Two useful XSLT runtime declarative techniques for XSL-FO

G. Ken Holman
XML Technology Lead
Réalta Online Publishing Solutions Ltd.
https://RealtaOnline.com

The business case

Many standards developing organizations (SDOs) have adopted NISO-STS (National Information Standards Organization - Standards Tag Suite), e.g.:

- the International Organization for Standardization (ISO),
- the International Electrotechnical Commission (IEC),
- the European Committee for Standardization (CEN), and
- the European Committee for Electrotechnical Standardization (CENELEC).

National Standards Bodies (NSBs) in Europe have business needs and legal obligations to publish standards, and some choose NISO-STS to create:

- their own country's national standards as standalone documents, and
- "adoptions" of standards created by SDOs, perhaps contextualized with national interpretations and supplemental content.

The organization writing a given NISO-STS XML document is the "originator"

Examples from Standards Norway

Norway has numerous national administrations to service with publishing, each with a unique document-wide appearance:

SVV (Statens vegvesen) - Norwegian public roads administration
NS3420 - Norwegian building and construction code standards
NORSOK - Norwegian petroleum industry standards

Norway is a member in CEN, CENELEC, ISO, and IEC:

NEK - Norwegian electrical and electronic standards
- **adoptions of CENELEC and IEC standards**

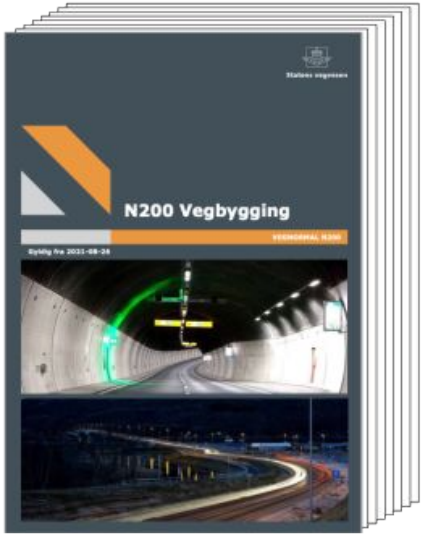
Norsk Standard
- **adoptions of CEN and ISO standards**

Examples from SN (Standard Norge) Norway (cont.)

National adoptions of regional and international standards

- one XML document with content created independently by up to three organizations;
- regional and international content is made available for download and announced in feeds that can be queried;
- up to three different appearances that are close but not identical ***in one aggregate document when merged***; and
- any given NISO-STS XML construct could be found in all three portions yet require a different appearance to be used in each published portion

Independent client publications:



Client adoptions of international standards:

Authored data for national standards

```
<standard>
  <front>
    <std-meta>
      <std-ident>
        <originator>SVV</originator>
        <doc-number>N200</doc-number>
      <custom-meta-group>
        <custom-meta>
          <meta-name>SVV author</meta-name>
          <meta-value>Erlend</meta-value>
    <sec sec-type="foreword">
      <title>Forord</title>
    <sec sec-type="intro">
      <title>Innledning</title>
    <sec sec-type="intro">
      <title>Krav til funksjon</title>
  <body>
    <sec>
      <label>1</label>
      <title>Underbygning og grunnforhold</title>
  <back>
    <ref-list content-type="bibl">
      <title>Referanser</title>
      <ref content-type="norm-refs">
        <label>[86]</label>
        <mixed-citation>Avfallsforskriften. Forskrift
        om gjenvinning og behandling av avfall - Kapittel
        14A. Betong og tegl fra riveprosjekter.
        FOR-2004-06-01-930</mixed-citation>
        </ref>
```

```
<standard>
  <front>
    <std-meta>
      <std-ident>
        <originator>SVV</originator>
        <doc-number>N200</doc-number>
      <custom-meta-group>
        <custom-meta>
          <meta-name>realta-merge nat xml</meta-name>
          <meta-value>local/nat/N200.xml</meta-value>
  <body/>
```

