
//
// Author:
//    Name  : Hugh Field-Richards
//    Email : hsfr@hsfr.org.uk
//
// Copyright 2024 Hugh Field-Richards.
//
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
//
// LICENSE
//
// Rexsel is free software: you can redistribute it and/or modify
// it under the terms of the GNU General Public License as published by
// the Free Software Foundation, either version 3 of the License, or
// (at your option) any later version.
//
// This program is distributed in the hope that it will be useful,
// but WITHOUT ANY WARRANTY; without even the implied warranty of
// MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
// GNU General Public License for more details.
//
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

stylesheet {
    version "1.0"
    id "xsl-rexsel"

    output {
        method text
```

```
    }

    xmlns "rng"   "http://relaxng.org/ns/structure/1.0"
    xmlns "a"     "http://relaxng.org/ns/annotation/1.0"
    xmlns "dc"    "http://purl.org/dc/elements/1.1/"
    xmlns "xhtml" "http://www.w3.org/1999/xhtml"
    xmlns "sch"   "http://www.ascc.net/xml/schematron"

    constant versionNumber "'1.0.1'"

    constant doubleQuote "'&#34;'"
    constant space "' '"
    constant ampersand "'&#38;'"

    constant openCurlyBracket "'&#123;'"
    constant closeCurlyBracket "'&#125;'"

    constant solidus "'&#47;'"
    constant revSolidus "'&#92;'"

    constant lessThan "'&#60;'"
    constant greaterThan "'&#62;'"

    constant identSpaces "'    '"
    constant return { text "
" }

    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-* PROCESS FILE -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

    match using "/" {
        value "concat( '// Uncompiler Version ', $versionNumber, $return )"
        text "stylesheet {"
        value "$return"
        apply-templates using "//xsl:stylesheet/@*" scope "prefix" {
            with spaces "$identSpaces"
        }
        call outputNamespaces
        apply-templates using "//xsl:stylesheet/*" {
            with spaces "$identSpaces"
        }
        text "}"
        value "$return"
   }

    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

    proc outputNamespaces {
        parameter spaces
        variable newSpaces "concat( $spaces, $identSpaces )"

        foreach "//xsl:stylesheet/namespace::*" {
            variable namespaceName "name(.)"
            variable namespaceValue "."

            value "concat( $newSpaces, 'xmlns ', $doubleQuote, $namespaceName, $doubleQuote )"
            value "concat( $space, $doubleQuote, $namespaceValue, $doubleQuote, $return )"
        }
    }

    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

    match using "@version" scope "prefix" {
        parameter spaces

        value "concat( $spaces, 'version ', $doubleQuote, ., $doubleQuote, $return )"
    }
```

```
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "@id" scope "prefix" {
    parameter spaces

    value "concat( $spaces, 'id ', $doubleQuote, ., $doubleQuote, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "@lang" scope "prefix" {
    parameter spaces

    value "concat( $spaces, 'lang ', $doubleQuote, ., $doubleQuote, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

proc  replaceStrings {
    parameter txt

    choose {
        when "contains( $txt, $revSolidus )" {
            value "substring-before( $txt, $revSolidus )"
            value "concat( $revSolidus, $revSolidus )"
            call replaceStrings {
                with txt "substring-after( $txt, $revSolidus )"
            }
        }
        when "contains( $txt, $doubleQuote )" {
            value "substring-before( $txt, $doubleQuote )"
            value "concat( $revSolidus, $doubleQuote )"
            call replaceStrings {
                with txt "substring-after( $txt, $doubleQuote )"
            }
        }
        when "contains( $txt, $lessThan )" {
            value "substring-before( $txt, $lessThan )"
            value "'&lt;'"
            call replaceStrings {
                with txt "substring-after( $txt, $lessThan )"
            }
        }
        when "contains( $txt, $greaterThan )" {
            value "substring-before( $txt, $greaterThan )"
            value "'&gt;'"
            call replaceStrings {
                with txt "substring-after( $txt, $greaterThan )"
            }
        }
        when "contains( $txt, $ampersand )" {
            value "substring-before( $txt, $ampersand )"
            value "'&#38;'"
            call replaceStrings {
                with txt "substring-after( $txt, $ampersand )"
            }
        }
        otherwise {
            value "$txt"
        }
    }
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

proc pvwOutput {
    parameter inName
    parameter inValue
    parameter inKeyword
```

