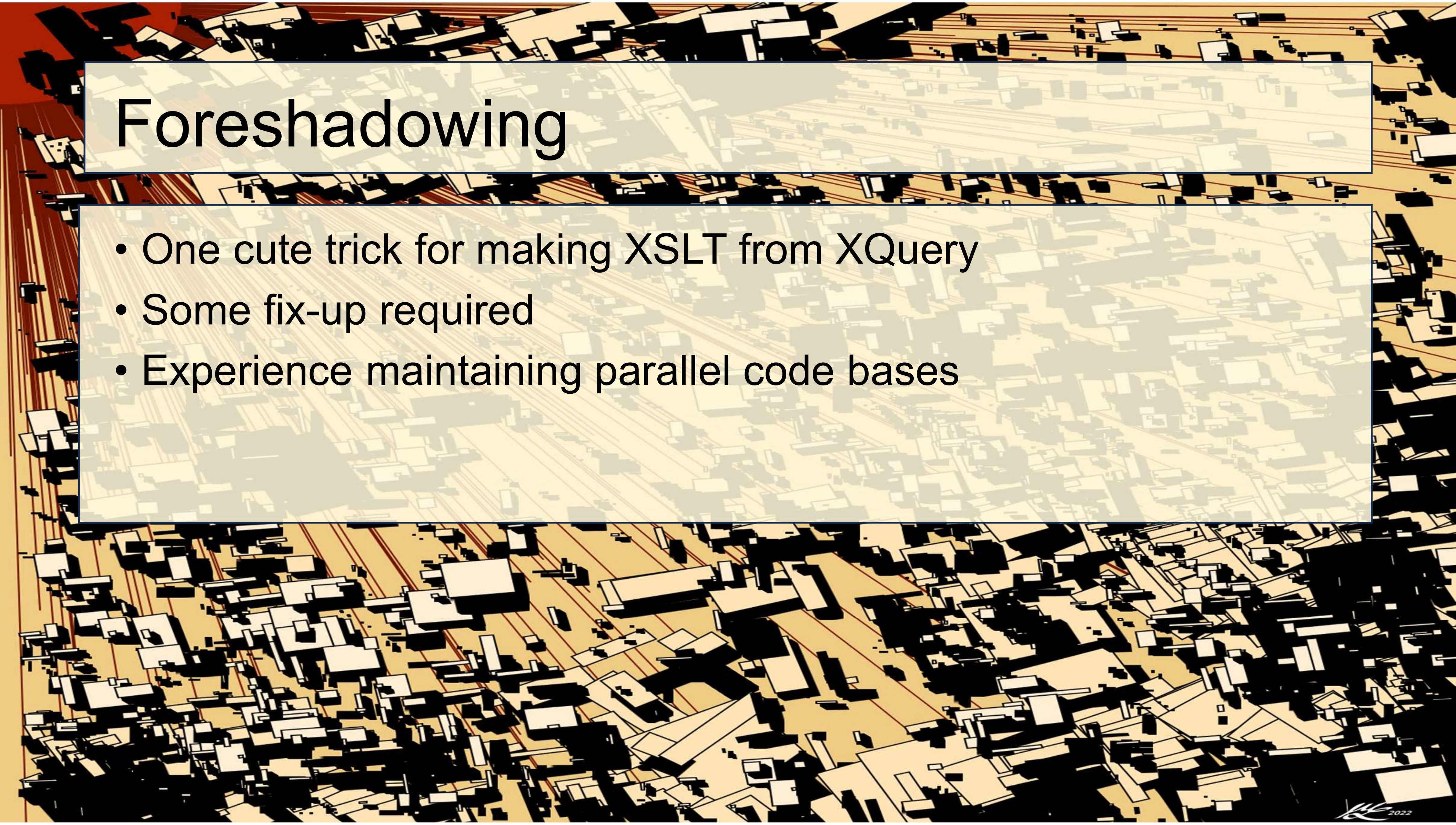
Adventures in Single-Sourcing XQuery and XSLT

Mary Holstege

@mathling@mastodon.social

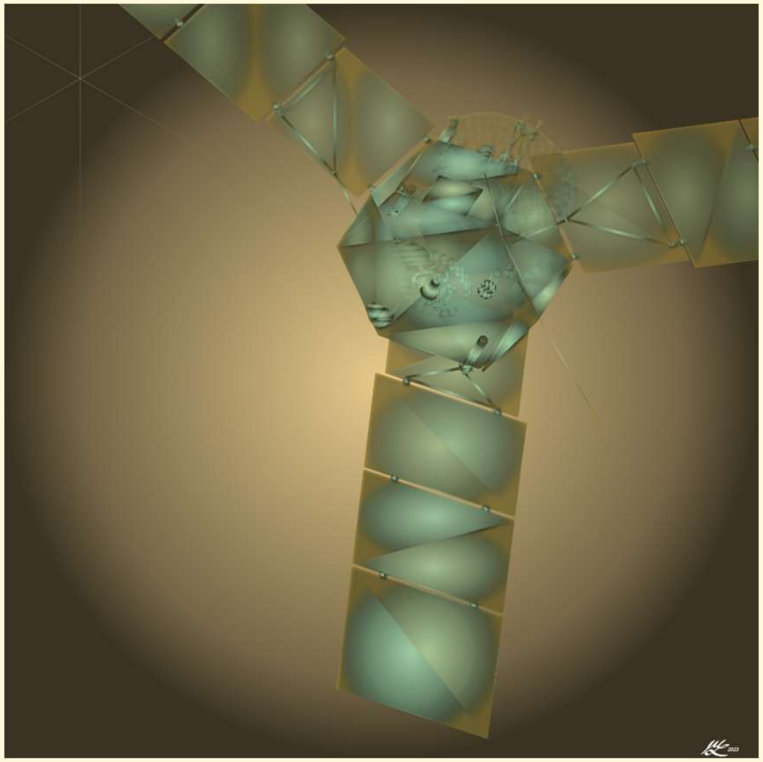


Foreshadowing

- One cute trick for making XSLT from XQuery
- Some fix-up required
- Experience maintaining parallel code bases

Effective solutions to over-constrained problems

- No perfect solution
- Effective compared to what?
- Opinions and feelings differ