Fetched non-nested data for adoptions

```
<adoption xml:lang="en">
  <adoption-front>
    <std-meta>
      <std-ref type="dated">NEK EN IEC 60034-33:2022</std-ref>
      <custom-meta-group>
        <custom-meta>
          <meta-name>realta-merge nat xml</meta-name>
          <meta-value>local/nat/60034-33-2022.xml</meta-value>
  <adoption>
    <adoption-front>
      <std-meta std-meta-type="european">
        <custom-meta-group originator="realta">
          <custom-meta>
            <meta-name>realta-fetch cenelec xml</meta-name>
            <meta-value>68956</meta-value>
    <standard>
      <front>
        <std-meta std-meta-type="international">
          <custom-meta-group originator="realta">
            <custom-meta>
              <meta-name>realta-fetch iec xml</meta-name>
              <meta-value>23163</meta-value>
      <body/>
    </standard>
  </adoption>
</adoption>
```

```
<adoption xml:lang="en">
  <adoption-front>
    <std-meta>
      <std-ref type="dated">NEK EN IEC 60034-33:2022</std-ref>
    <sec id="sec_nat-foreword_en" sec-type="foreword">...
    <sec id="sec_nat-foreword_no" sec-type="foreword" xml:lang="nb">...
  <back>
    <app-group>
      <app id="nat-annex">...
  <adoption>
    <adoption-front>
      <std-meta std-meta-type="european">
        <std-ref type="dated">EN IEC 60034-33:2022</std-ref>...
    <back>
      <app-group>
        <app id="cen-annex">...
    <standard>
      <front>
        <std-meta std-meta-type="international">
          <std-ref type="dated">IEC 60034-33:2022</std-ref>...
      <body>
        ...
      </body>
      <back>
        <app-group>
          <app id="iec-annex">...
    </standard>
  </adoption>
</adoption>
```

Fetched nested data for adoptions

```
<adoption xml:lang="en">
  <adoption-front>
    <std-meta>
      <std-ref type="dated">NEK EN IEC 60034-33:2022</std-ref>
      <custom-meta-group>
        <custom-meta>
          <meta-name>realta-merge nat xml</meta-name>
          <meta-value>local/nat/60034-33-2022.xml</meta-value>
  <adoption>
    <adoption-front>
      <std-meta std-meta-type="european">
        <custom-meta-group originator="realta">
          <custom-meta>
            <meta-name>realta-fetch cenelec xml nested</meta-name>
            <meta-value>68956</meta-value>
    <standard>
      <front>
        <std-meta std-meta-type="international">
          <custom-meta-group originator="realta">
            <custom-meta>
              <meta-name>realta-fetch iec xml nested</meta-name>
              <meta-value>23163</meta-value>
      <body/>
    </standard>
  </adoption>
</adoption>
```

```
<adoption xml:lang="en">
  <adoption-front>
    <std-meta>
      <std-ref type="dated">NEK EN IEC 60034-33:2022</std-ref>
    <sec id="sec_nat-foreword_en" sec-type="foreword">...
    <sec id="sec_nat-foreword_no" sec-type="foreword" xml:lang="nb">...
  <adoption>
    <adoption-front>
      <std-meta std-meta-type="european">
        <std-ref type="dated">EN IEC 60034-33:2022</std-ref>...
    <standard>
      <front>
        <std-meta std-meta-type="international">
          <std-ref type="dated">IEC 60034-33:2022</std-ref>...
      <body>
        ...
      </body>
      <back>
        <app-group>
          <app id="iec-annex">...
    </standard>
    <back>
      <app-group>
        <app id="cen-annex">...
  </adoption>
  <back>
    <app-group>
      <app id="nat-annex">...
</adoption>
```

XSLT 2 compile-time techniques that fall short

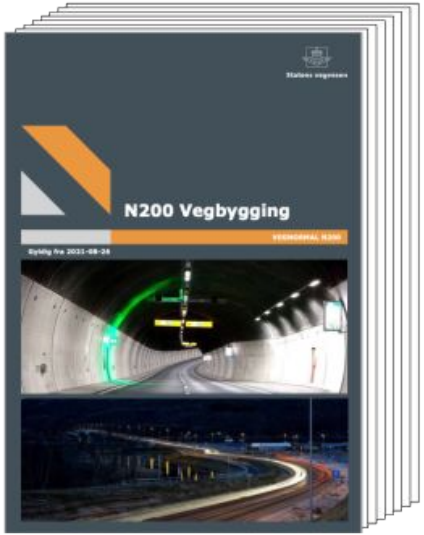
Straightforward requirement for single document-wide appearance

