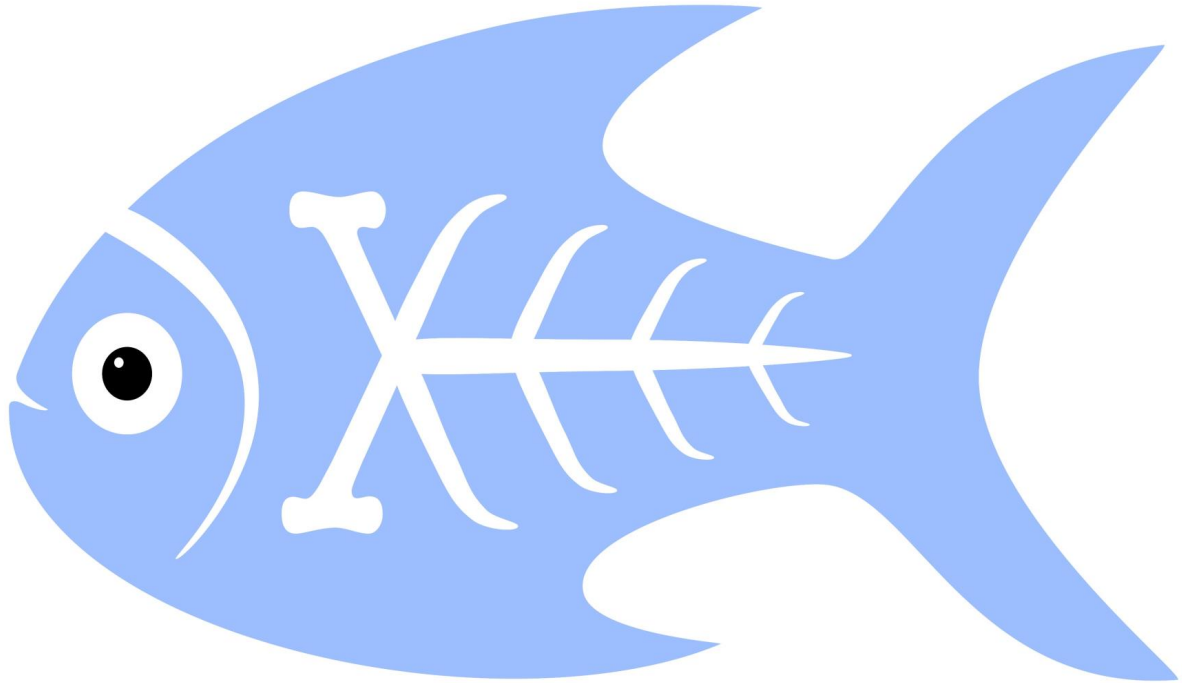
Excellent XProc 3.0



What's up?

- XProc is an XML based programming language for complex data processing - pipelining
- V1 (2010) turned out to be
  - hard to use and understand
  - verbose
- People that use it, use it extensively
- V1 became outdated with respect to underlying standards

What happened?

- V2 Initiative (non-XML) – not enough support
- V3 Initiative (2016) - W3C Community Group
  - Stay close to existing syntax
  - Make language more usable, understandable and concise
  - Update underlying standards
  - Allow other document types to flow through
  - Clean up loose ends
- Editors
  - Norman Walsh, Achim Berndzen, Gerrit Imsieke, Erik Siegel
- Meetings, proposals, discussions, …

Helicopter status

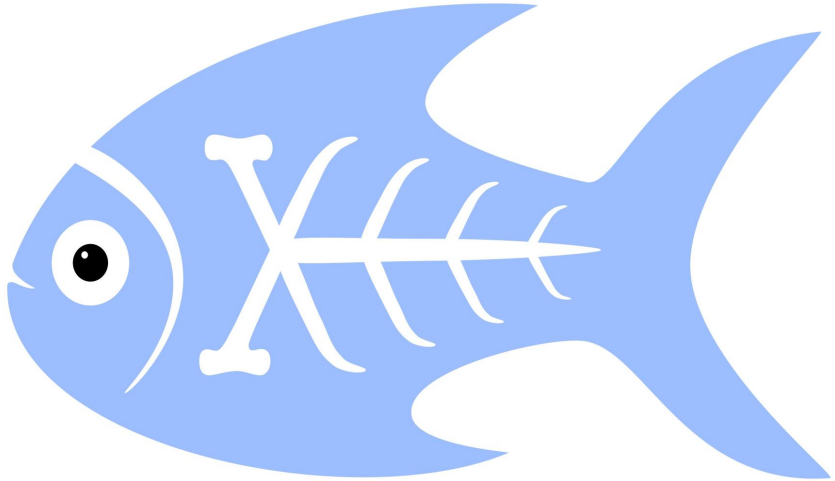
- Final call core spec
- Working on the steps
- End before end 2019
- In the making:
  - A specification (http://spec.xproc.org)
  - Two processor implementations (XML Calabash, MorganaXProc)
  - A programmer's reference book

And its name is…

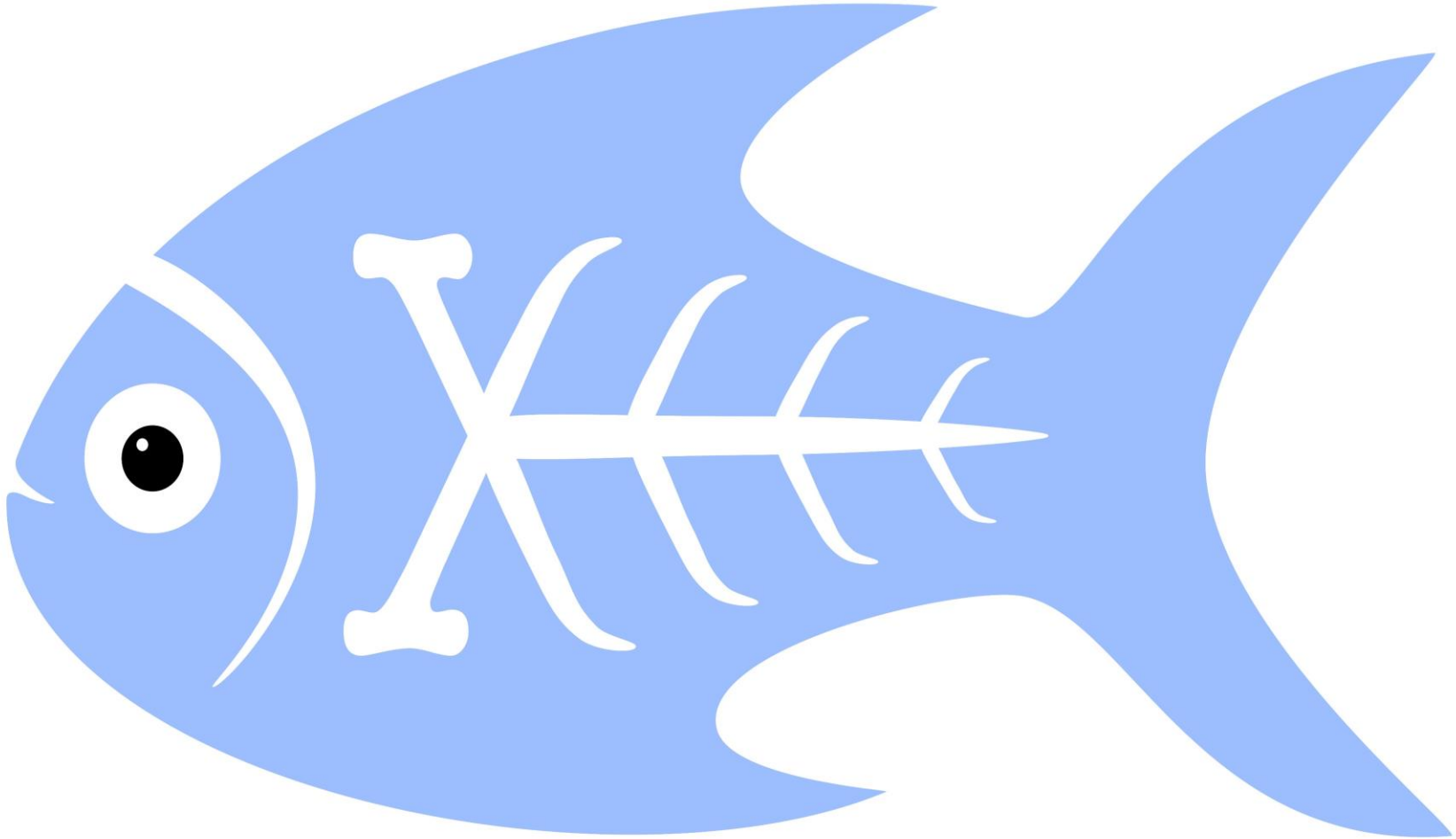
Kanava

Logo!

