Revised proposal for streamable stylesheet functions

MHK 2015-02-18

In 9.3 xsl:param

Add two new attributes to xsl:param: streamable=boolean and consuming=boolean. Explain that they are applicable only when xsl:param is a child of xsl:function. Define error conditions: it's an error if streamable=yes is used on a param other than an xsl:function param, or on more than one xsl:param within a function. Define an error condition where exists(@consuming) and not(@streamable='yes'). Define an error condition where @streamable=yes and empty(@as) or where the as attribute has an empty intersection with U{N}.

Outline the purpose of these attributes: streamable declares the function to be streamable, and the relevant parameter to be the parameter to which streamed nodes can be supplied. Consuming (which defaults to true) declares that the function consumes any streamed nodes supplied to the parameter.

In 10.3.5 Streamability of Stylesheet Functions

Under specific conditions, described in this section, a stylesheeet function can be used to process nodes from a streamed input document.

A stylesheet function is declared-streamable if exactly one xsl:param child has the attribute streamable=yes. The argument corresponding to this xsl:param child is called the streaming argument.

Let PF and SF be the posture and sweep computed by applying the general streamability rules to a construct F whose sole operand is the sequence constructor making up the body of the function, with an operand usage that is the type-determined usage based on the declared return type of the function (that is, the value of xsl:function/@as, defaulting to item()*).

A stylesheet function is guaranteed-streamable if all the following conditions apply:

1. The function is declared streamable.
2. PF is grounded.
3. At least one of the following conditions is true:
    1. The streaming argument has consuming=no, and SF is motionless.
    2. The streaming argument has consuming=yes (the default), and SF is either motionless or consuming.

In 19.8.7.12 Streamability of Function Calls

<snip />

For a call to a stylesheet function, the general streamability rules apply. There is one operand for each supplied argument, and its operand usage depends on the corresponding xsl:param element P of the target function as follows:

1. In the case of an argument where P has the attribute streamable=yes, then the first of the following that applies:
    1. If P has the attribute consuming=no, then inspection
    2. If the posture of the supplied expression for argument P is crawling, then navigation
    3. Otherwise, absorption
2. For any other argument, the type-determined usage based on the as attribute of P.

In 3.6.3.3 Overriding Components

Two functions with the same name and arity are compatible if and only if they satisfy all the following rules:

- The types of the arguments are pairwise identical.
- The return types are identical.
- If the overridden function specifies identity-sensitive="no" then the overriding function also specifies identity-sensitive="no".
- The effective values of the streamable and consuming arguments on every xsl:param declaration in the overriding function are the same as the effective values of the same attributes on the corresponding xsl:param declaration in the overridden function.

In 19.8.7.10 Streamability of variable references:

The following rules apply to a variable reference that is bound to a parameter of a stylesheet function declared with streamable="yes" ~~where the declared type permits nodes. The declared type permits nodes if the as attribute on the xsl:param element is absent, or if it is a SequenceType that maps to a U-type that has a non-empty intersection with *U{N}* (see *19.2 Determining the Static Type of a Construct*)~~:

<snip/>

Examples

The first example in 10.3.5 remains streamable; it now needs to be written

```
<xsl:function name="f:depth" as="xs:integer"
              visibility="final">

  <xsl:param name="e" as="element()"
        streamable="true"/>

  <xsl:sequence select="max($e//*/count(ancestor::*)) -
count($e/ancestor::*)"/>

</xsl:function>
```

The second example in 10.3.5 remains streamable; it now needs to be written:

```
<xsl:function name="f:contains-PI" as="xs:boolean"
          visibility="final">

  <xsl:param name="e" as="node()" streamable="true"/>

  <xsl:sequence select="$e/self::processing-instruction()
or $e/*/f:contains-PI(.) = true()"/>

</xsl:function>
```

The third example, f:alternate-children, is not guaranteed streamable under the new rules because it returns streamed nodes. It is not streamable because we cannot determine the posture of those nodes without either (a) analyzing the body of the target function, or (b) declaring the result posture in the function signature. Note, a vendor could make this streamable, perhaps by adding an extension attribute that allows the result posture to be declared; or by inlining the function (given that its visibility is final).