- built-in XSLT and XSL-FO facilities for overriding a core behaviour with a single fragment's overrides
  - import precedence of:
    - overriding named variables
    - overriding named templates
    - combining named attribute sets
  - import precedence fully established at compile time for single named construct

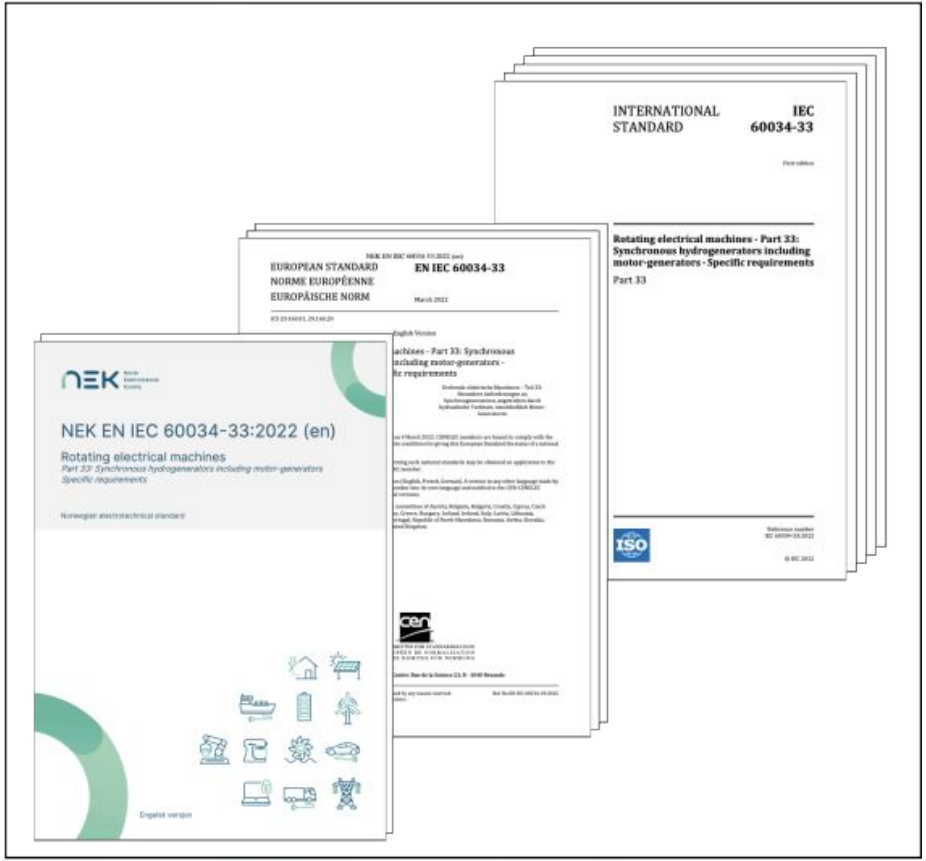
Much more challenging requirement for differing results in one document

- maintaining code blocks using imperative methods is untenable as the number of differentiations grows
  - awkward to use if/else-if/else in the core to react to possible configurations
  - only a single variable or named template of a given name can be in-play
  - putting client-specific appearances in the core code risks disturbing long-running results

Independent client publications:



Client adoptions of international standards:



The technical challenge (cont.)

Using declarative methods is the only reasonable approach to maintenance

- treat the core code as read-only and put environment in variables
- abandon traditional import overrides for all client requirements

Encapsulate the client environment in stylesheet fragments

- assemble the piecemeal support fragments into stylesheets
- ensure fragments accommodate but not depend on other fragments

Use runtime techniques to access the data layers individually to meet client requirements:

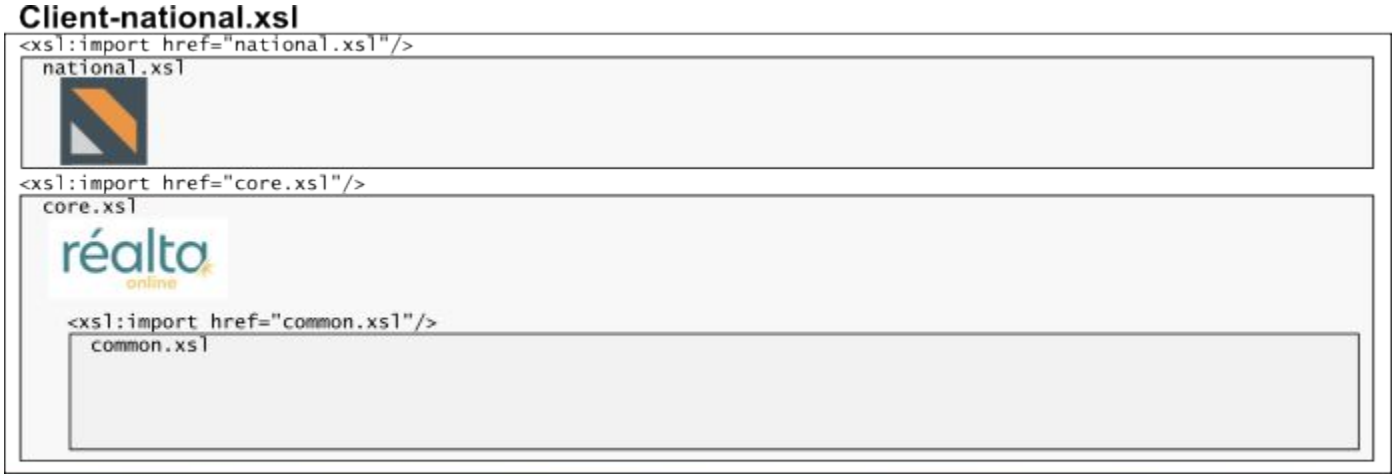
- "is this layer of NISO-STS supported by this stylesheet?"
- "process this NISO-STS construct using this layer's environment"
- "accommodate client overrides of a layer written by a supplier"

Single document-wide appearance

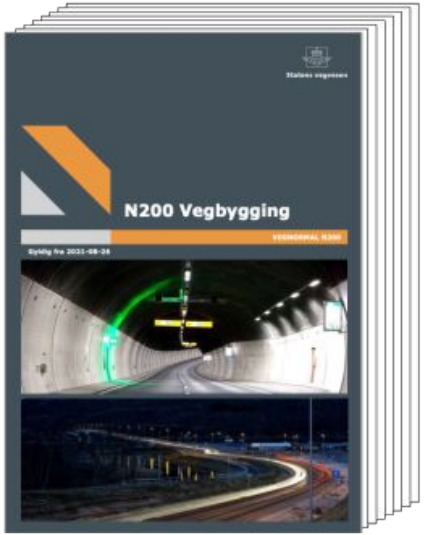
Different entry points in the server API trigger different layout results

- **the core code has no awareness of the fragments calling it**
- **a single national fragment added to the core provides the overrides**