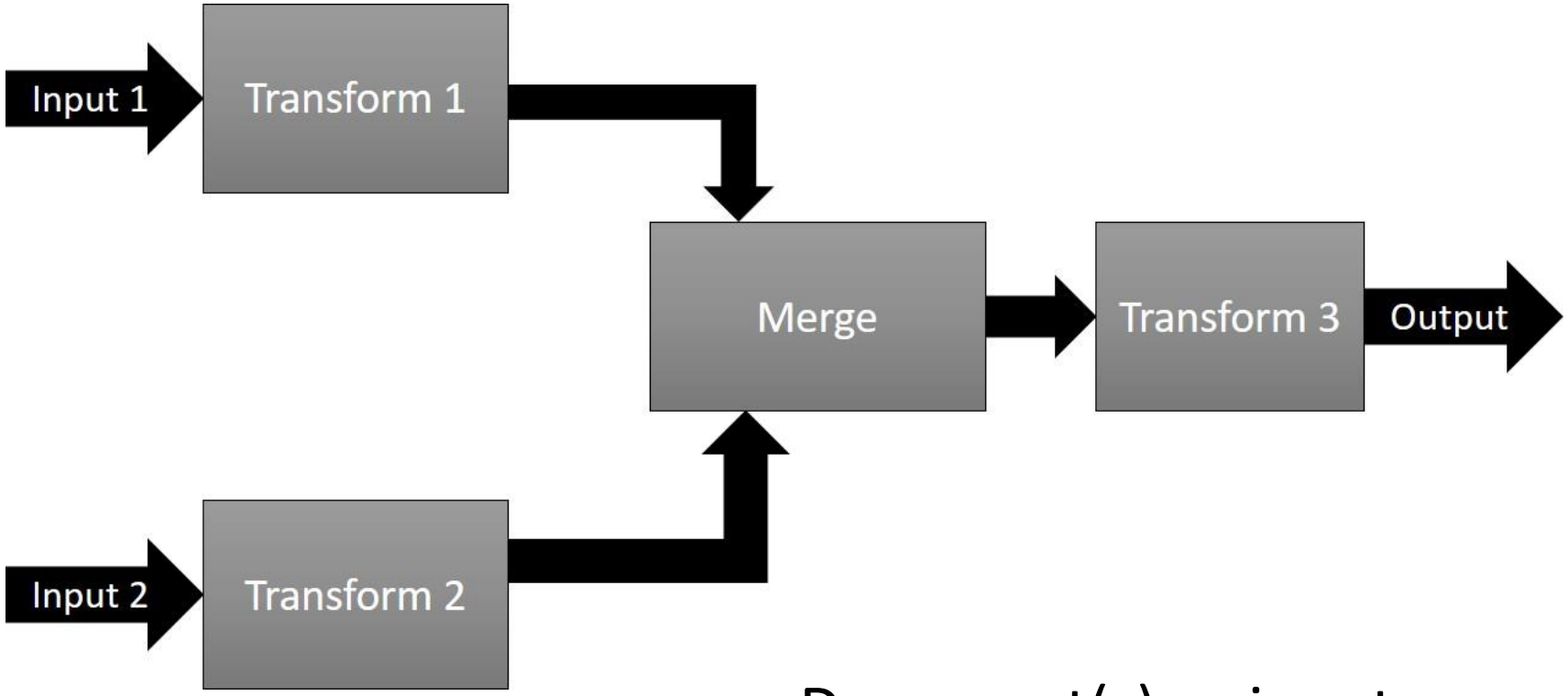
Thanks to Bethan Tovey



Crash Course XProc fundamentals

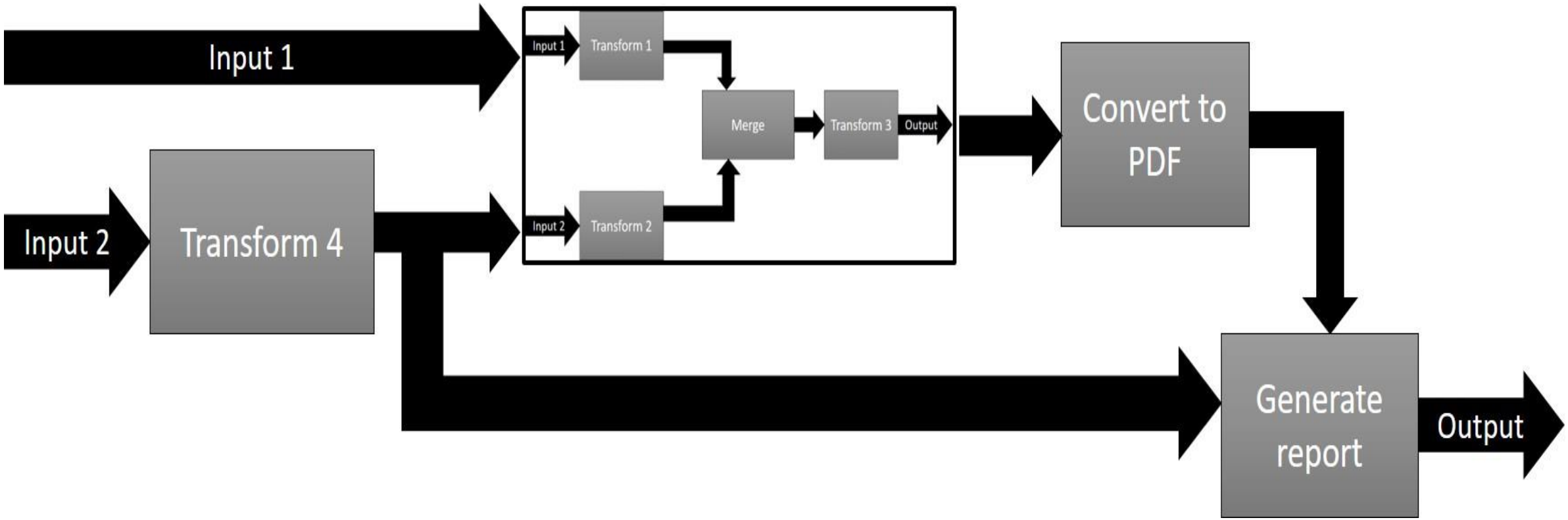


Pipelines, steps (1.0 & 3.0)