@mathling@mastodon.social

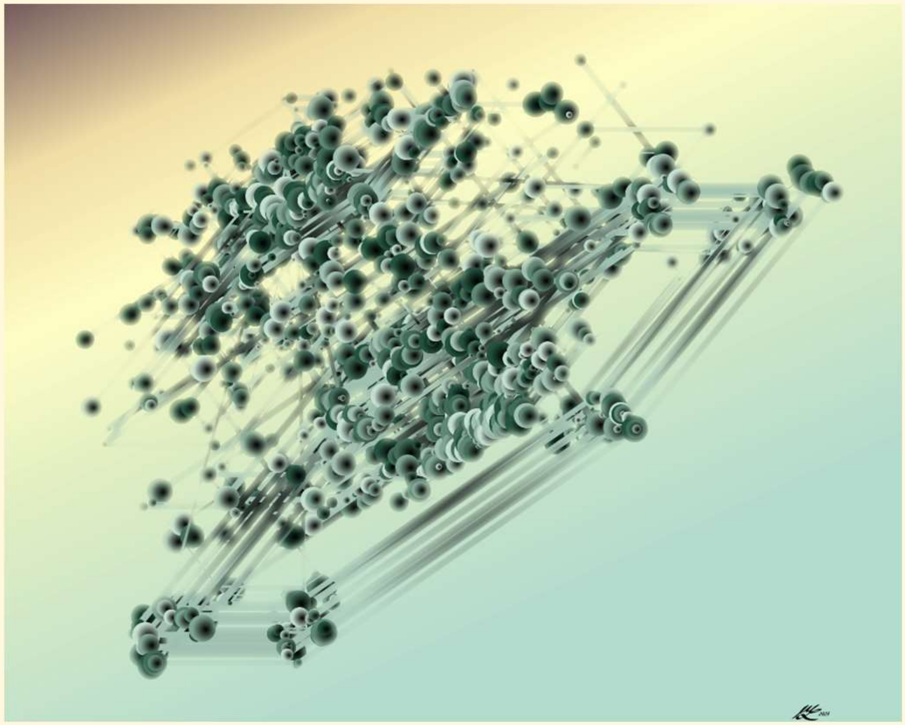
Goals

Easy to implement

Doesn’t have to be completely automated

Maintainability over idiomatic usage

XQuery Libraries for Making Art

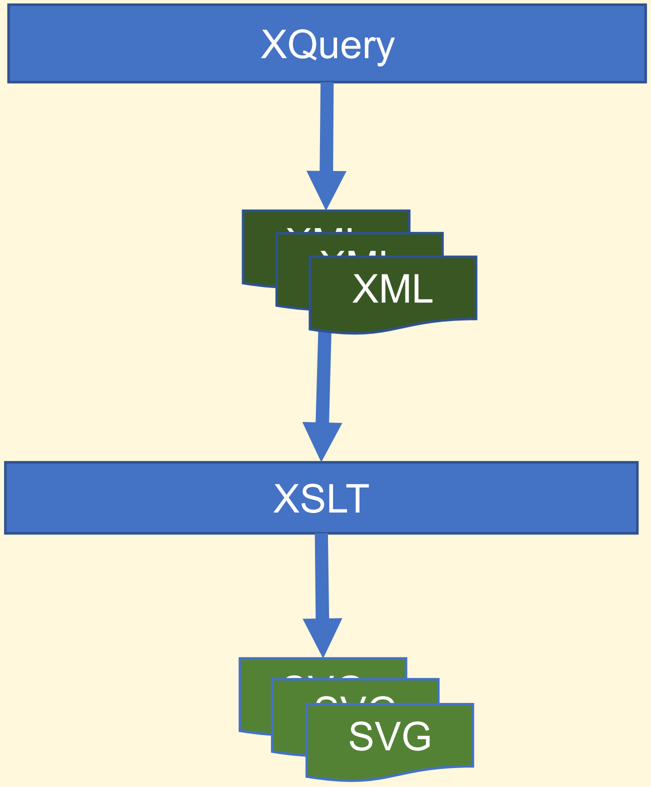


- 200 libraries
- 4700 functions
- 150Kloc

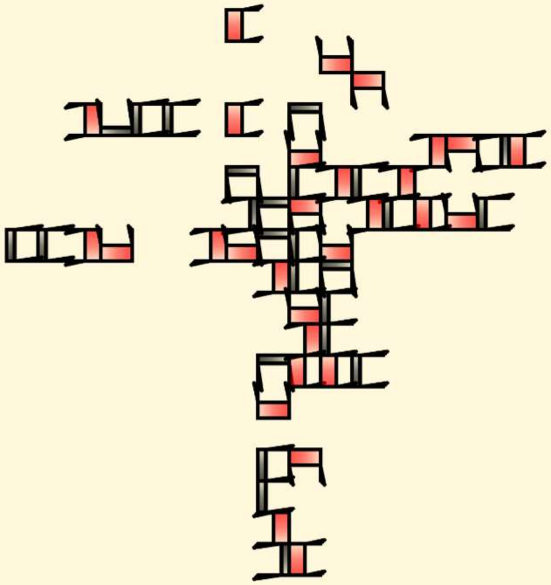
- Geometries, random distributions, colour manipulations, tilings, curve plotting, image manipulation, …

@mathling@mastodon.social

XQuery from XSLT

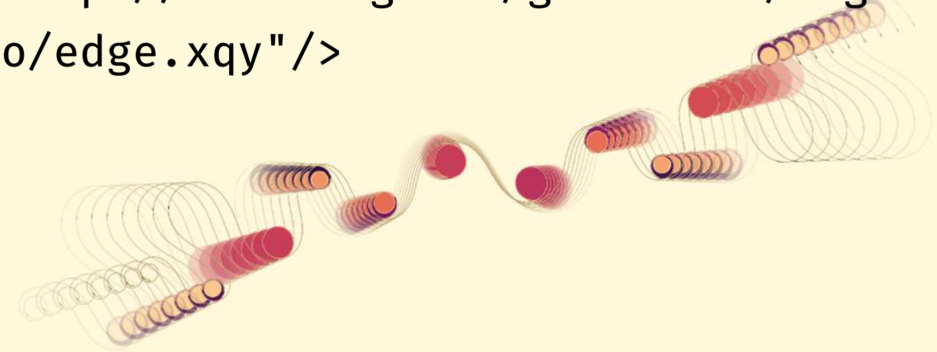


- Make XQuery functions available to my stylesheets



XQuery from XSLT: the easy way

```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:edge="http://mathling.com/geometric/edge"
                xmlns:saxon="http://saxon.sf.net/"
                exclude-result-prefixes="saxon edge“
                extension-element-prefixes="saxon"
                version="3.0">
  <saxon:import-query namespace="http://mathling.com/geometric/edge"
                      href="../geo/edge.xqy"/>
</xsl:stylesheet>
```