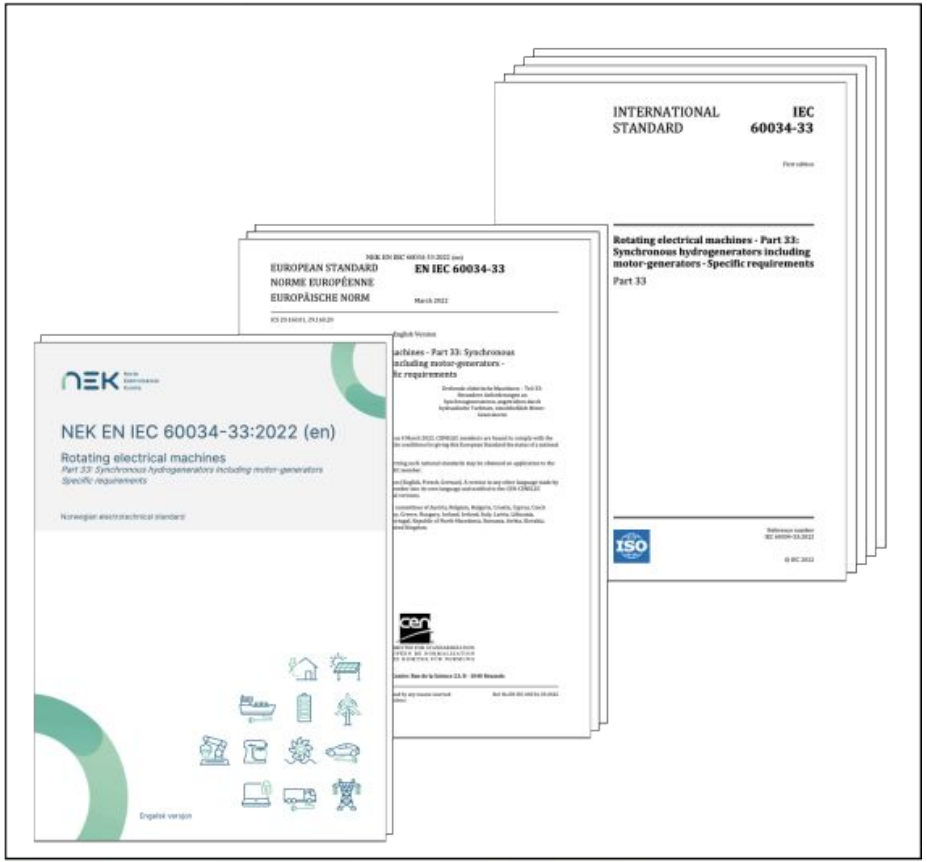
(Note in these diagrams, the importation statements are depicted in the reverse order of the importation precedence in order to relate, vertically, a top-down cascading nature of precedence)



Independent client publications:



Client adoptions of international standards:



Multiple appearances in one document

Challenge: can the core be informed differently on a per-layer basis?

NEK-adoption.xsl

```
<xsl:import href="nek.xsl"/>
  nek.xsl - national layer
<xsl:import href="ccmc.xsl"/>
  ccmc.xsl - regional layer
<xsl:import href="iso.xsl"/>
  iso.xsl - international layer
<xsl:import href="core.xsl"/>
  core.xsl - STS interpretation
    <xsl:import href="common.xsl"/>
      common.xsl - default handling
```

Typical layered stylesheet adaptation

```
<block>
  <xsl:apply-templates select="$this-std-meta" mode="c:copyright"/>
</block>
```

NEK-adoption.xsl

```
<xsl:import href="nek.xsl"/>
```

nek.xsl - national layer

Norsk Elektroteknisk Komite

```
<xsl:template match="std-meta[c:originator(.)=$i-orig]" mode="c:copyright">
  <xsl:next-match/><xsl:text> Copyright © NEK</xsl:text>
<xsl:template match="std-meta[c:originator(.)=$n-orig]" mode="c:copyright">
  <xsl:text>Copyright © NEK</xsl:text>
```

```
<xsl:import href="ccmc.xsl"/>
```

ccmc.xsl - regional layer

```
<xsl:template match="std-meta[c:originator(.)=$c-orig]" mode="c:copyright">
  <xsl:text>Copyright © CEN</xsl:text>
```

```
<xsl:import href="iso.xsl"/>
```

iso.xsl - international layer

```
<xsl:template match="std-meta[c:originator(.)=$i-orig]" mode="c:copyright">
  <xsl:text>Copyright © ISO</xsl:text>
```

```
<xsl:import href="core.xsl"/>
```

core.xsl - STS interpretation

```
<xsl:template match="std-meta[c:originator(.)=$r-orig]" mode="c:copyright">
</xsl:template>
```

```
<xsl:import href="common.xsl"/>
```

common.xsl - default handling

```
<xsl:template match="std-meta" mode="c:copyright">
</xsl:template>
```

Atypical declarative assignment:

```
<xsl:variable name="c:unsupported-layers" as="xsd:string*">
  <xsl:for-each select="$c:layer/(adoption-front,front)">
    <xsl:variable name="c:supported" as="xsd:boolean">
      <xsl:apply-templates select="std-meta" mode="c:check"/>
    </xsl:variable>
    <xsl:if test="not($c:supported)">
      <xsl:value-of select="c:originator(std-meta)"/>
```