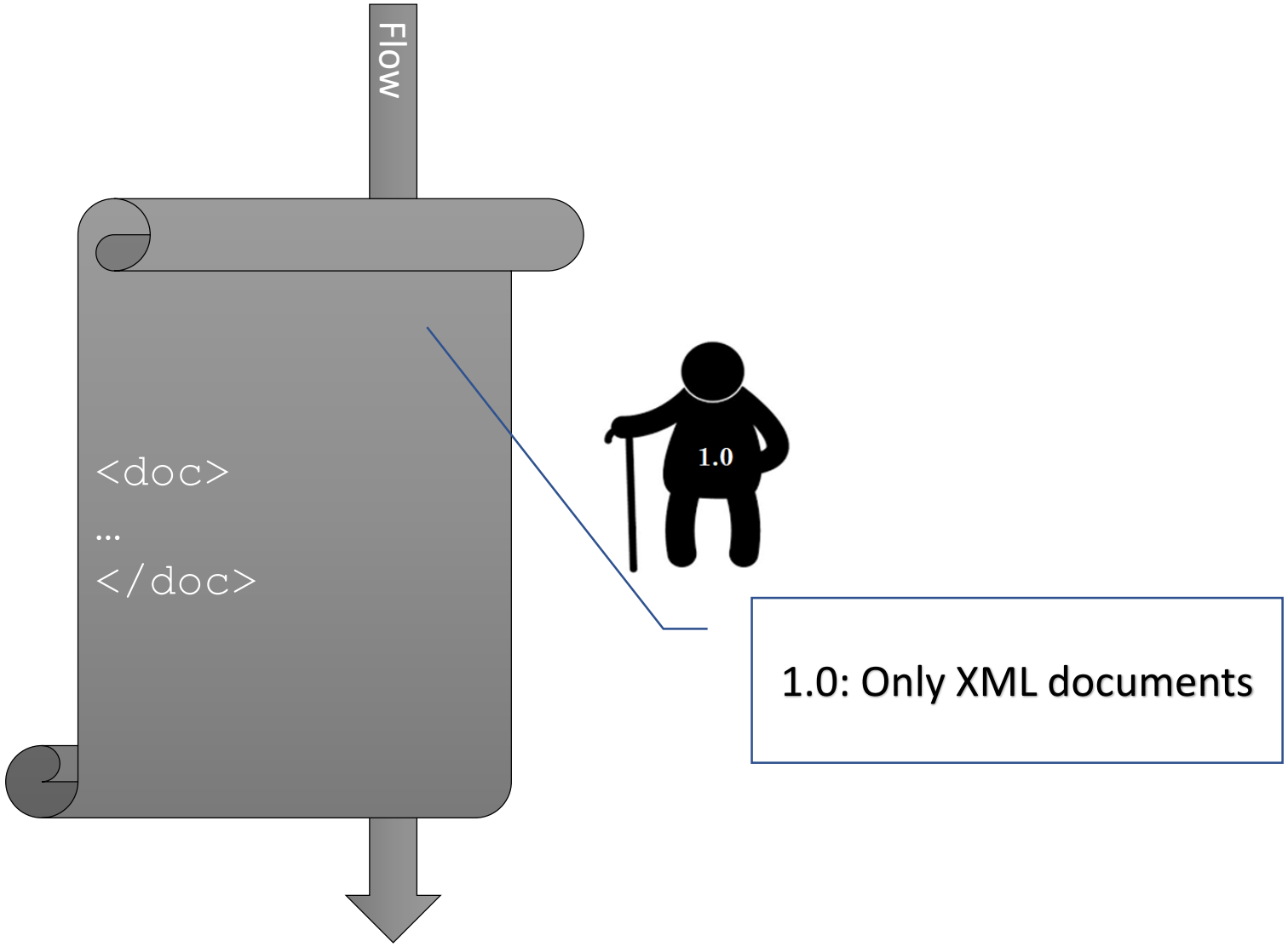


- Document(s) as input
- Process the data flowing through using steps
- Produce output(s)

Pipelines, steps (1.0 & 3.0)



Documents flowing through (1.0)

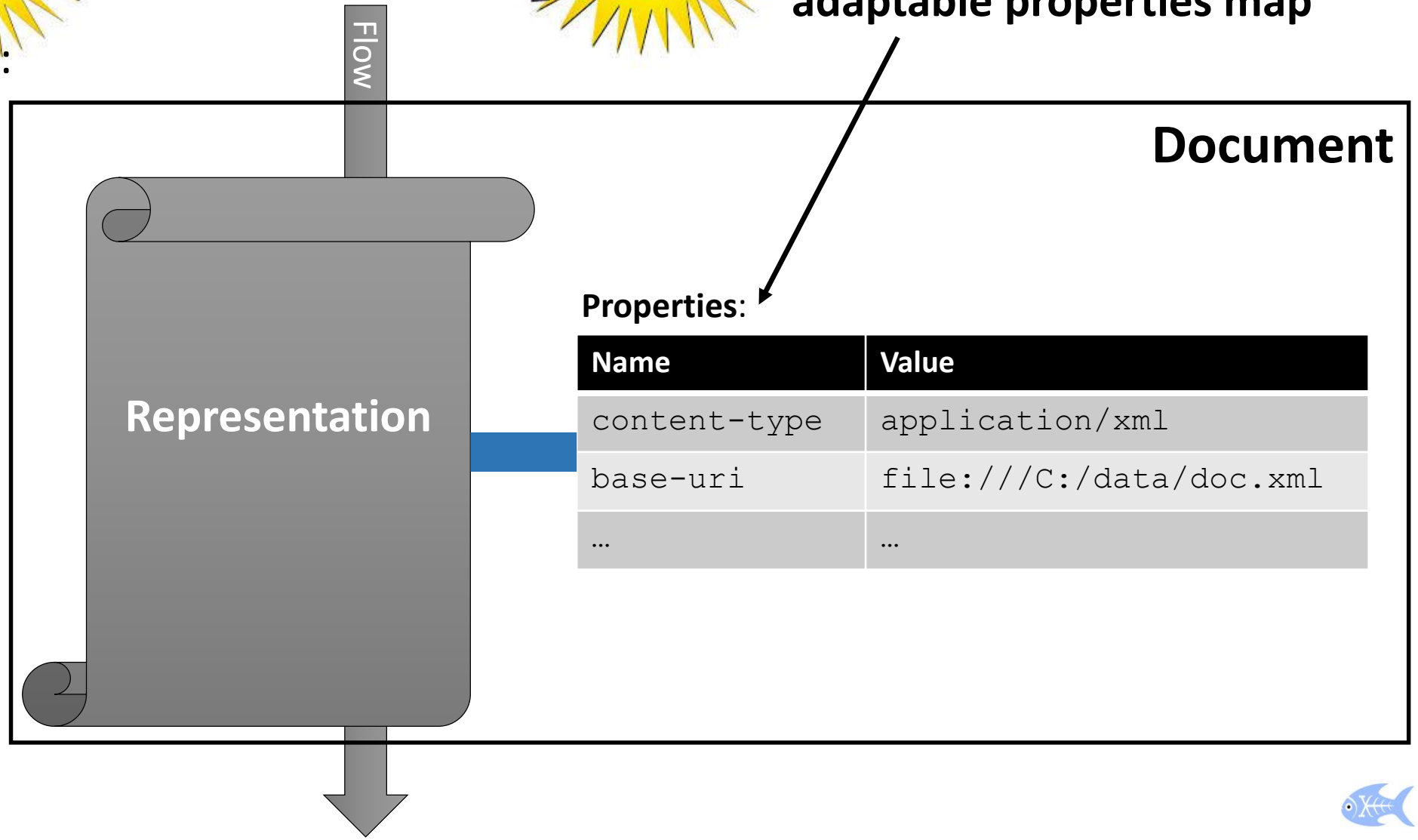


Documents flowing through (3.0)

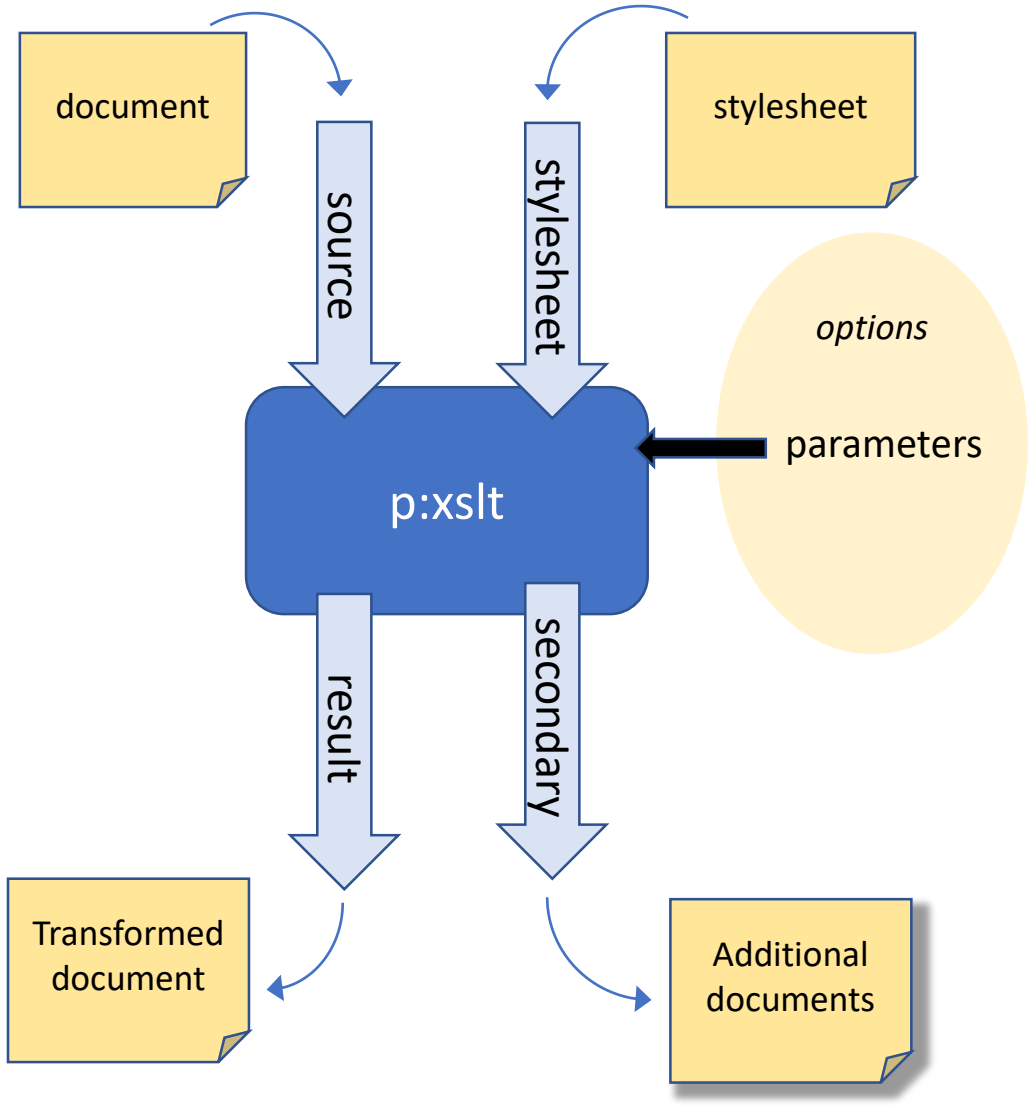
All documents carry an adaptable properties map