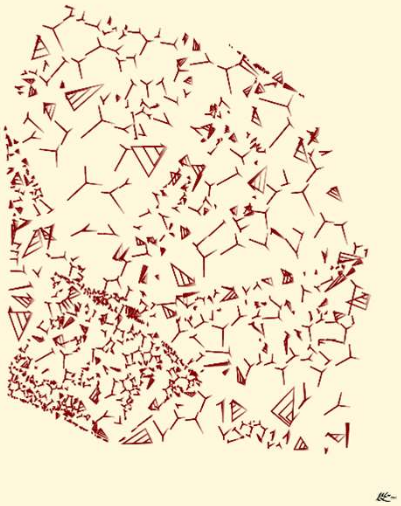


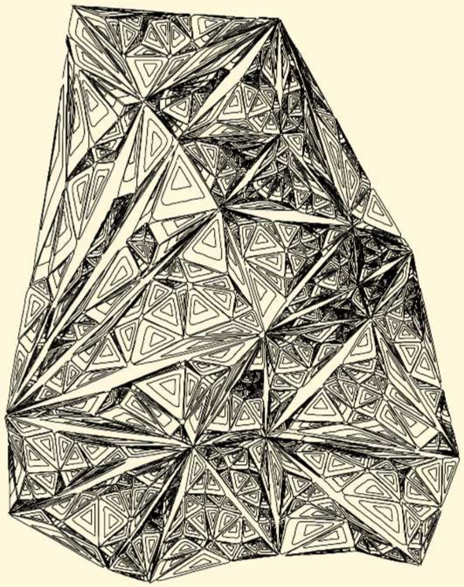
XQuery from XSLT: the standard way

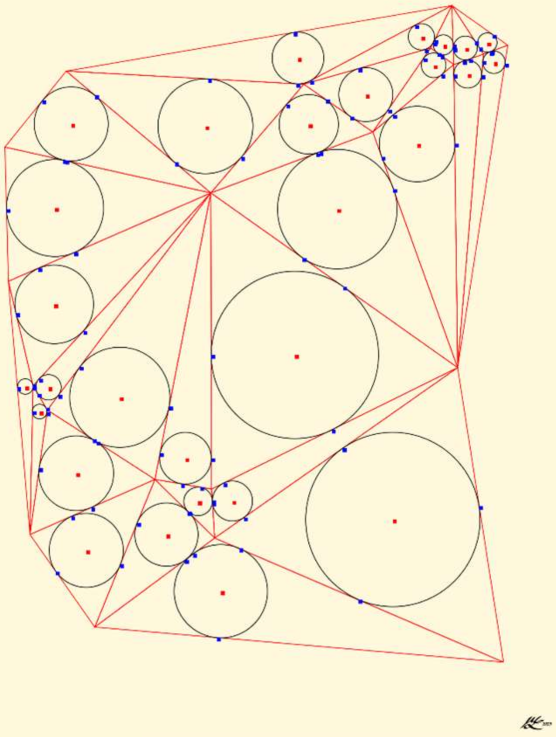
```
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                xmlns:map="http://www.w3.org/2005/xpath-functions/map"
                xmlns:util="http://mathling.com/core/utilities"
                xmlns:xs="http://www.w3.org/2001/XMLSchema"
                exclude-result-prefixes="util xs map" version="3.0">
<xsl:variable name="mod.util" as="map(*)"
   select="load-xquery-module('http://mathling.com/core/utilities',
           map {'location-hints':'../core/utilities.xqy'})"/>
<xsl:function name="util:is-prime" as="xs:boolean">
  <xsl:param name="i" as="xs:integer"/>
  <xsl:sequence select="
    $mod.util('functions')(QName('http://mathling.com/core/utilities','is-prime'))(1)($i)
  "/>
</xsl:function>...
```

Approaches

- By hand
- Substring tokenization
- XQDoc

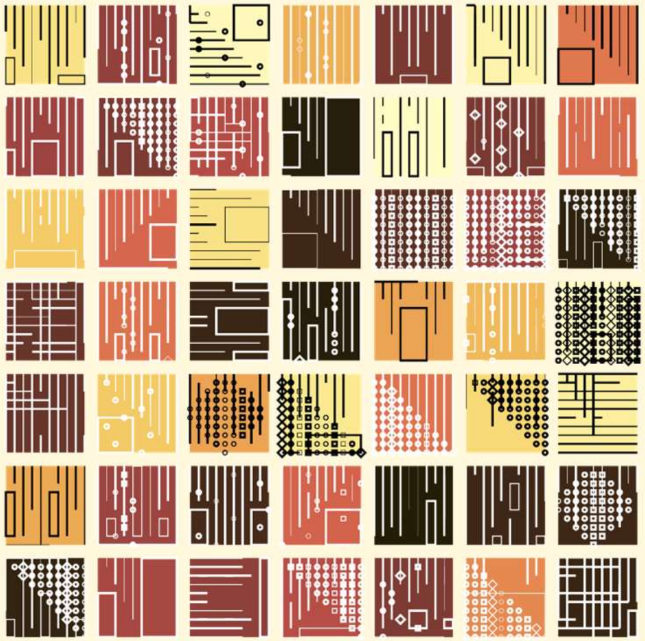






XQDoc

- Literate programming for XQuery
- Special comments
- Special markers



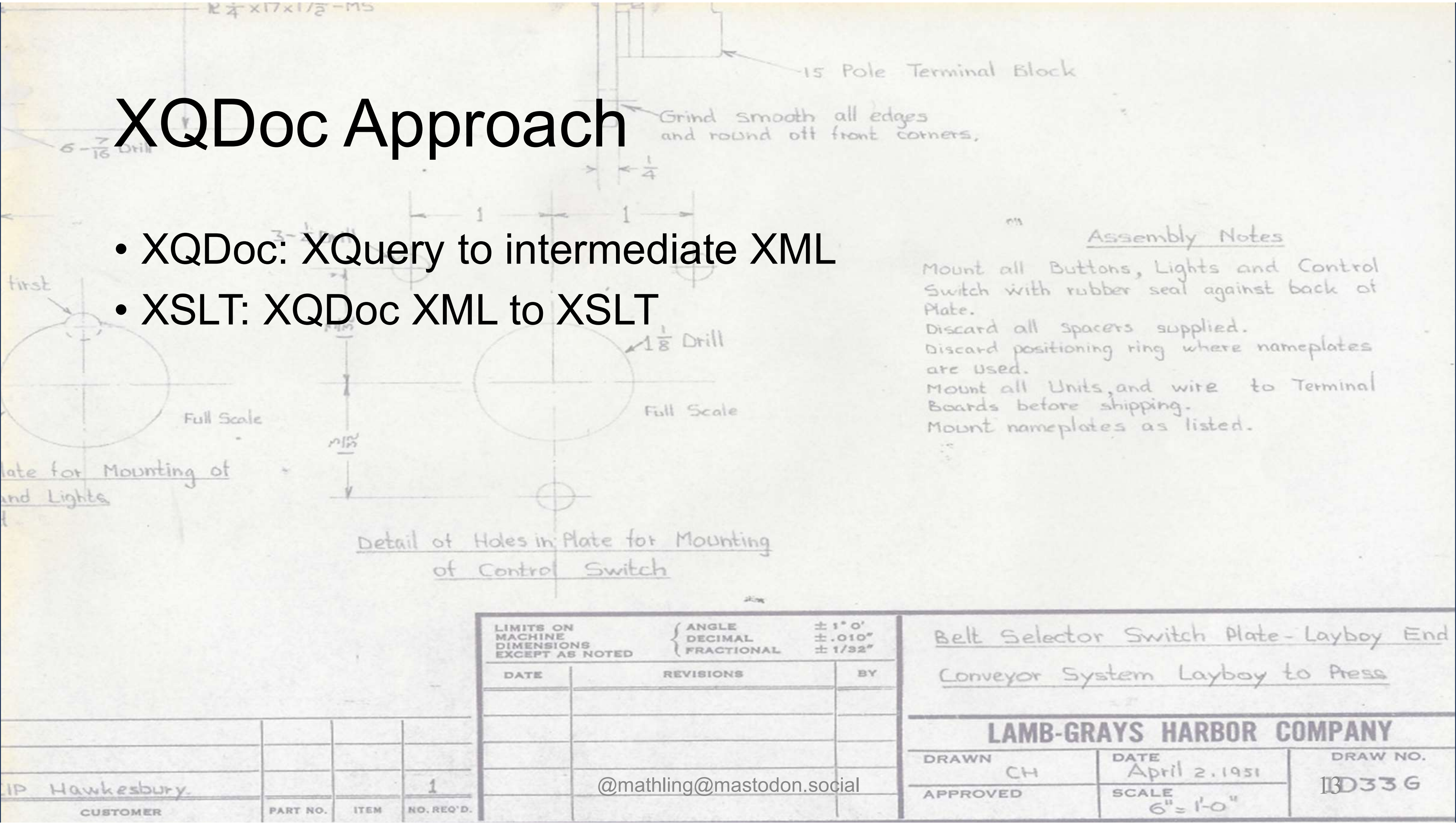
@mathling@mastodon.social

```xml
<xqdoc:xqdoc xmlns:xqdoc="http://www.xqdoc.org/1.0">
  <xqdoc:module type="library">
    <xqdoc:uri>http://mathling.com/core/utilities</xqdoc:uri>
    <xqdoc:name>this</xqdoc:name>
    <xqdoc:comment end="210" start="23">
      <xqdoc:description><![CDATA[
 Module with functions providing some basic utility operations.
 Copyright© Mary Holstege 2020-2023
 CC-BY (https://creativecommons.org/licenses/by/4.0/)]]></xqdoc:description>
        <xqdoc:since><![CDATA[March 2021]]></xqdoc:since>
    </xqdoc:comment>
  </xqdoc:module>
  <xqdoc:imports>
    <xqdoc:import location="../core/callable.xqy" prefix="callable" type="library">
      <xqdoc:uri>http://mathling.com/core/callable</xqdoc:uri>
        <xqdoc:body end="568" start="468" xml:space="preserve"><![CDATA[import module
namespace callable="http://mathling.com/core/callable"
       at "../core/callable.xqy"]]></xqdoc:body>
    </xqdoc:import>
```

```
<xqdoc:function>
  <xqdoc:comment end="6892" start="6781">
    <xqdoc:description><![CDATA[Is the number a prime?]]></xqdoc:description>
    <xqdoc:param><![CDATA[$i: positive integer]]></xqdoc:param>
    <xqdoc:return><![CDATA[whether it is prime]]></xqdoc:return>
  </xqdoc:comment>
  <xqdoc:name>is-prime</xqdoc:name>
  <xqdoc:parameters>
    <xqdoc:parameter><xqdoc:name>i</xqdoc:name><xqdoc:type>xs:integer</xqdoc:type></xqdoc:parameter>
  </xqdoc:parameters>
  <xqdoc:return><xqdoc:type>xs:boolean</xqdoc:type></xqdoc:return>
  <xqdoc:body end="7140" start="6894" xml:space="preserve"><![CDATA[
declare function this:is-prime($i as xs:integer) as xs:boolean
{
   ...(body here)...
}]]></xqdoc:body>
</xqdoc:function>
```