NEK-adoption.xsl

```
<xsl:import href="nek.xsl"/>
```

nek.xsl - national layer

Norsk Elektroteknisk Komite

```
<xsl:variable name="n-orig" select="'NEK'"/>
...
<xsl:template match="std-meta[c:originator(.)=$n-orig]" mode="c:check"
              as="xsd:boolean"> <xsl:sequence select="true()"/>
```

```
<xsl:import href="ccmc.xsl"/>
```

ccmc.xsl - regional layer

cen

```
<xsl:variable name="c-orig" select="'CEN'"/>
...
<xsl:template match="std-meta[c:originator(.)=$c-orig]" mode="c:check"
              as="xsd:boolean"> <xsl:sequence select="true()"/>
```

```
<xsl:import href="iso.xsl"/>
```

iso.xsl - international layer

ISO

```
<xsl:variable name="i-orig" select="'ISO'"/>
...
<xsl:template match="std-meta[c:originator(.)=$i-orig]" mode="c:check"
              as="xsd:boolean"> <xsl:sequence select="true()"/>
```

```
<xsl:import href="core.xsl"/>
```

core.xsl - STS interpretation

réalta online

```
<xsl:variable name="r-orig" select="'Realta'"/>
...
<xsl:template match="std-meta[c:originator(.)=$r-orig]" mode="c:check"
              as="xsd:boolean"> <xsl:sequence select="true()"/>
```

```
<xsl:import href="common.xsl"/>
```

common.xsl - default handling

```
<xsl:template match="std-meta" mode="c:check" as="xsd:boolean">
  <xsl:sequence select="false()"/>
```

Layered control:

```
<xsl:variable name="c:core-format" as="document-node()">
  <xsl:document>
    <c:note space-before="1em" font-size="10pt"/>
    <c:notePrefix font-size="1.2em" font-weight="bold"/>
<xsl:apply-templates select="$c:this-layer-top">
  <xsl:with-param name="c:f" tunnel="yes" as="document-node()*">
    <xsl:apply-templates select="$this-std-meta" mode="c:format"/>
  </xsl:with-param>
```

NEK-adoption.xsl

```
<xsl:import href="nek.xsl"/>
  nek.xsl - national layer
<xsl:variable name="c:n-format" as="document-node()">
  <xsl:document>
<xsl:template match="std-meta[c:originator(.)=$n-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:n-format"/>
```

```
<xsl:import href="ccmc.xsl"/>
  ccmc.xsl - regional layer
<xsl:variable name="c:c-format" as="document-node()">
  <xsl:document> <c:notePrefix font-weight="normal"/>
<xsl:template match="std-meta[c:originator(.)=$c-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:c-format"/>
```

```
<xsl:import href="iso.xsl"/>
  iso.xsl - international layer
<xsl:variable name="c:i-format" as="document-node()">

<xsl:template match="std-meta[c:originator(.)=$i-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:i-format"/>
```

```
<xsl:import href="core.xsl"/>
  core.xsl - STS interpretation
<xsl:template match="non-normative-note">
  <xsl:param name="c:f" tunnel="yes" as="document-node()*"/>
  <block> <xsl:copy-of select="$c:f / c:note / @*"/>
    <inline> <xsl:copy-of select="$c:f / c:notePrefix / @*"/>
      <xsl:apply-templates select="label"> </inline>
    <xsl:apply-templates select="* except label/>
```

```
  <xsl:import href="common.xsl"/>
    common.xsl - default handling
<xsl:template match="std-meta" mode="c:format" as="document-node()*">
  <xsl:sequence select="$c:core-format"/>
```

Layered control over XSL-FO formatting (cont.)

Simulate declarative attribute sets in real time by using a tunnel parameter:

```
<xsl:variable name="c:core-format" as="document-node()">
  <xsl:document>
      <c:note space-before="1em" font-size="10pt"/>
      <c:notePrefix font-size="1.2em" font-weight="bold"/>

<xsl:apply-templates select="$c:this-layer-top">
  <xsl:with-param name="c:f" tunnel="yes" as="document-node()*">
      <xsl:apply-templates select="$c:this-std-meta" mode="c:format"/>

<xsl:template match="non-normative-note">
  <xsl:param name="c:f" tunnel="yes" as="document-node()*"/>
  <block>
    <xsl:copy-of select="$c:f / c:note / @*"/>
    <inline>
      <xsl:copy-of select="$c:f / c:notePrefix / @*"/>
      <xsl:apply-templates select="label">
    </inline>
    <xsl:apply-templates select="* except label/>

<xsl:template match="std-meta" mode="c:format" as="document-node()*">
  <xsl:sequence select="$c:core-format"/>
```

Layered control over XSL-FO formatting (cont.)