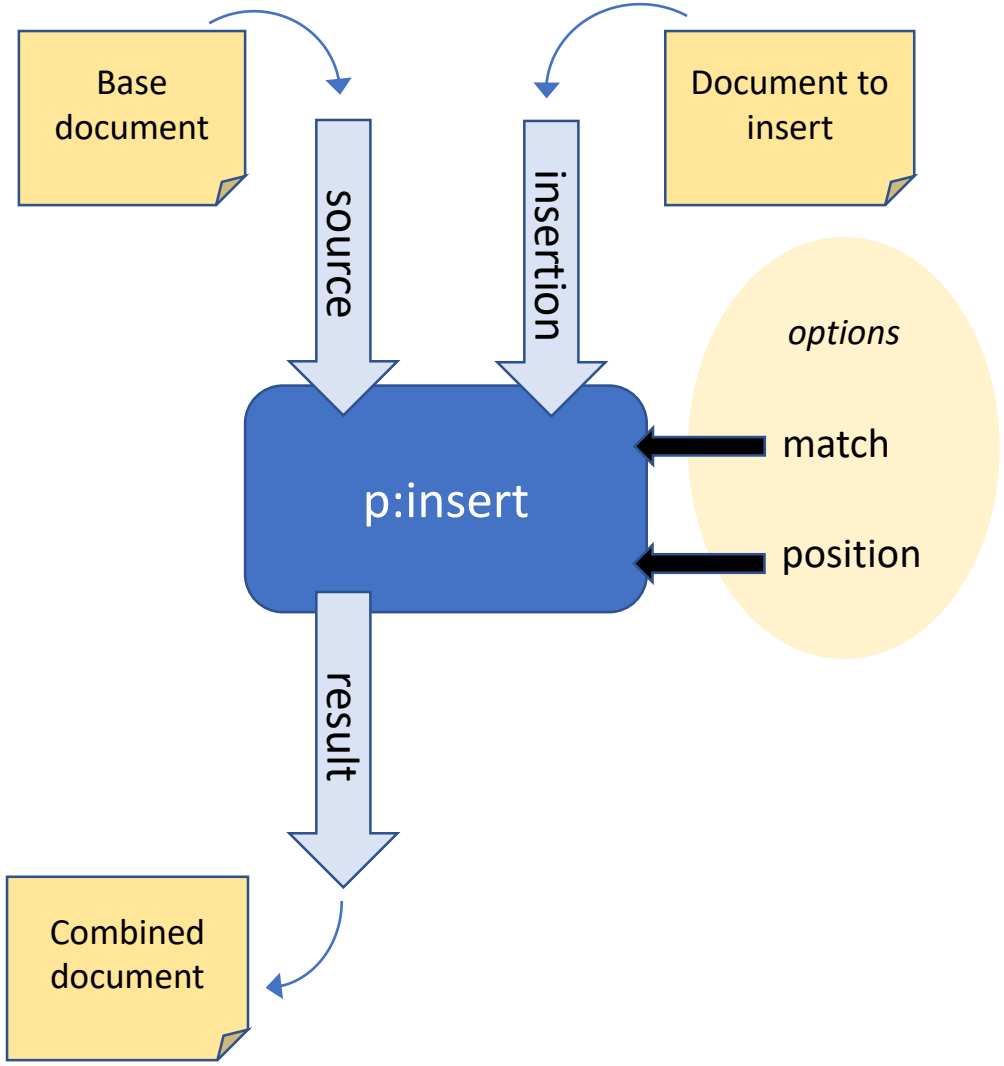
Native document types:

- XML
- HTML
- JSON
- Text
- Other

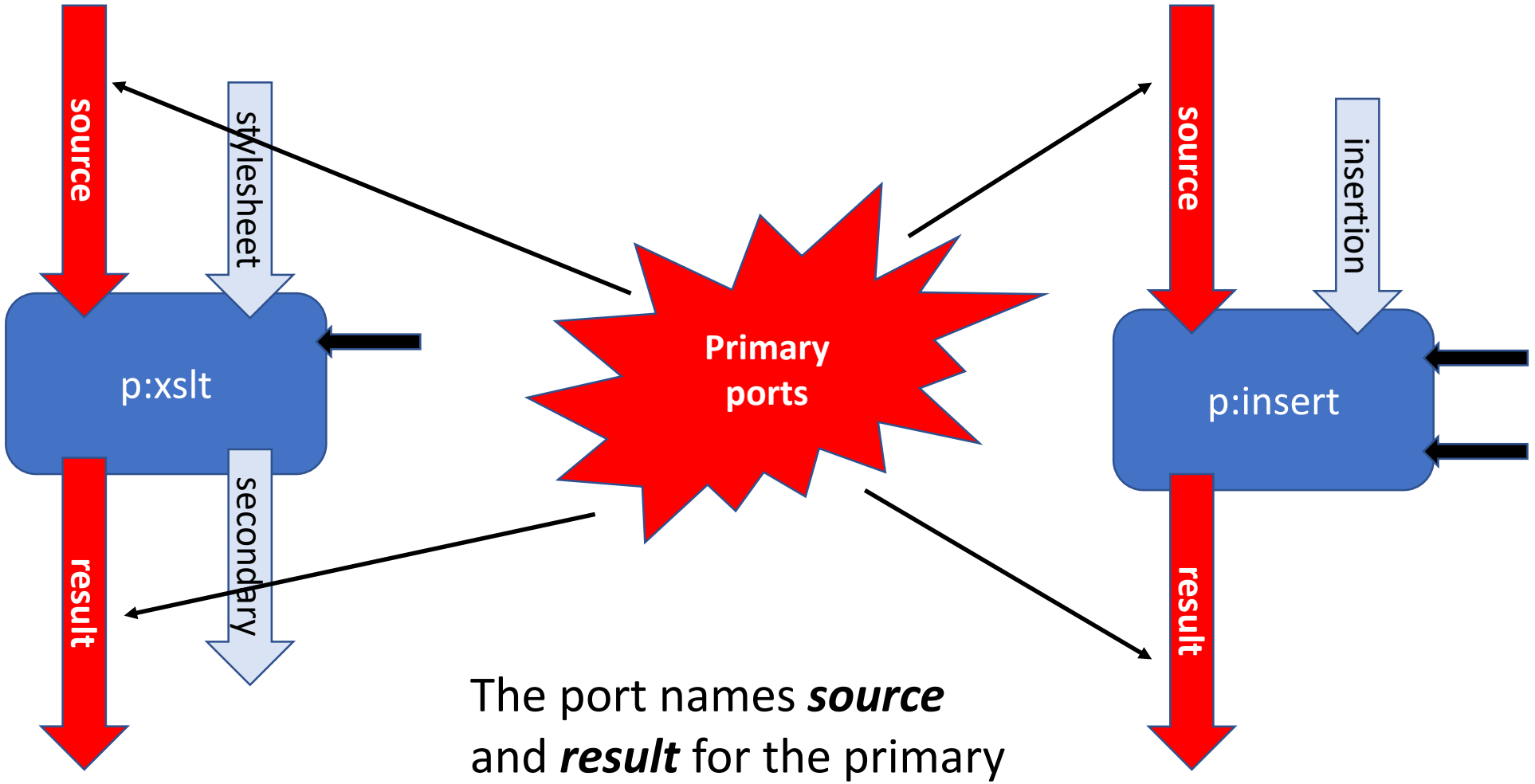


Steps, ports, options (1.0 & 3.0)