XQDoc Approach

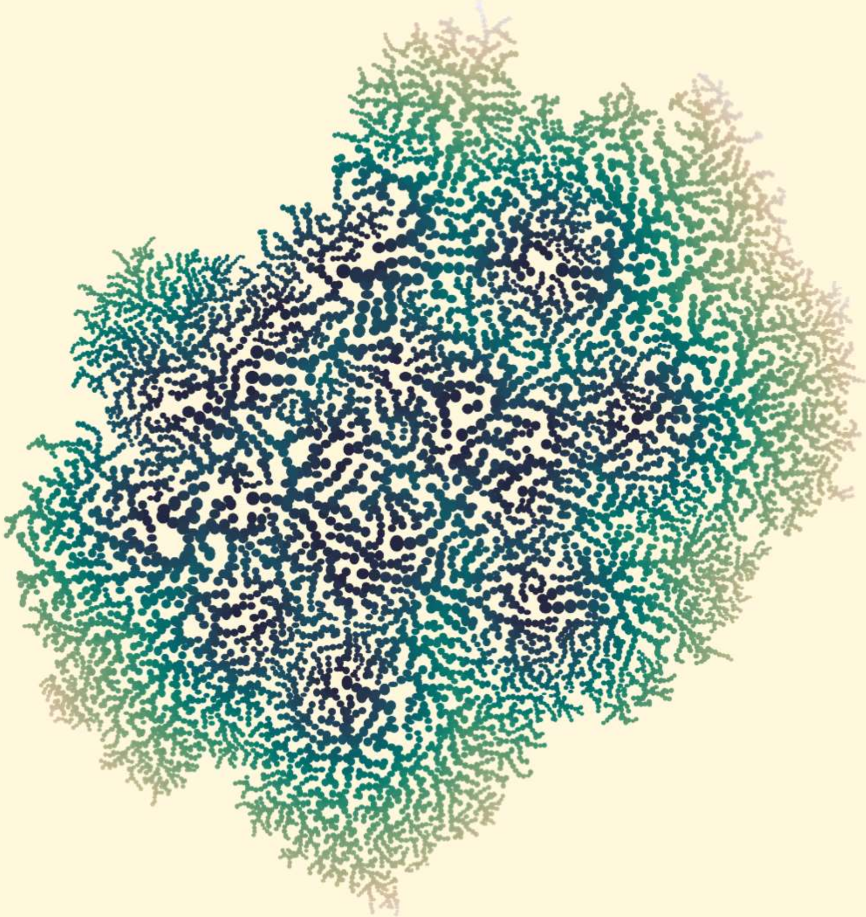
- XQDoc: XQuery to intermediate XML
- XSLT: XQDoc XML to XSLT


15 Pole Terminal Block
Grind smooth all edges and round off front corners,
6 - 7/16 Drill
1 1/8 Drill
Full Scale
Full Scale
Assembly Notes
Mount all Buttons, Lights and Control Switch with rubber seal against back of Plate.
Discard all spacers supplied.
Discard positioning ring where nameplates are used.
Mount all Units, and wire to Terminal Boards before shipping.
Mount nameplates as listed.
Detail of Holes in Plate for Mounting of Control Switch
LIMITS ON MACHINE DIMENSIONS EXCEPT AS NOTED
ANGLE ± 1° 0'
DECIMAL ± .010"
FRACTIONAL ± 1/32"
DATE
REVISIONS
BY
Belt Selector Switch Plate - Layboy End
Conveyor System Layboy to Press
LAMB-GRAYS HARBOR COMPANY
DRAWN CH
DATE April 2, 1951
DRAW NO. D33G
APPROVED
SCALE 6" = 1'-0"
Hawkesbury
CUSTOMER
PART NO.
ITEM
1
NO. REQ'D.

@mathling@mastodon.social

XQuery as XSLT

- Share more widely
- Interactive art with Saxon-JS



@mathling@mastodon.social

XQDoc Approach: XQuery as XSLT

- XQDoc: XQuery to intermediate XML
- XSLT: XQDoc XML to XSLT package
- XQuery expressions as XPath expressions