A given fragment can add following overriding attribute sets without changing the core:

```
<xsl:variable name="c:core-format" as="document-node()">
  <xsl:document>
      <c:note space-before="1em" font-size="10pt"/>
      <c:notePrefix font-size="1.2em" font-weight="bold"/>

<xsl:variable name="c:c-format" as="document-node()">
  <xsl:document>
    <c:notePrefix font-weight="normal"/>

<xsl:template match="std-meta[c:originator(.)=$c-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:c-format"/>

<xsl:template match="non-normative-note">
  <xsl:param name="c:f" tunnel="yes" as="document-node()*"/>
  <block>
    <xsl:copy-of select="$c:f / c:note / @*"/>
    <inline>
      <xsl:copy-of select="$c:f / c:notePrefix / @*"/>
      <xsl:apply-templates select="label"/>
    </inline>
    <xsl:apply-templates select="* except label"/>
```

Overriding another layer's appearance

Consider that a client might want the ISO note prefixes also not to be bolded

- **one fragment can override another fragment's processing by declaring a set of overriding attributes when intercepting the fragment's definition**

- in the core fragment:

```
<xsl:variable name="c:core-format" as="document-node()">
 <xsl:document>
  <c:note space-before="1em" font-size="10pt"/>
  <c:notePrefix font-size="1.2em" font-weight="bold"/>
```

- in the ISO fragment:

```
<xsl:template match="std-meta[c:originator(.)=$c:i-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:i-format"/>
```

- in the client's fragment (at higher precedence than the ISO fragment):

```
<xsl:template match="std-meta[c:originator(.)=$c:i-orig]" mode="c:format">
  <xsl:next-match/><!--first use whatever ISO needs to use-->
  <xsl:document><!--now add this fragment's overrides to the ISO fragment-->
    <c:notePrefix font-weight="normal"/>
```

Layered control:

```
<xsl:variable name="c:core-format" as="document-node()">
  <xsl:document>
    <c:note space-before="1em" font-size="10pt"/>
    <c:notePrefix font-size="1.2em" font-weight="bold"/>
<xsl:apply-templates select="$c:this-layer-top">
  <xsl:with-param name="c:f" tunnel="yes" as="document-node()*">
    <xsl:apply-templates select="$this-std-meta" mode="c:format"/>
  </xsl:with-param>
```

NEK-adoption.xsl

```
<xsl:import href="nek.xsl"/>
  nek.xsl - national layer
<xsl:variable name="c:n-format" as="document-node()">
  <xsl:document>
<xsl:template match="std-meta[c:originator(.)=$n-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:n-format"/>
<xsl:import href="ccmc.xsl"/>
  ccmc.xsl - regional layer
<xsl:variable name="c:c-format" as="document-node()">
  <xsl:document> <c:notePrefix font-weight="normal"/>
<xsl:template match="std-meta[c:originator(.)=$c-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:c-format"/>
<xsl:import href="iso.xsl"/>
  iso.xsl - international layer
<xsl:variable name="c:i-format" as="document-node()">
<xsl:template match="std-meta[c:originator(.)=$i-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:i-format"/>
<xsl:import href="core.xsl"/>
  core.xsl - STS interpretation
<xsl:template match="non-normative-note">
  <xsl:param name="c:f" tunnel="yes" as="document-node()*"/>
  <block> <xsl:copy-of select="$c:f / c:note / @*"/>
    <inline> <xsl:copy-of select="$c:f / c:notePrefix / @*"/>
      <xsl:apply-templates select="label"> </inline>
    <xsl:apply-templates select="* except label/>
  <xsl:import href="common.xsl"/>
    common.xsl - default handling
<xsl:template match="std-meta" mode="c:format" as="document-node()*">
  <xsl:sequence select="$c:core-format"/>
```

And it worked ... until it didn't work

For years the technique worked until...

