XML Corrupting the Youth

Florian Guthmann

2024-11-14

```
$Id: talk.xml,v 1.16 2024/11/13 17:37:33 flo Exp $
$Id: talk.xsl,v 1.10 2024/11/13 17:37:52 flo Exp $
```

Why this talk?

information

polemics and apologetics

XML's Essence

e**X**tensible **M**arkup **L**anguage

e**X**tensible **M**arkup **L**anguage

Def. A *Markup Language* is a *text-based* language that specifies

- the *structure*
- the *formatting*
- the *relations* between parts

of a document.

Example HTML, Markdown, reStructuredText, XML, ...

Opinionated Requirements

A Markup Language should be

1. Easy to write
2. Easy to read
3. Easy to extend

	1.	2.	3.
HTML	~	✓	✖
Markdown	✓	✓	✖
XML	~	✓	✓

An Example

```
<syllogism>
    <premise type="major">
        All humans are mortal.
    </premise>
    <premise type="minor">
        Socrates is human.
    </premise>
    <consequent>
        Socrates is mortal.
    </consequent>
</syllogism>
```

XML Nodes

```
<namespace:premise type="major" … >
…
</namespace:premise>
```

Opening *tag* and closing *tag*

```
<self-closing foo="5"/>
```

Self-closing tags

XML Nodes

```
<premise type="major" … >
…
</premise>
```

Attribute *name* and *value*

XML Syntax

```
node() | text() | …[0]
```

[0] other stuff I won't talk about

XPath

A query language for XML Documents.

- Basic path syntax like Unix-style file paths (XML nodes as inodes)
- Extend were necessary (e.g. @ for attributes and [ ] for conditions)

```
//premise

/syllogism[1]

premise[@type = 'major']

syllogism/conclusion[count(preceding-sibling::premise) = 2]
```

XML Schema

Describe and *constrain* the structure and contents of XML documents.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/
TR/html4/strict.dtd">
```

XSL

Q: So we can extend the markup language with new constructs. But how do we specify their semantics?

A: A domain specific programming language that allows assigning meaning to constructs declaratively.

Q: How should this language look like?

A: Well, we already have XML...

e**X**tensible **S**tylesheet **L**anguage

- A programming language based on *XML* and *XPath*.
- *Templates* are the basic construct
- Recursively walk through an XML document. At any node, check if any templates are applicable and execute those that match.

A simple Template

```
<foo>
    <bar>
        baz
    </bar>
</foo>
```

↓

```
<bar>
    <foo>
        baz
    </foo>
</bar>
```

```
<xsl:template match="foo">
    <bar>
        <xsl:apply-templates/>
    </bar>
</xsl:template>

<xsl:template match="bar">
    <foo>
        <xsl:apply-templates/>
    </foo>
</xsl:template>

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

A revisited example

```
<syllogism>
    <premise type="major">
        All humans are
        mortal.
    </premise>
    <premise type="minor">
        Socrates is human.
    </premise>
    <consequent>
        Socrates is mortal.
    </consequent>
</syllogism>
```

All humans are mortal.
Socrates is human.

Socrates is mortal.

A revisited example

```
<xsl:template match="syllogism">
    <fo:block-container>
        <xsl:apply-templates select="premise"/>
        <fo:block>
            <fo:leader leader-pattern="rule" rule-thickness="3pt"/>
        </fo:block>
        <xsl:apply-templates select="consequent"/>
    </fo:block-container>
</xsl:template>

<xsl:template match="syllogism/premise">
    <fo:block>
        <xsl:if test="@type = 'major'">
            <xsl:attribute name="font-weight">
                bold
            </xsl:attribute>
        </xsl:if>
        <xsl:apply-templates/>
    </fo:block>
</xsl:template>

<xsl:template match="syllogism/consequent">
    <fo:block font-style="italic">
        <xsl:apply-templates/>
    </fo:block>
</xsl:template>
```

XSL Interpreters and Where to Find Them

- libxslt's `xsltproc` 

- Apache FOP 

- Your Browser![0] 

[0] Unless you use lynx, ladybird, eww…

AD PRIMUM SIC PROCEDITUR.

XML is too verbose!

SED CONTRA,

Verbosity is good, actually. What's not to like about

```
<xsl:call-template name="upper-case">
    <xsl:with-param name="the-string">
        foo
    </xsl:with-param>
</xsl:call-template>
```

RESPONDEO

Use a better editor.

Tahāfut al-Tahāfut

AD SECUNDUM SIC PROCEDITUR. XSLT is too limited!

SED CONTRA, Just use XSLT 2.0!

RESPONDEO But *nobody* supports it.

AD TERTIUM SIC PROCEDITUR.

My XML Parser failed because of a DNS error

SED CONTRA,

How could you trust your data if it wasn't validated?

RESPONDEO

uff

About this presentation

- Written in a bespoke XML dialect, translated via XSL-FO to PDF using Apache FOP
- Version-controlled using RCS, because why stop pretending it's 2002 now?
- Can be found at https://wwwcip.cs.fau.de/~oc45ujef/misc/talks/xml