Function/variable bodies

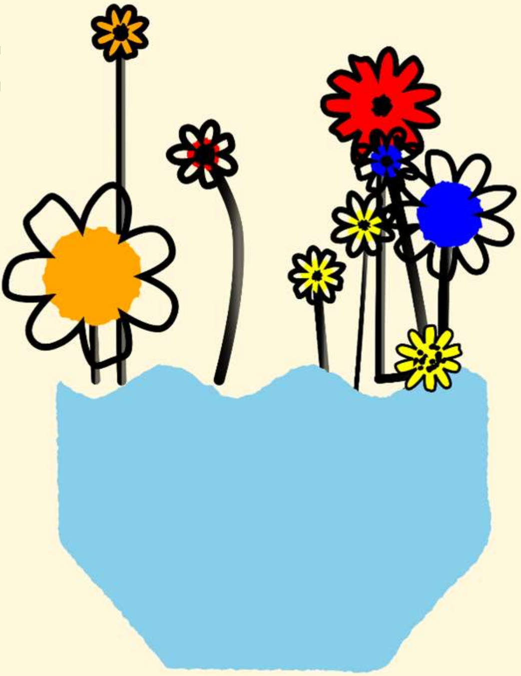
```
<xsl:function name="this:map-invert" as="map(*)">
  <xsl:param name="map" as="map(*)"/>
  <xsl:sequence select="&#xA;  map:merge(&#xA;    for $submap in
this:map-deconstruct($map)&#xA;    for $entry in
$submap=&gt;this:map-entries()&#xA;    let $new-key :=&#xA;
typeswitch($entry)&#xA;      case xs:anyAtomicType return
$entry&#xA;      default return this:quote($entry)&#xA;    return
(&#xA;      map {&#xA;        $new-key: $submap=&gt;map:keys()
&#xA;}&#xA;    )&#xA;  )"/>
</xsl:function>
```

@mathling@mastodon.social

Fixup: format junk

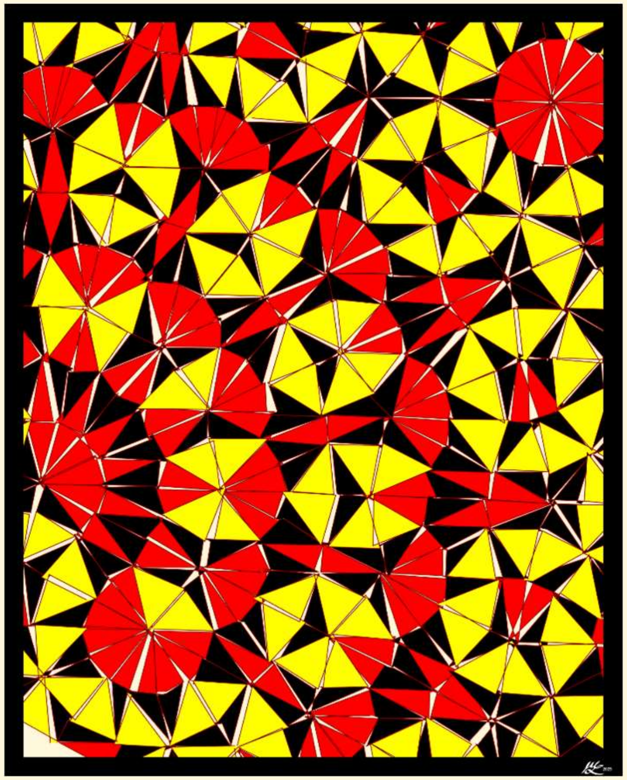
- Character map for &#xA; and &gt;
- Saxon single-quote for &quot;

- Automatic



Function/variable bodies

```
<xsl:function name='this:map-invert' as='map(*)'>
  <xsl:param name='map' as='map(*)'/>
  <xsl:sequence select='
  map:merge(
    for $submap in this:map-deconstruct($map)
    for $entry in $submap=>this:map-entries()
    let $new-key :=
      typeswitch($entry)
      case xs:anyAtomicType return $entry
      default return this:quote($entry)
    return (
      map {
        $new-key: $submap=>map:keys()
      }
    )
  )'/>
</xsl:function>
```



@mathling@mastodon.social

Corrections required

Issue	**Fix**
Multiple `let` or `for` clauses	Sprinkle `return` like fairy-dust
`order by`	Wrap in `sort()`
`where` clause	`if-then-else` in body

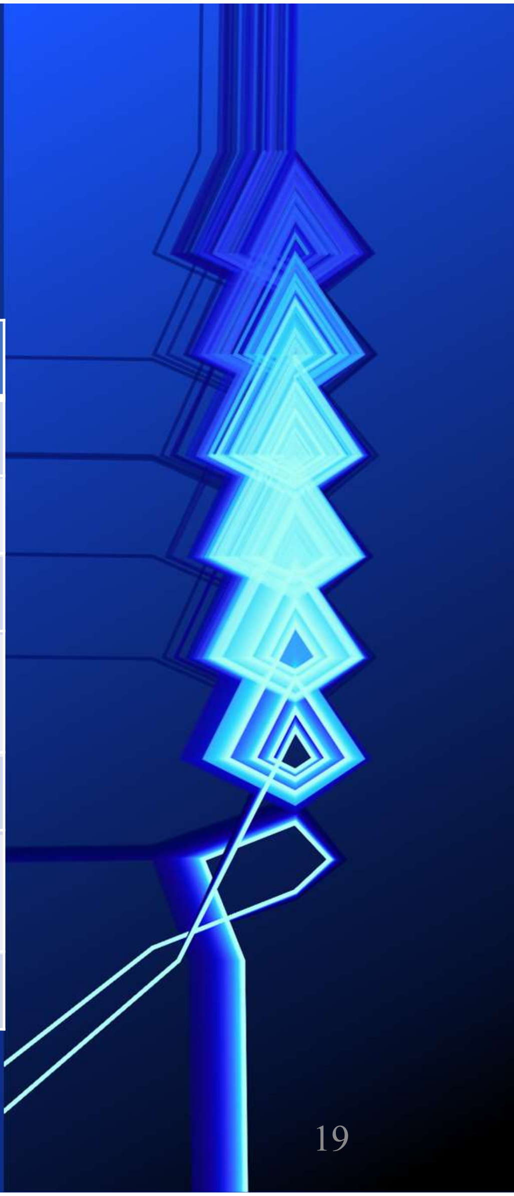
`for $x at $i in $seq`	`for $i in count($seq) return`
	`let $x := $seq[$i]`

`let $x as xs:integer`	Drop `as` clause

`switch`	`if…else if…else if…else`
	`if…else`

`typeswitch`	Same, plus `instance of`

Fairly mechanical



Corrected function

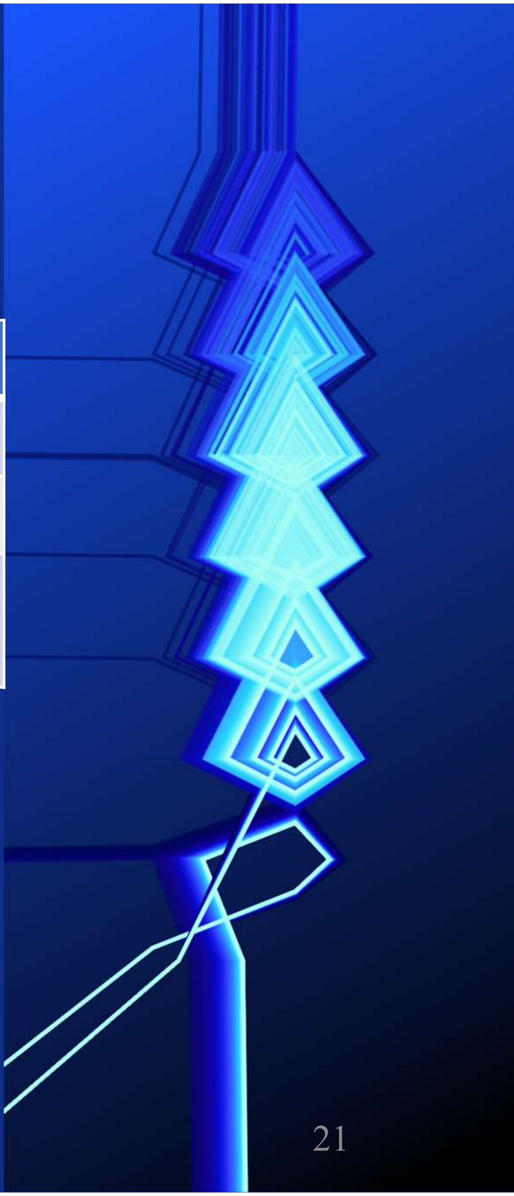
```
<xsl:function name='this:map-invert' as='map(*)'>
  <xsl:param name='map' as='map(*)'/>
  <xsl:sequence select='
  map:merge(
    for $submap in this:map-deconstruct($map) return
    for $entry in $submap=>this:map-entries() return
    let $new-key :=
      if ($entry instance of xs:anyAtomicType) then $entry
      else this:quote($entry)
    return (
      map {
        $new-key: $submap=>map:keys()
      }
    )
  )'/>
</xsl:function>
```