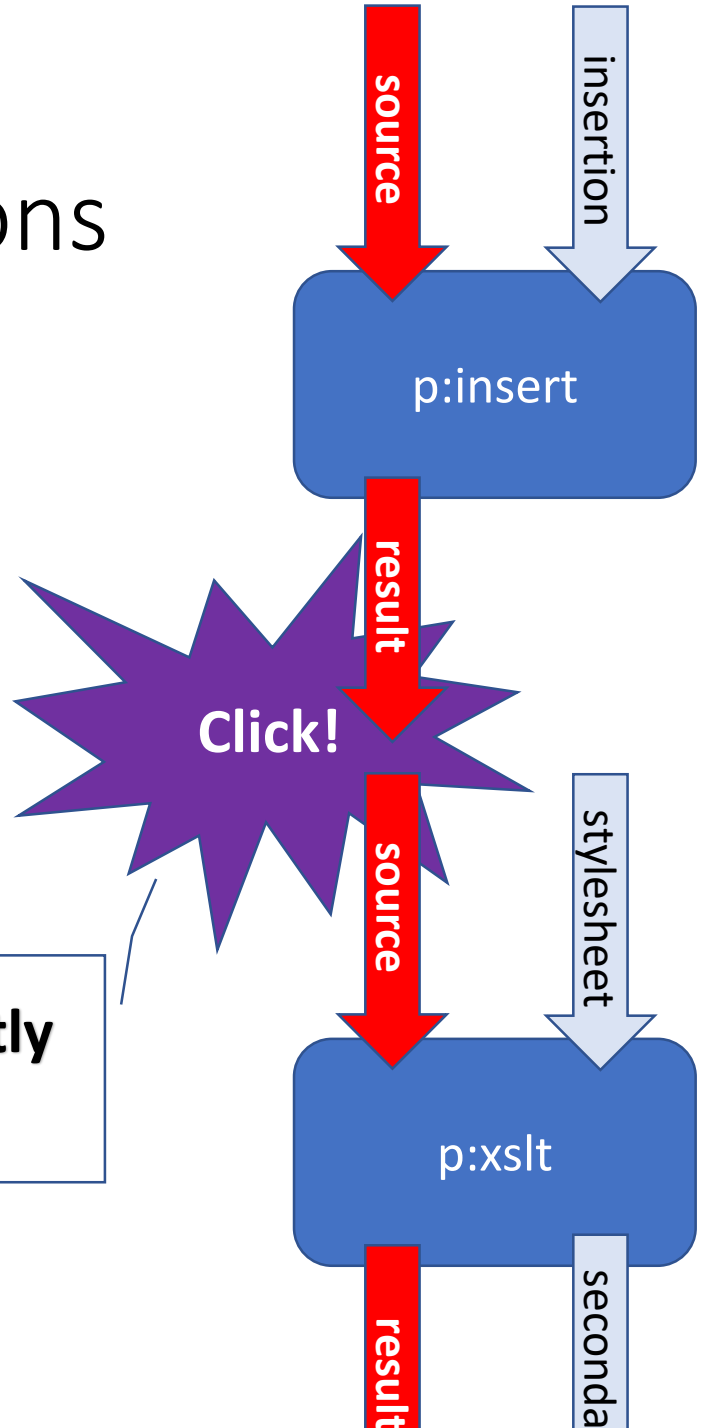


Primary ports (1.0 & 3.0)



The port names ***source*** and ***result*** for the primary ports are convention

Primary ports, implicit connections (1.0 & 3.0)



Primary ports implicitly connect

```
<p:insert …>
  …
</p:insert>

<p:xslt …>
  …
</p:xslt>
```

Primary ports, implicit connections (1.0 & 3.0)

Implicit connection of primary input port to first step

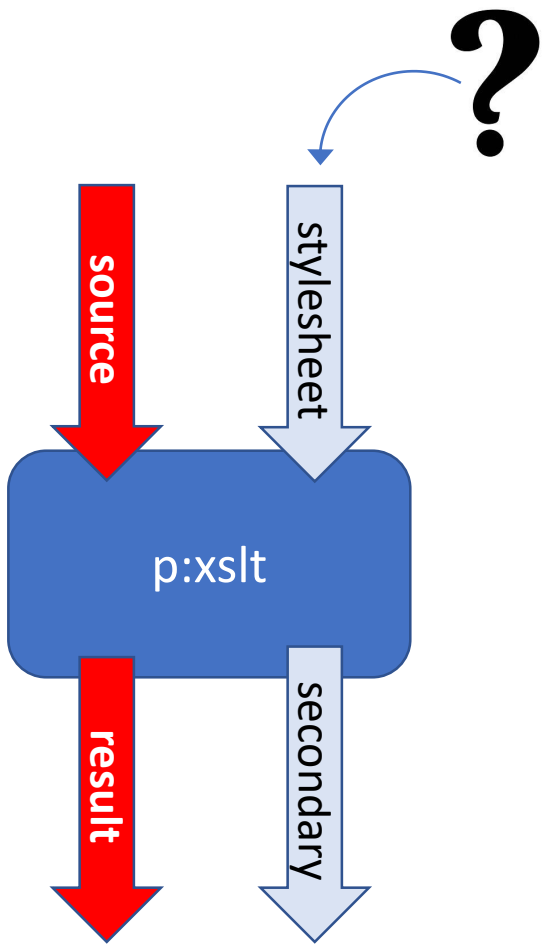
Implicit connection of steps

Implicit connection of last step to primary output port

```
<p:declare-step xmlns:p="http://www.w3.org/ns/xproc">

  <p:input port="source" primary="true"/>
  <p:output port="result" primary="true"/>

  <p:xslt>
    <p:with-input port="stylesheet" href="myxslt1.xsl"/>
  </p:xslt>

  <p:xslt>
    <p:with-input port="stylesheet" href="myxslt2.xsl"/>
  </p:xslt>

</p:declare-step>
```

Explicitly connecting ports (3.0)

```
<p:xslt>
  <p:with-input port="stylesheet">
    …
  </p:with-input>
</p:xslt>
```



<p:input> ➔ <p:with-input> NEW!

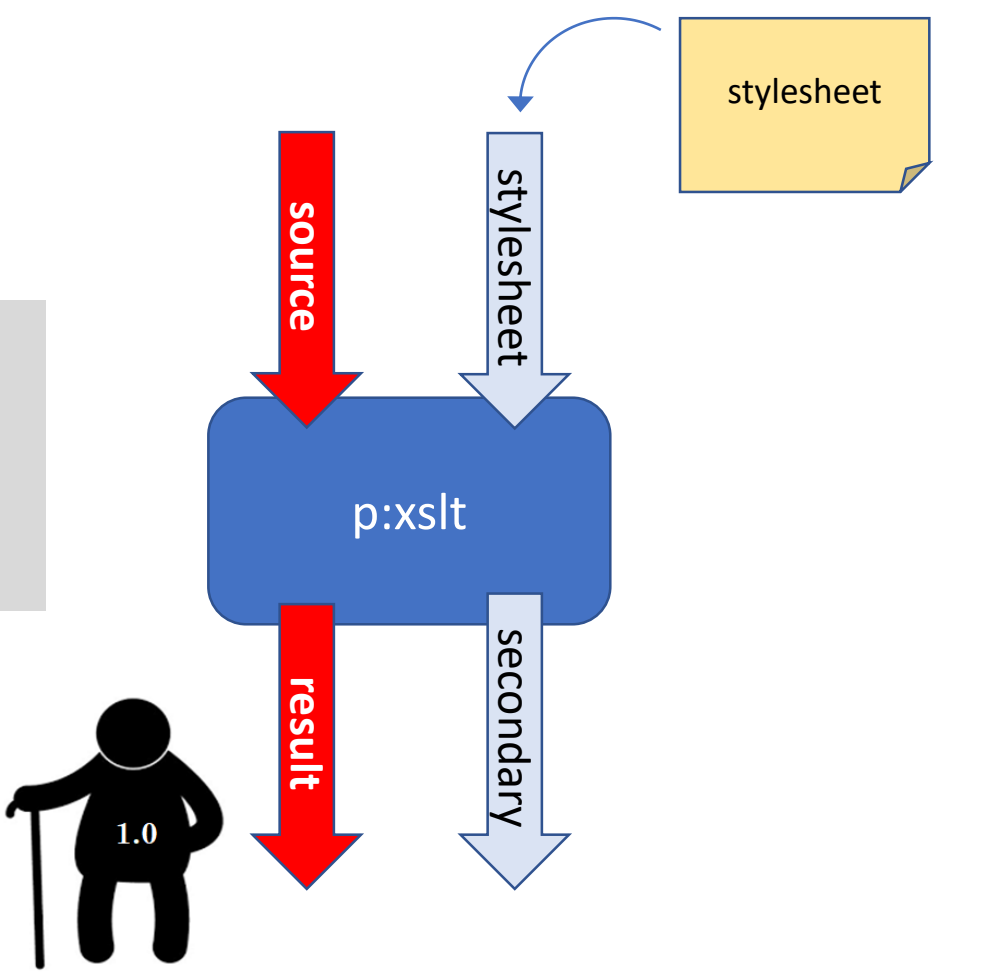
Rules:

- Input ports *must* be connected (or have a default)
- Output ports can be left dangling (*also primary output ports*) NEW!