```
    parameter spaces

    variable conditionedValue {
       call replaceStrings {
          with txt "$inValue"
       }
    }

    variable newSpaces "concat( $spaces, $identSpaces )"
    variable plainText "text()"
    variable conditionedPlainText {
       call replaceStrings {
          with txt "$plainText"
       }
    }

    choose {
        when "$inValue" {
            // For example
            //   <with-param name="<name>" select="<value>"/> ===> with <name> "<value>"
            value "concat( $spaces, $inKeyword, $space, $inName, $space,
                           $doubleQuote, $conditionedValue, $doubleQuote, $return )"
        }
        when "not( $inValue ) and not( . )" {
            // For example
            //   <with-param name="<name>"/> ===> with <name>
            value "concat( $spaces, $inKeyword, $space, $inName, $space,
                           $doubleQuote, $doubleQuote, $return )"
        }
        otherwise {
            // <with-param name="<name>"> text </with-param> ===> with <name> { ... }
            // However the contents (first element might be plain text).
            value "concat( $spaces, $inKeyword, $space, $inName )"
            choose {
                when "not( *[1] )" {
                    if "count( $plainText ) > 0" {
                        value "concat( $space, $openCurlyBracket, $return )"
                        value "concat( $spaces, 'text ', $doubleQuote, $conditionedPlainText, $doubleQuote )"
                        value "concat( $space, $closeCurlyBracket )"
                    }
                }
                otherwise {
                    // <with-param name="<name>"> elements </with-param> ===> with <name> { ... }
                    value "concat( $space, $openCurlyBracket, $return )"
                    apply-templates using "./*" {
                        with spaces "$newSpaces"
                    }
                    value "concat( $space, $closeCurlyBracket )"
                }
            }
            value "$return"
        }
    }
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-* XSLT ELEMENTS *-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:apply-imports" {
    parameter spaces

    value "concat( $spaces, 'apply-imports', $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:apply-templates" {
```

```
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"

    value "concat( $spaces, ’apply-templates ’ )"
    if "@select" {
        value "concat( ’ using ’, $doubleQuote, @select, $doubleQuote )"
    }
    if "@mode" {
        value "concat( ’ scope ’, $doubleQuote, @mode, $doubleQuote )"
    }
    if "./*" {
        value "concat( $space, $openCurlyBracket, $return )"
        apply-templates using "./*" {
            with spaces "$newSpaces"
        }
        value "concat( $spaces, $closeCurlyBracket )"
    }
    value "$return"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:attribute" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"
    variable plainText "text()"

    value "concat( $spaces, ’attribute ’, $doubleQuote, @name, $doubleQuote )"
    if "@namespace" {
        value "concat( ’ namespace’, $doubleQuote, @namespace, $doubleQuote )"
    }
    choose {
        when "not( *[1] )" {
            if "count( $plainText ) > 0" {
                value "concat( $space, $openCurlyBracket, $return )"
                value "concat( $newSpaces, ’text ’, $doubleQuote, $plainText, $doubleQuote, $return )"
                value "concat(  $spaces, $closeCurlyBracket, $return )"
            }
        }
        otherwise {
            value "concat( $space, $openCurlyBracket, $return )"
            apply-templates using "./*" {
                with spaces "$newSpaces"
            }
            value "concat( $spaces, $closeCurlyBracket )"
        }
    }
    value "$return"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:attribute-set" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"
    variable attributeName "@name"
    variable useAttributeSets "@use-attribute-sets"
    variable keyword "’attribute-set’"

    value "concat( $spaces, $keyword, $space, $doubleQuote, $attributeName, $doubleQuote )"
    value "concat( $space, $openCurlyBracket, $return )"
    apply-templates using "./*" {
        with spaces "$newSpaces"
    }
    value "concat( $spaces,’}’ )"
    value "$return"
```

```
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:call-template" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"
    variable nameValue "@name"

    choose {
        when "./*" {
            value "concat( $spaces, 'call ', $nameValue, $space, $openCurlyBracket, $return )"
            apply-templates using "./*" {
                with spaces "$newSpaces"
            }
            value "concat( $spaces, $closeCurlyBracket, $return )"
        }
        otherwise {
            value "concat( $spaces, 'call ', $nameValue, $return )"
        }
    }
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:choose" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"

    value "concat( $spaces, 'choose {', $return )"
    apply-templates using "./*" {
        with spaces "$newSpaces"
    }
    value "concat( $spaces, $closeCurlyBracket, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:comment" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"
    variable keyword "'comment'"

    value "concat( $spaces, 'comment {', $return )"
        apply-templates using "./*" {
            with spaces "$newSpaces"
        }
        value "concat( $spaces, $closeCurlyBracket, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:copy" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"
    variable useAttributeSets "@use-attribute-sets"

    choose {
        when "$useAttributeSets" {
            value "concat( $spaces, 'copy use-attribute-sets', $useAttributeSets, $space,
                           $openCurlyBracket, $return )"
        }
        otherwise {
```