Corrections required

Issue	**Fix**
Node construction	Convert to XSLT
`try/catch`	Convert to XSLT
Dynamic function item annotations with eval*	Complete rework of APIs (ouch!); wrappers

Painful
My use of node construction and try/catch fairly limited

*For rich metadata capture



Fix up node construction

```
for $node in $nodes return typeswitch ($node)
case element(svg:feDisplacementMap) return
  element {node-name($node)} {
    $node/(@* except (@scale)),
    attribute scale {
      util:decimal($node/@scale * $rescale, 1)
    },
    this:replace-scale($node/*, $rescale)
  }
case element() return
  element {node-name($node)} {
    $node/@*,
    this:replace-scale($node/*, $rescale)
  }
default return $node
```

```
<xsl:for-each select='$nodes'>
  <xsl:variable name='node' select='.' as='node()'/>
  <xsl:choose>
    <xsl:when test='$node instance of element(svg:feDisplacementMap)'>
      <xsl:element name='{node-name($node)}'>
        <xsl:copy-of select='$node/(@* except @scale)'/>
        <xsl:attribute name='scale' select=‘
           util:decimal($node/@scale * $rescale, 1)'/>
        <xsl:sequence select=‘
           this:replace-scale($node/*, $rescale)'/>
      </xsl:element>
     </xsl:when>
     <xsl:when test='$node instance of element()'>
      <xsl:element name='{node-name($node)}'>
        <xsl:copy-of select='$node/@*'/>
        <xsl:sequence select='this:replace-scale($node/*, $rescale)'/>
      </xsl:element>
    </xsl:when>
    <xsl:otherwise>
      <xsl:copy-of select='$node'/>
    </xsl:otherwise>
  </xsl:choose>
</xsl:for-each>
```

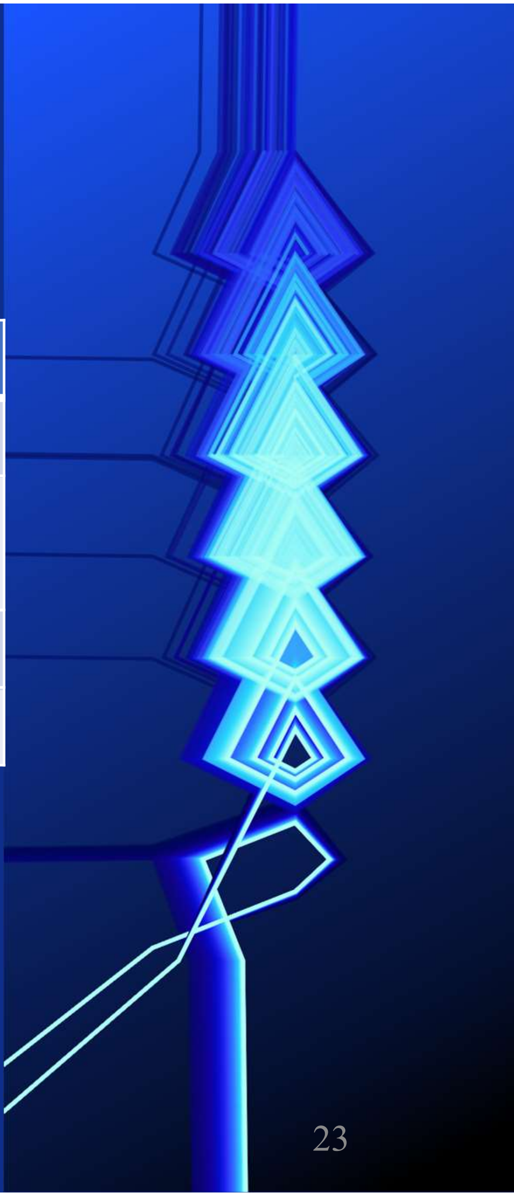
Corrections required

Issue	**Fix**
Grouping/windowing	Convert to XSLT

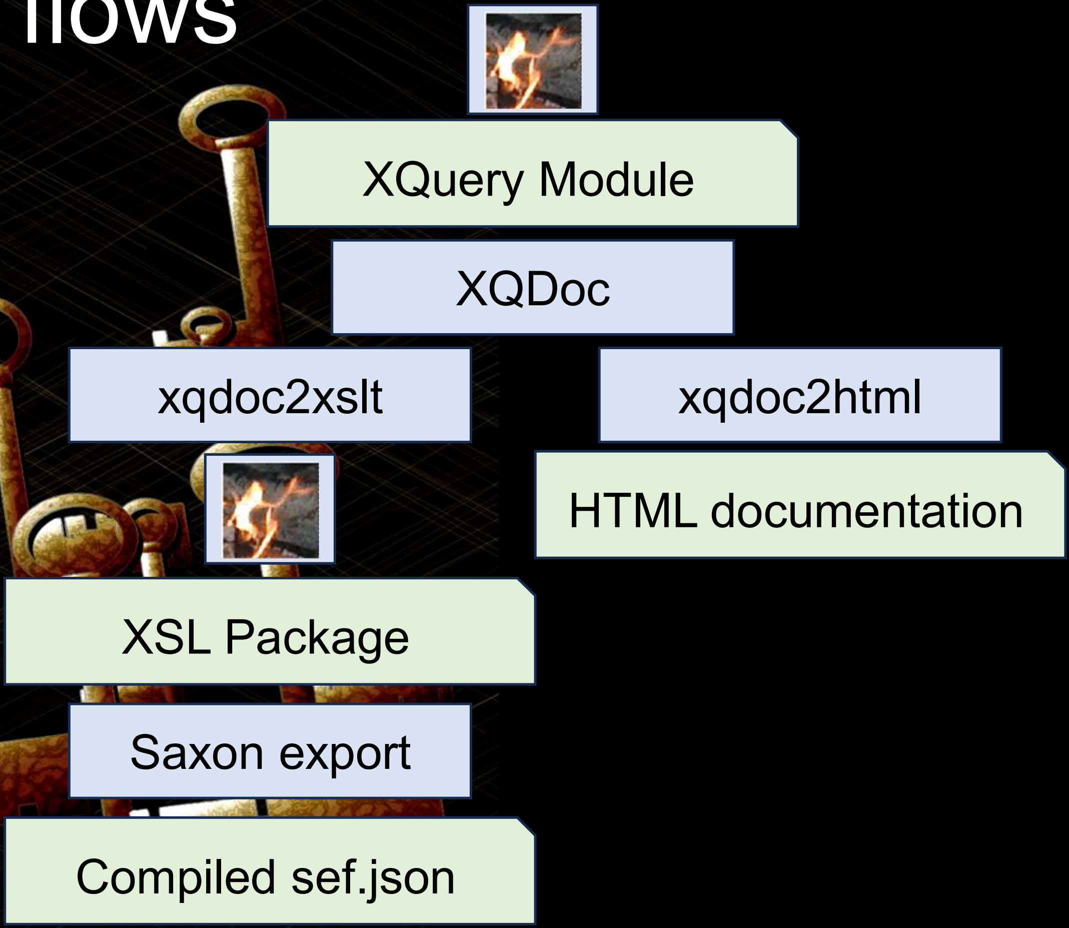
`ordered{}`
`unordered{}`
	Performance hint: drop