Explicitly connecting ports – with an external document (1.0 & 3.0)

```
<p:xslt>
  <p:with-input port="stylesheet">
    <p:document href="/some/stylesheet.xsl"/>
  </p:with-input>
</p:xslt>
```

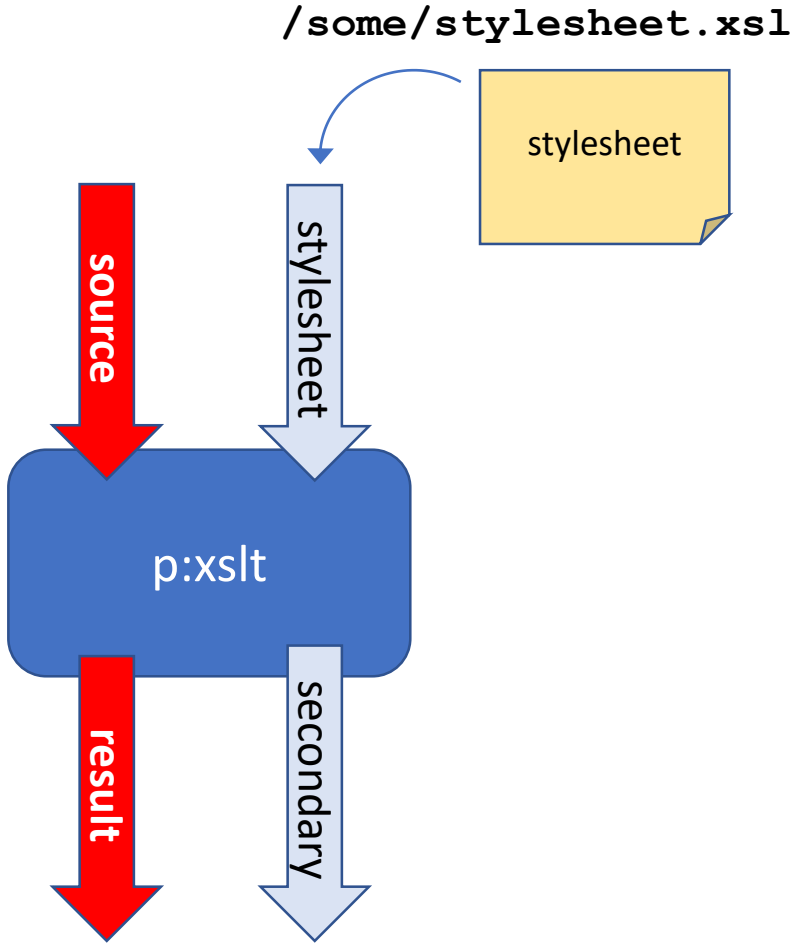
1.0: You *had* to use <p:document> and @href had to be a literal



Explicitly connecting ports (3.0)

```
<p:variable name="path" select="'/some'"/>

<p:xslt>
  <p:with-input port="stylesheet">
    <p:document href="{$path}/stylesheet.xsl"/>
  </p:with-input>
</p:xslt>
```

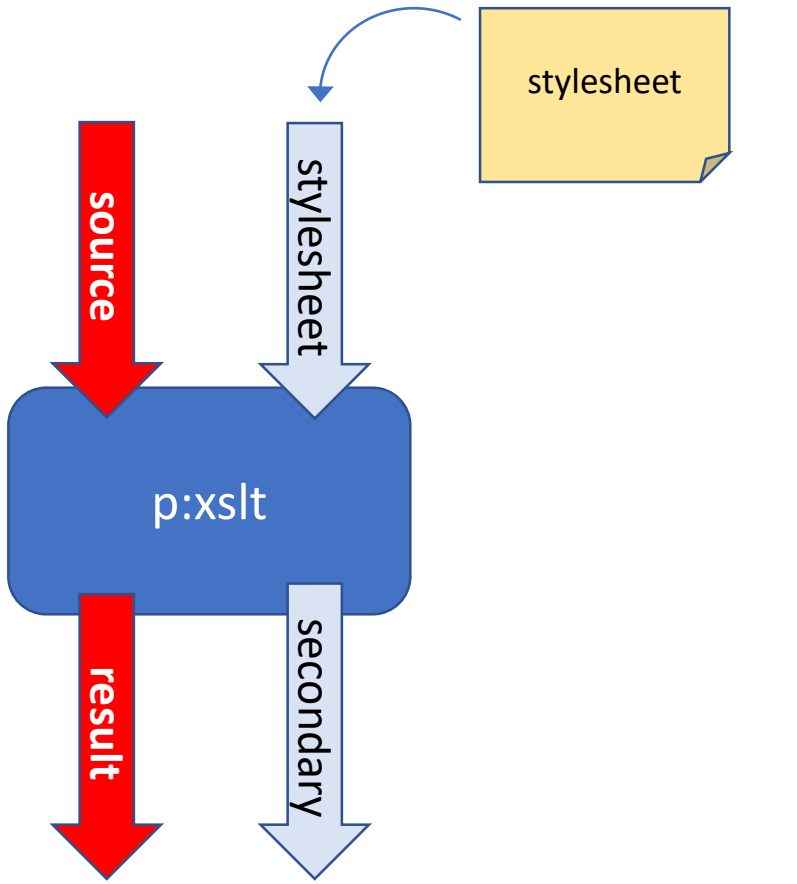
Attribute-Value Templates



Explicitly connecting ports (3.0)

```
<p:xslt>
  <p:with-input port="stylesheet"
     href="{$path}/stylesheet.xsl"/>
</p:xslt>
```

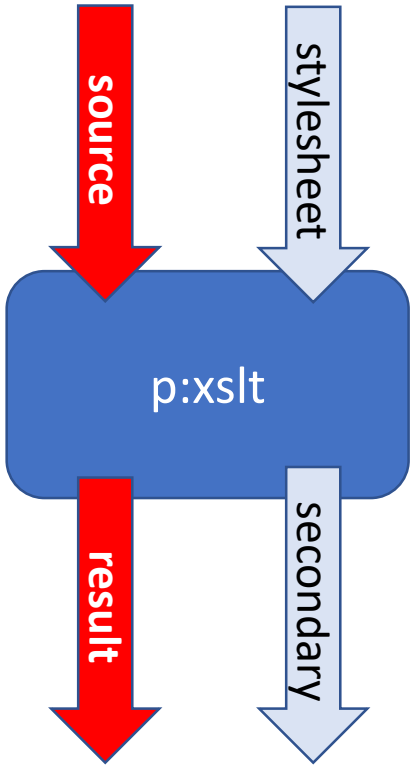
<p:document> can be shortened to a @href (just one example of lots of syntactic sugar shortcuts)



Explicitly connecting ports – to an inline document (1.0 & 3.0)

```
<p:xslt>
  <p:with-input port="stylesheet">
    <p:inline>
       <xsl:stylesheet …>
          …
       </xsl:stylesheet>
    </p:inline>
  </p:with-input>
</p:xslt>
```

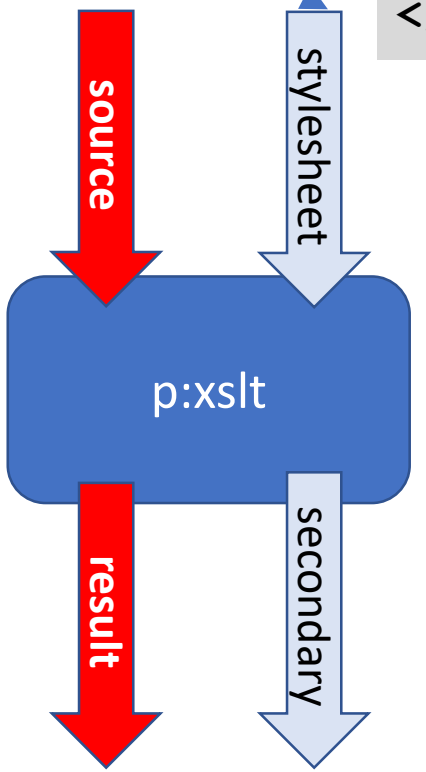
1.0: You *had* to use <p:inline>



Explicitly connecting ports – to an inline document (3.0)

```
<p:with-input port="source">
  <root date="{current-date()}">
   {$contents}
  </root>
</p:with-input>
```

```
<p:xslt>
  <p:with-input port="stylesheet">
    <xsl:stylesheet …>
          …
    </xsl:stylesheet>
  </p:with-input>
</p:xslt>
```