- coding for a new client, a new set of overriding attributes appeared to be ignored ... this didn't work anymore:

```
<xsl:variable name="c:core-format" as="document-node()">
<xsl:variable name="c:n-format" as="document-node()">

$c:f assignment:
<xsl:sequence select="$c:core-format,$c:n-format"/>

<xsl:copy-of select="$c:f / c:notePrefix / @*"/>
```

And it worked ... until it didn't work (cont.)

There is no control over the document order of a set of document nodes

- the "/" operator rearranges the LHS in document order before evaluating the RHS, and `$n-format` ended up before `$core-format`, so use "!"!

```
<xsl:variable name="c:core-format" as="document-node()">
<xsl:variable name="c:n-format" as="document-node()">

$c:f assignment:
<xsl:sequence select="$c:core-format,$c:n-format"/>

<xsl:copy-of select="$c:f ! c:notePrefix ! @*"/>
```

Note that "!" is XSLT 3 ... for a pure XSLT 2 solution, $c:f assignment can create and return a single document node with ordered children.

Working code:

```
<xsl:variable name="c:core-format" as="document-node()">
  <xsl:document>
    <c:note space-before="1em" font-size="10pt"/>
    <c:notePrefix font-size="1.2em" font-weight="bold"/>
<xsl:apply-templates select="$c:this-layer-top">
  <xsl:with-param name="c:f" tunnel="yes" as="document-node()*">
    <xsl:apply-templates select="$this-std-meta" mode="c:format"/>
  </xsl:with-param>
```

NEK-adoption.xsl

```
<xsl:import href="nek.xsl"/>
  nek.xsl - national layer
<xsl:variable name="c:n-format" as="document-node()">
  <xsl:document>
<xsl:template match="std-meta[c:originator(.)=$n-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:n-format"/>
<xsl:import href="ccmc.xsl"/>
  ccmc.xsl - regional layer
<xsl:variable name="c:c-format" as="document-node()">
  <xsl:document> <c:notePrefix font-weight="normal"/>
<xsl:template match="std-meta[c:originator(.)=$c-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:c-format"/>
<xsl:import href="iso.xsl"/>
  iso.xsl - international layer
<xsl:variable name="c:i-format" as="document-node()">
<xsl:template match="std-meta[c:originator(.)=$i-orig]" mode="c:format">
  <xsl:sequence select="$c:core-format,$c:i-format"/>
<xsl:import href="core.xsl"/>
  core.xsl - STS interpretation
<xsl:template match="non-normative-note">
  <xsl:param name="c:f" tunnel="yes" as="document-node()*"/>
  <block> <xsl:copy-of select="$c:f ! c:note ! @*"/>
    <inline> <xsl:copy-of select="$c:f ! c:notePrefix ! @*"/>
      <xsl:apply-templates select="label"> </inline>
    <xsl:apply-templates select="* except label/>
  <xsl:import href="common.xsl"/>
    common.xsl - default handling
<xsl:template match="std-meta" mode="c:format" as="document-node()*">
  <xsl:sequence select="$c:core-format"/>
```

$c:f can be used for even more

Having a configuration parameter for every fragment promotes customization

- not limited just to XSL-FO attributes: can do the same for strings, booleans, or any value ... just always act on the last one in $c:f

```
<c:toc c:scope="{'layer' (: global = all layers are printed
                            lower = given and lower layers are printed
                            layer = only the given layer is printed
                            none = no ToC is printed for the given layer
                            :)}"
    c:includePageOnPage="true" c:includeTitleOnPage="true"
    c:includeTitleInToC="false" c:titlePhraseLookup="Contents"
    c:separator="dots" c:pageBreakType="page" c:pageAfterBreakType="page"/>
<c:tocContentsPageLine text-align-last="justify" text-align="justify"
                       space-after="1em"/>
```

- *(note the commenting technique for the attribute)*

Two useful XSLT runtime declarative techniques for XSL-FO

G. Ken Holman
XML Technology Lead
Réalta Online Publishing Solutions Ltd.
https://RealtaOnline.com