Other declarations	Some XSLT equivalents (see paper)
Crossing module	Rearchitect, play games with `use-when`

I didn’t need these



Key conversion flows



XQuery Module

XQDoc

xqdoc2xslt

xqdoc2html

HTML documentation

XSL Package

Saxon export

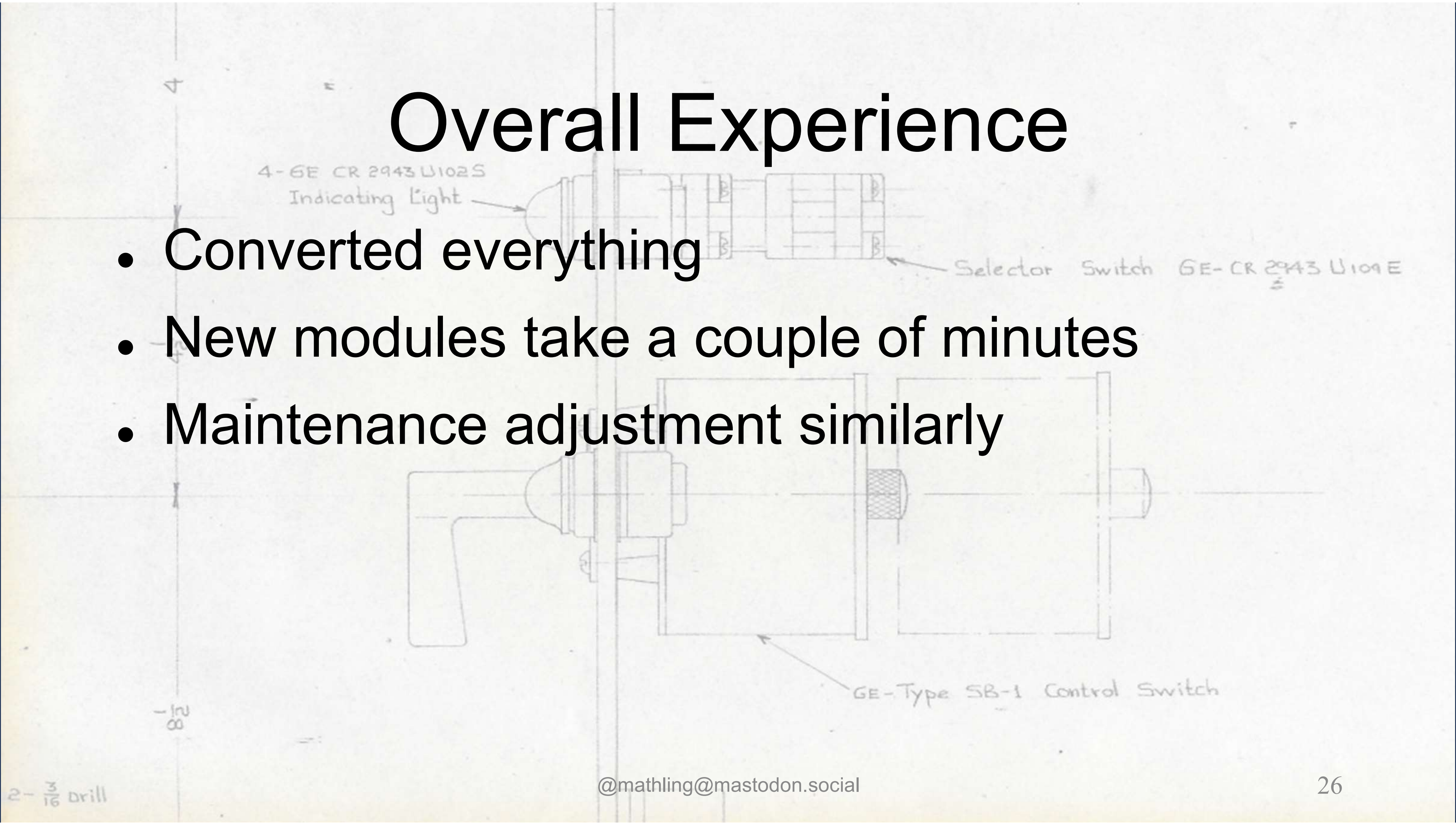
Compiled sef.json

Maintenance

- Differences of differences
  - Filter out “return” for better comparison
- Regenerate + re-edit

Overall Experience

- Converted everything
- New modules take a couple of minutes
- Maintenance adjustment similarly

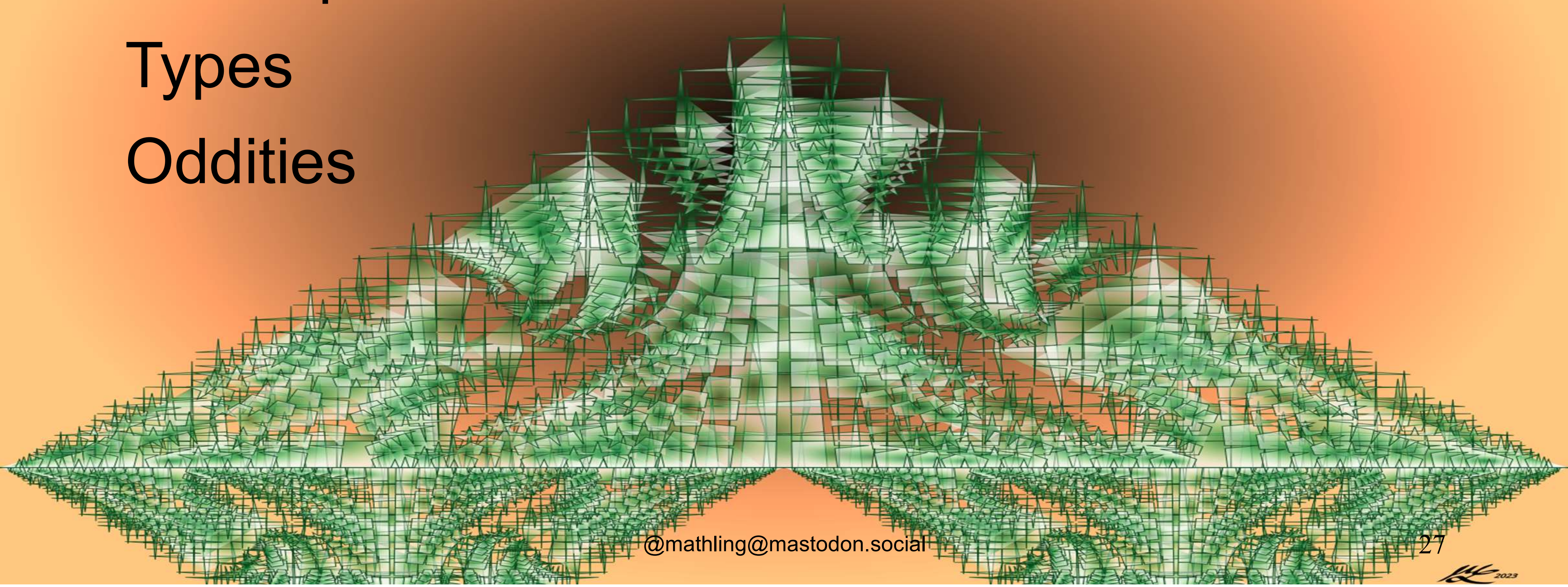


Danger, Will Robinson!