```
            value "concat( $spaces, 'copy {', $return )"
        }
    }
    apply-templates using "./*" {
        with spaces "$newSpaces"
    }
    value "concat( $spaces, $closeCurlyBracket, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:copy-of" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"
    variable selectValue "@select"

    value "concat( $spaces, 'copy ', $doubleQuote, $selectValue, $doubleQuote, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:decimal-format" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"

    value "concat( $return, $spaces, 'decimal-format ', $openCurlyBracket, $return )"
    if "@name" {
        value "concat( $newSpaces, 'name ', $doubleQuote, @name, $doubleQuote, $return )"
    }
    if "@decimal-separator" {
        value "concat( $newSpaces, 'decimal-separator ',
                       $doubleQuote, @decimal-separator, $doubleQuote, $return )"
    }
    if "@grouping-separator" {
        value "concat( $newSpaces, 'grouping-separator ',
                       $doubleQuote, @grouping-separator, $doubleQuote, $return )"
    }
    if "@infinity" {
        value "concat( $newSpaces, 'infinity ',
                       $doubleQuote, @infinity, $doubleQuote, $return )"
    }
    if "@minus-sign" {
        value "concat( $newSpaces, 'minus-sign ',
                       $doubleQuote, @minus-sign, $doubleQuote, $return )"
    }
    if "@NaN" {
        value "concat( $newSpaces, 'NaN ',
                       $doubleQuote, @NaN, $doubleQuote, $return )"
    }
    if "@percent" {
        value "concat( $newSpaces, 'percent ',
                       $doubleQuote, @percent, $doubleQuote, $return )"
    }
    if "@per-mille" {
        value "concat( $newSpaces, 'per-mille ',
                       $doubleQuote, @per-mille, $doubleQuote, $return )"
    }
    if "@zero-digit" {
        value "concat( $newSpaces, 'zero-digit ',
                       $doubleQuote, @zero-digit, $doubleQuote, $return )"
    }
    if "@digit" {
        value "concat( $newSpaces, 'digit ',
                       $doubleQuote, @digit, $doubleQuote, $return )"
    }
    if "@pattern-separator" {
        value "concat( $newSpaces, 'pattern-separator ',
```

```
                       $doubleQuote, @pattern-separator, $doubleQuote, $return )"
     }
     value "concat( $return, $closeCurlyBracket, $return )"

}

 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

 match using "xsl:element" {
     parameter spaces

     variable newSpaces "concat( $spaces, $identSpaces )"
     variable nameValue "@name"

     value "concat( $spaces, 'element ', $doubleQuote, $nameValue,
                    $doubleQuote, $space, $openCurlyBracket, $return )"
     apply-templates using "./*" {
         with spaces "$newSpaces"
     }
     value "concat( $spaces, $closeCurlyBracket, $return )"
 }

 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

 match using "xsl:for-each" {
     parameter spaces

     variable newSpaces "concat( $spaces, $identSpaces )"
     variable selectValue "@select"

     value "concat( $spaces, 'foreach ', $doubleQuote, $selectValue,
                    $doubleQuote, $space, $openCurlyBracket, $return )"
     apply-templates using "./*" {
         with spaces "$newSpaces"
     }
     value "concat( $spaces, $closeCurlyBracket, $return )"
 }

 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

 match using "xsl:if" {
     parameter spaces

     variable newSpaces "concat( $spaces, $identSpaces )"
     variable testValue "@test"
     variable conditionedValue {
        call replaceStrings {
           with txt "$testValue"
        }
     }

     value "concat( $spaces, 'if ', $doubleQuote, $conditionedValue,
                    $doubleQuote, $space, $openCurlyBracket, $return )"
     apply-templates using "./*" {
         with spaces "$newSpaces"
     }
     value "concat( $spaces, $closeCurlyBracket, $return )"
 }

 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

 match using "xsl:include" {
     parameter spaces

     value "concat( $spaces, 'include ', $doubleQuote, @href, $doubleQuote, $return )"
 }
```