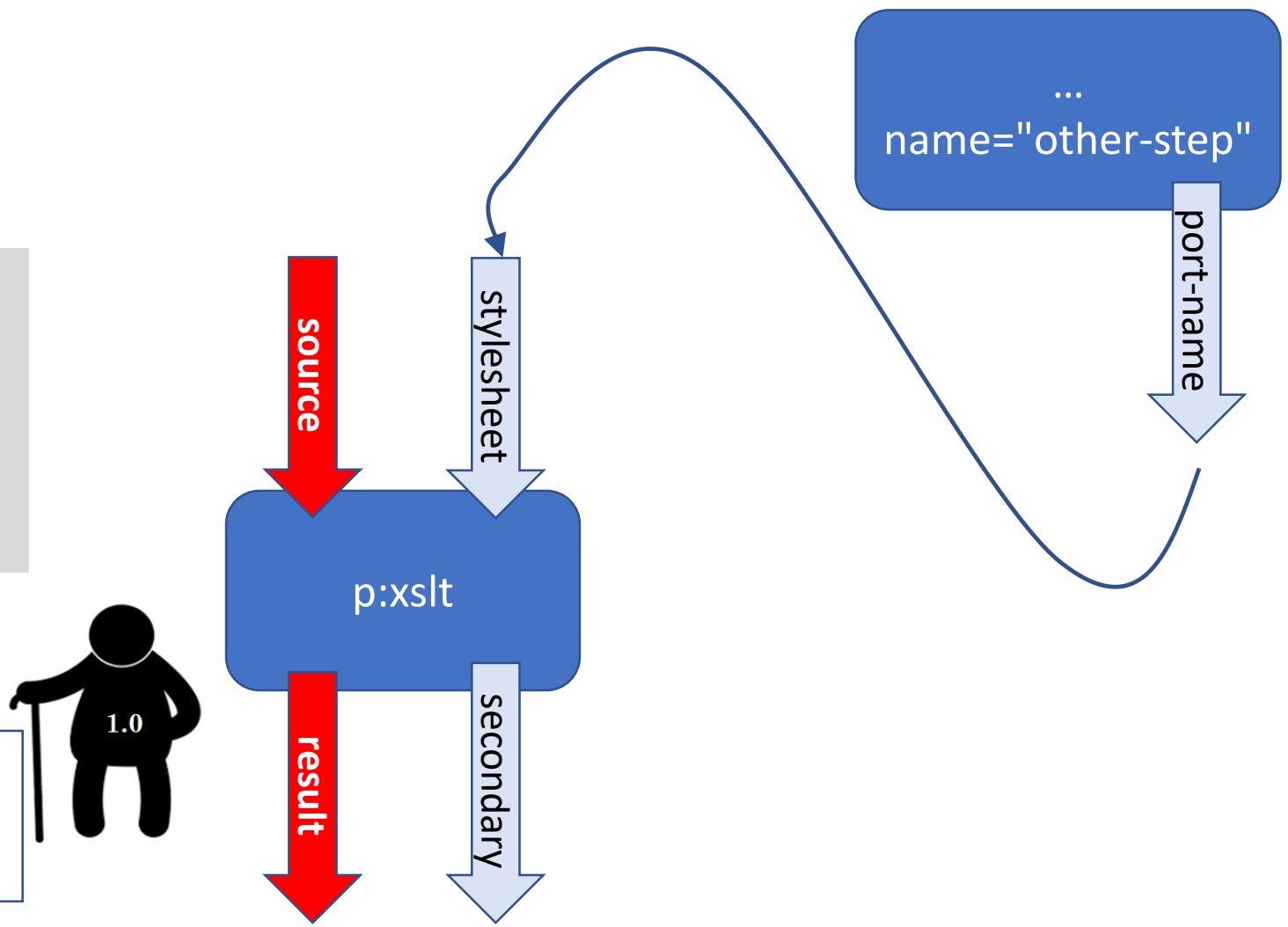


Inline documents can contain Text- and Attribute-Value Templates

<p:inline> no longer necessary (in most cases)

Explicitly connecting ports – to an output port of another step (1.0 & 3.0)

```
<p:xslt>
  <p:with-input port="stylesheet">
    <p:pipe step="other-step"
       port="port-name"/>
  </p:with-input>
</p:xslt>
```

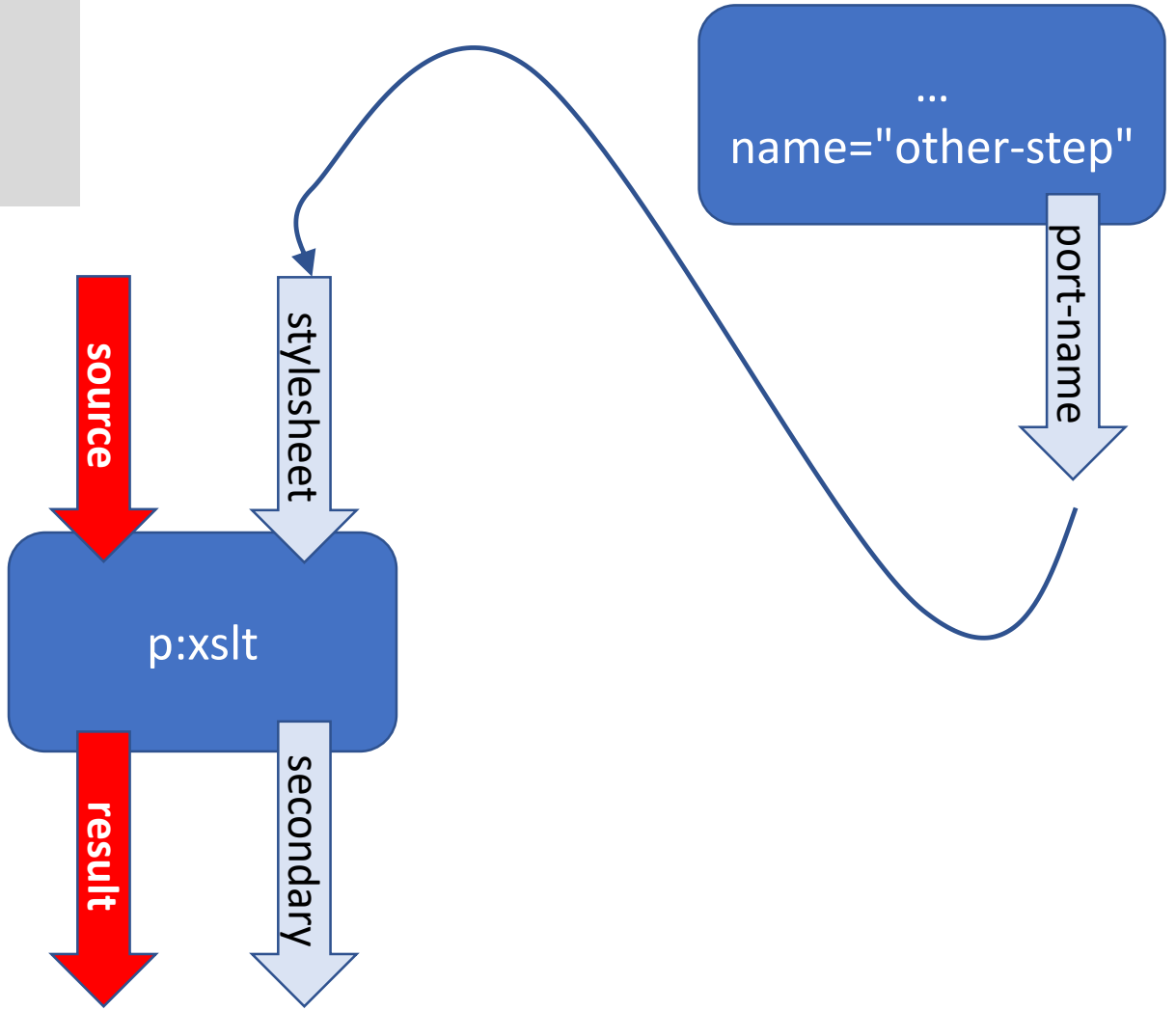


1.0: You *had* to use <p:pipe>

Explicitly connecting ports – to an output port of another step (3.0)

```
<p:xslt>
  <p:with-input port="stylesheet"
    pipe="port-name@other-step"/>
</p:xslt>
```

<p:pipe> can be shortened to @pipe



Setting step parameters (1.0)

```
<p:xslt>
  …
  <p:with-param name="title" select="'bla bla bla'"/>
  <p:with-param name="heading-level" select="2"/>
</p:xslt>
```

```
<xsl:param name="title"/>
<xsl:param name="heading-level"/>
```