Assumptions

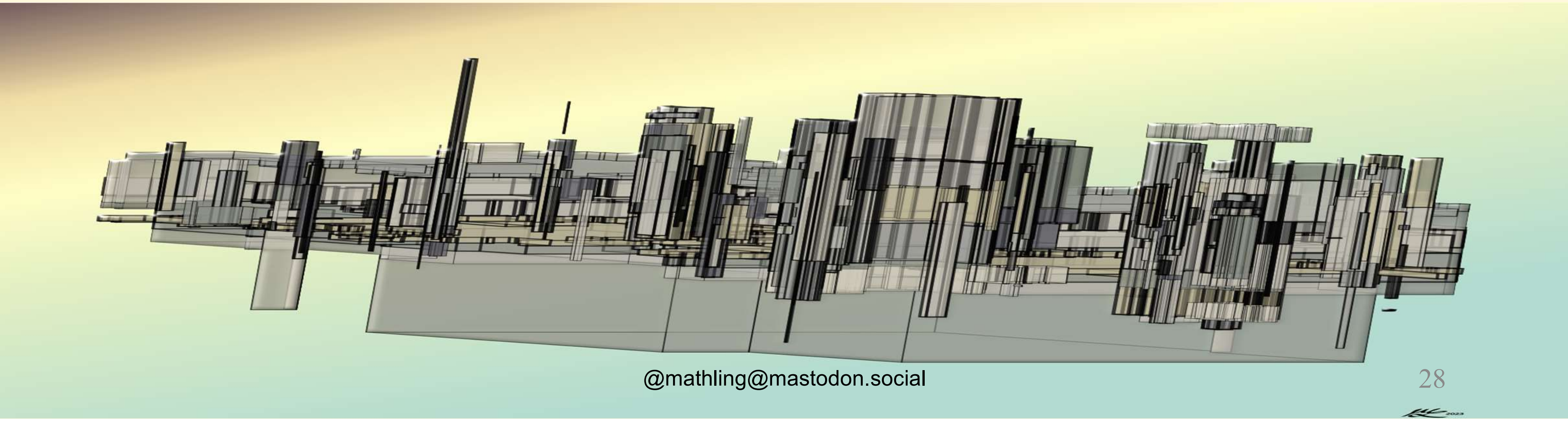
Types

Oddities



Unexpected Benefit: Better Code

- Full code review during manual step
- Three processors: unearth hidden assumptions
- “Compile” steps: early detection of errors



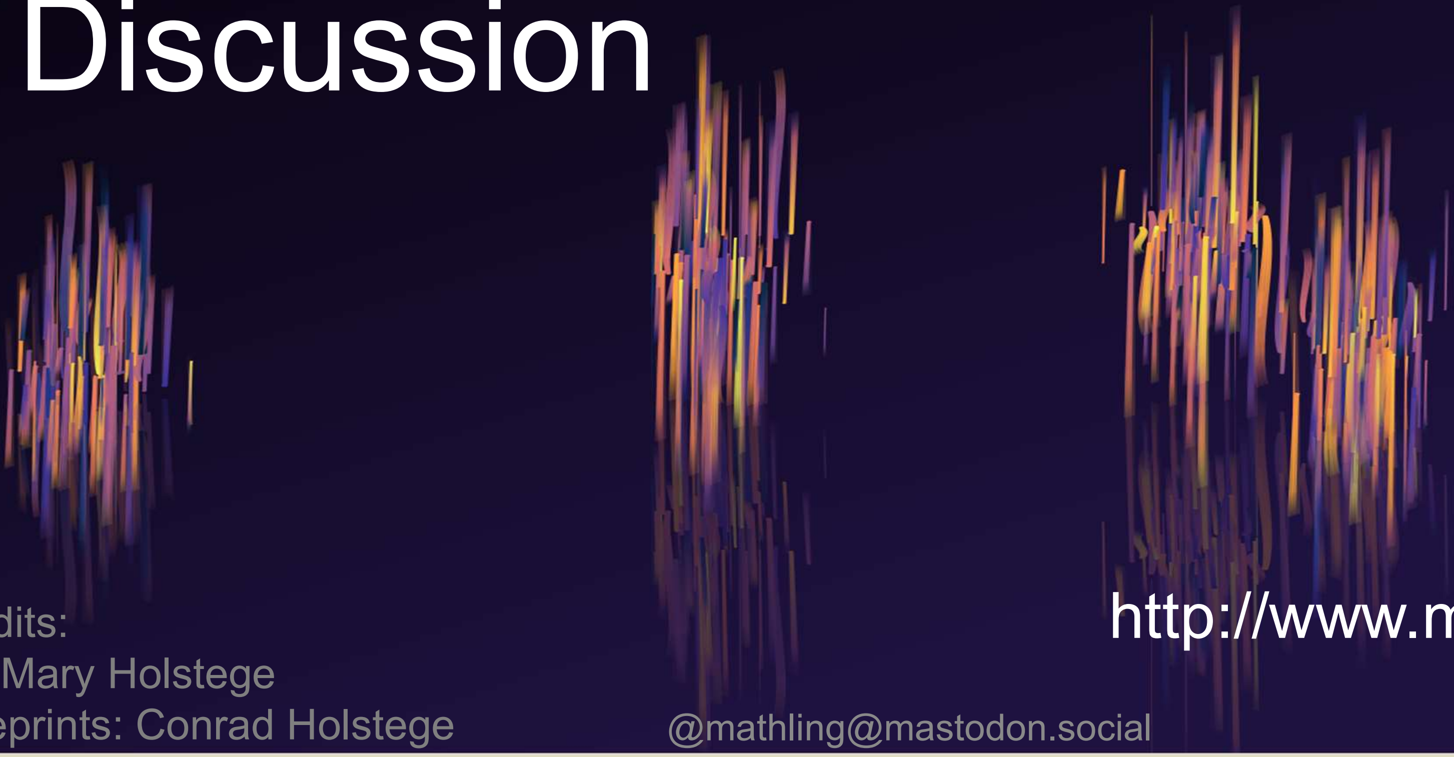
@mathling@mastodon.social

Pondering Tradeoffs

- One set of bugs or idiomatic usage?
- Easier conversion or easier development?
- Automation worth the effort?
    - Art or automation? Art or automation?

- My answers: one set of bugs, easier development, art over automation

Discussion



http://www.mathling.com/

Credits:
Art: Mary Holstege
Blueprints: Conrad Holstege

@mathling@mastodon.social