```
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

 match using "xsl:import" {
     parameter spaces

     value "concat( $spaces, 'import ', $doubleQuote, @href, $doubleQuote, $return )"
 }

 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

 match using "xsl:key" {
     parameter spaces

     variable newSpaces "concat( $spaces, $identSpaces )"

     value "concat( $return, $spaces, 'key' )"
     value "concat( ' name ', $doubleQuote, @name, $doubleQuote )"
     value "concat( ' using ', $doubleQuote, @match, $doubleQuote )"
     value "concat( ' keyNodes ', $doubleQuote, @use, $doubleQuote, $return )"
}

 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

 match using "xsl:message" {
     parameter spaces

     variable newSpaces "concat( $spaces, $identSpaces )"

     value "concat( $spaces, 'message' )"
     if "@terminate" {
         value "concat( ' terminate', $doubleQuote, @terminate, $doubleQuote )"
     }
     value "concat( $space, $openCurlyBracket, $return )"
     apply-templates using "./*" {
         with spaces "$newSpaces"
     }
     value "concat( $spaces, $closeCurlyBracket, $return )"
 }

 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

 match using "xsl:namespace-alias" {
     parameter spaces

     variable newSpaces "concat( $spaces, $identSpaces )"

     value "concat( $return, $spaces, 'namespace-alias' )"
     value "concat( ' map-from ', $doubleQuote, @stylesheet-prefix, $doubleQuote )"
     value "concat( ' to ', $doubleQuote, @result-prefix, $doubleQuote, $return )"
}

 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
 // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

 match using "xsl:number" {
     parameter spaces

     variable newSpaces "concat( $spaces, $identSpaces )"

     value "concat( $return, $spaces, 'number ', $openCurlyBracket, $return )"
     if "@count" {
         value "concat( $newSpaces, 'count ', $doubleQuote, @count, $doubleQuote, $return )"
     }
     if "@level" {
         value "concat( $newSpaces, 'level ', @level, $return )"
     }
     if "@from" {
```

```
        value "concat( $newSpaces, 'from ', $doubleQuote, @from, $doubleQuote, $return )"
    }
    if "@value" {
        value "concat( $newSpaces, 'value ', $doubleQuote, @value, $doubleQuote, $return )"
    }
    if "@format" {
        value "concat( $newSpaces, 'format ', $doubleQuote, @format, $doubleQuote, $return )"
    }
    if "@lang" {
        value "concat( $newSpaces, 'lang ', $doubleQuote, @lang, $doubleQuote, $return )"
    }
    if "@letter-value" {
        value "concat( $newSpaces, 'letter-value ', @letter-value, $return )"
    }
    if "@grouping-separator" {
        value "concat( $newSpaces, 'grouping-separator ',
                       $doubleQuote, @grouping-separator, $doubleQuote, $return )"
    }
    if "@grouping-size" {
        value "concat( $newSpaces, 'grouping-size ',
                       $doubleQuote, @grouping-size, $doubleQuote, $return )"
    }
    value "concat( $return, $closeCurlyBracket, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:otherwise" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"

    value "concat( $spaces, 'otherwise {', $return )"
    apply-templates using "./*" {
        with spaces "$newSpaces"
    }
    value "concat( $spaces, $closeCurlyBracket, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:output" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"

    value "concat( $return, $spaces, 'output ', $openCurlyBracket, $return )"
    if "@method" {
        value "concat( $newSpaces, 'method ', @method, $return )"
    }
    if "@version" {
        value "concat( $newSpaces, 'version ',
                       $doubleQuote, @version, $doubleQuote, $return )"
    }
    if "@encoding" {
        value "concat( $newSpaces, 'encoding ',
                       $doubleQuote, @encoding, $doubleQuote, $return )"
    }
    if "@omit-xml-declaration" {
        value "concat( $newSpaces, 'omit-xml-declaration ', @omit-xml-declaration, $return )"
    }
    if "@standalone" {
        value "concat( $newSpaces, 'standalone ', @standalone, $return )"
    }
    if "@doctype-public" {
        value "concat( $newSpaces, 'doctype-public ',
                       $doubleQuote, @doctype-public, $doubleQuote, $return )"
    }
    if "@doctype-system" {
```

```
        value "concat( $newSpaces, 'doctype-system ',
                       $doubleQuote, @doctype-system, $doubleQuote, $return )"
    }
    if "@cdata-section-elements" {
        value "concat( $newSpaces, 'cdata-section-elements ',
                       $doubleQuote, @cdata-section-elements, $doubleQuote, $return )"
    }
    if "@indent" {
        value "concat( $newSpaces, 'indent ', @indent, $return )"
    }
    if "@media-type" {
        value "concat( $newSpaces, 'media-type ',
                       $doubleQuote, @media-type, $doubleQuote, $return )"
    }
    value "concat( $spaces, $closeCurlyBracket, $return)"

}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:param"  {
    parameter spaces

    call pvwOutput {
        with inName "@name"
        with inValue "@select"
        with inKeyword "'parameter'"
        with spaces "$spaces"
    }
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:preserve-space" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"

    value "concat( $spaces, 'preserve-space ',
                   $doubleQuote, @elements, $doubleQuote, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:processing-instruction" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"
    variable nameValue "@name"

    value "concat( $spaces, 'processing-instruction ',
                   $doubleQuote, $nameValue, $doubleQuote, $space, $openCurlyBracket, $return )"
    apply-templates using "./*" {
        with spaces "$newSpaces"
    }
    value "concat( $spaces, $closeCurlyBracket, $return )"
 }

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:sort" {
    parameter spaces

    value "concat( $spaces, 'sort ' )"
    if "@select" {
        value "concat( ' using ', $doubleQuote, @select, $doubleQuote )"
    }
```

```
    if "@order" {
        value "concat( $space, @order )"
    }
    if "@case-order" {
        value "concat( $space, @case-order )"
    }
    if "@lang" {
        value "concat( ' lang ', $doubleQuote, @lang, $doubleQuote )"
    }
    if "@data-type" {
        // Avoids conflict with reserved words.
        value "concat( $space, @data-type, '-sort' )"
    }
    value "$return"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:strip-space" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"

    value "concat( $spaces, 'strip-space ',
                   $doubleQuote, @elements, $doubleQuote, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:template" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"

    choose {
        when "@name" {
            value "concat( $return, $spaces, 'proc ', @name, $space,
                           $openCurlyBracket, $return )"
            apply-templates using "./*" {
                with spaces "$newSpaces"
            }
            value "concat( $spaces, $closeCurlyBracket, $return )"
        }
        otherwise {
            value "concat( $return, $spaces, 'match ' )"
            if "@match" {
                value "concat( ' using ', $doubleQuote, @match, $doubleQuote )"
            }
            if "@mode" {
                value "concat( ' scope ', $doubleQuote, @mode, $doubleQuote )"
            }
            if "@priority" {
                value "concat( ' priority ', $doubleQuote, @priority, $doubleQuote )"
            }
            value "concat( $space, $openCurlyBracket, $return )"
            apply-templates using "./*" {
                with spaces "$newSpaces"
            }
            value "concat( $spaces, $closeCurlyBracket, $return )"
        }
    }
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:text" {
    parameter spaces
```

```
    variable conditionedText {
        call replaceStrings {
           with txt "."
        }
    }

    value "concat( $spaces, 'text' )"
    if "@disable-output-escaping" {
        value "concat( ' disable-output-escaping',
                       $doubleQuote, @disable-output-escaping, $doubleQuote )"
    }
    value "concat( $space, $doubleQuote, $conditionedText, $doubleQuote, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:value-of" {
    parameter spaces

    value "concat( $spaces, 'value ', $doubleQuote, @select, $doubleQuote, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:variable" {
    parameter spaces

    call pvwOutput {
        with inName "@name"
        with inValue "@select"
        with inKeyword "'variable'"
        with spaces "$spaces"
   }
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:when" {
    parameter spaces

    variable newSpaces "concat( $spaces, $identSpaces )"
    variable conditionedValue {
       call replaceStrings {
          with txt "@test"
       }
    }
    value "concat( $spaces, 'when ', $doubleQuote, $conditionedValue,
                   $doubleQuote, $space, $openCurlyBracket, $return )"
    apply-templates using "./*" {
        with spaces "$newSpaces"
    }
    value "concat( $spaces, $closeCurlyBracket, $return )"
}

// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
// -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*

match using "xsl:with-param"  {
    parameter spaces

    call pvwOutput {
      with inName "@name"
      with inValue "@select"
      with inKeyword "'with'"
      with spaces "$spaces"
    }
}
```

```
    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    // -*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*-*
    //
    // Process Remaining Elements
    //
    // Soak up any remaining elements not processed by the above.

    match using "node() | @*" { }


}
```