1.0: You had to use <p:with-param> or parameter ports

Setting step parameters

```
<xsl:param name="title"/>
<xsl:param name="heading-level"/>
```

```
<p:xslt>
  …
  <p:with-option name="parameters" select="
    map{ 'title': 'bla bla bla',
         'heading-level': 2
     }
  "/>
</p:xslt>
```

Pass parameters in a map

XPath 3.1 support

Variables (1.0)

```
<p:variable name="path" select="/*/@path"/>
```

1.0: String only

1.0: Only at the top of a sub-pipeline

```
<p:group>
  <p:variable name="…" select="…"/>

  …

</p:group
```

Variables (3.0)

```
<p:variable name="path" as="xs:string" select="/*/@path"/>

<p:variable name="debug" as="xs:boolean" select="true()"/>

<p:variable name="mymap" as="map(*)" select="map{ … }"/>
```

XDM support and strong typing

Variables anywhere

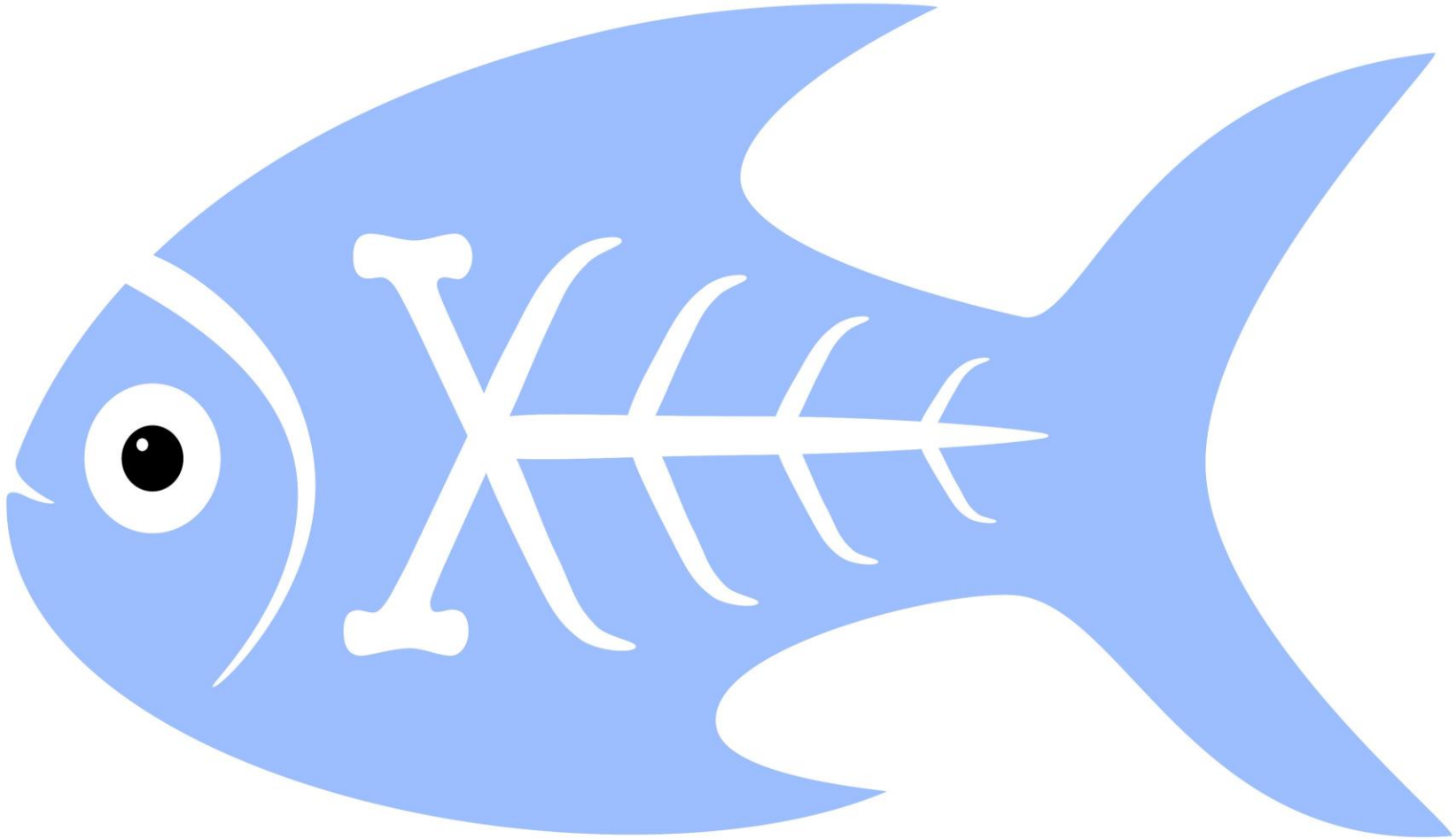
Static variables

- @use-when expressions
- Constants for setting up the pipeline
- Constants in libraries

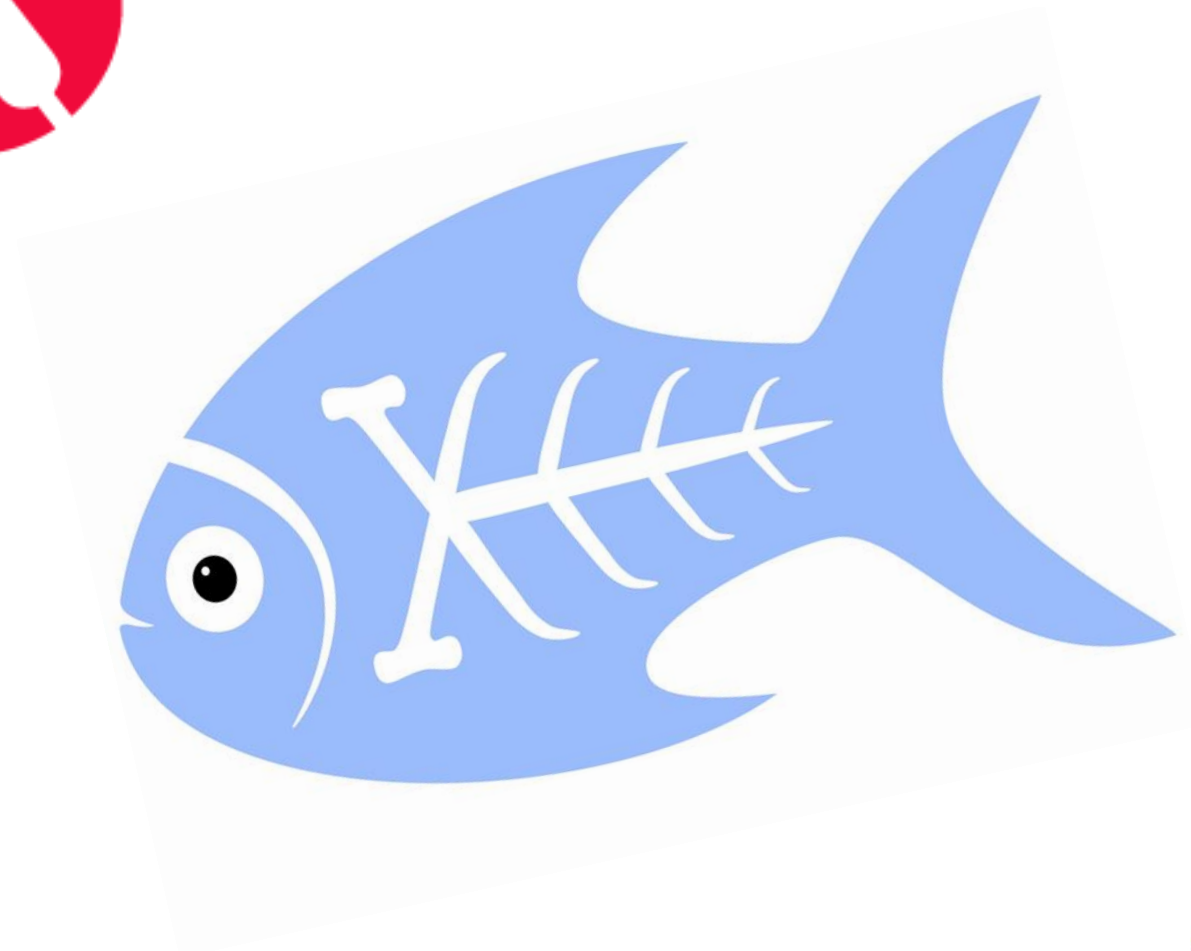
Structure (1.0 & 3.0)

- <p:for-each>
  - ~~<p:iteration-source>~~ 
- <p:choose> / <p:when> / <p:otherwise>
- <p:if> 
- <p:viewport>
- <p:group>
- <p:try> / <p:catch>* 

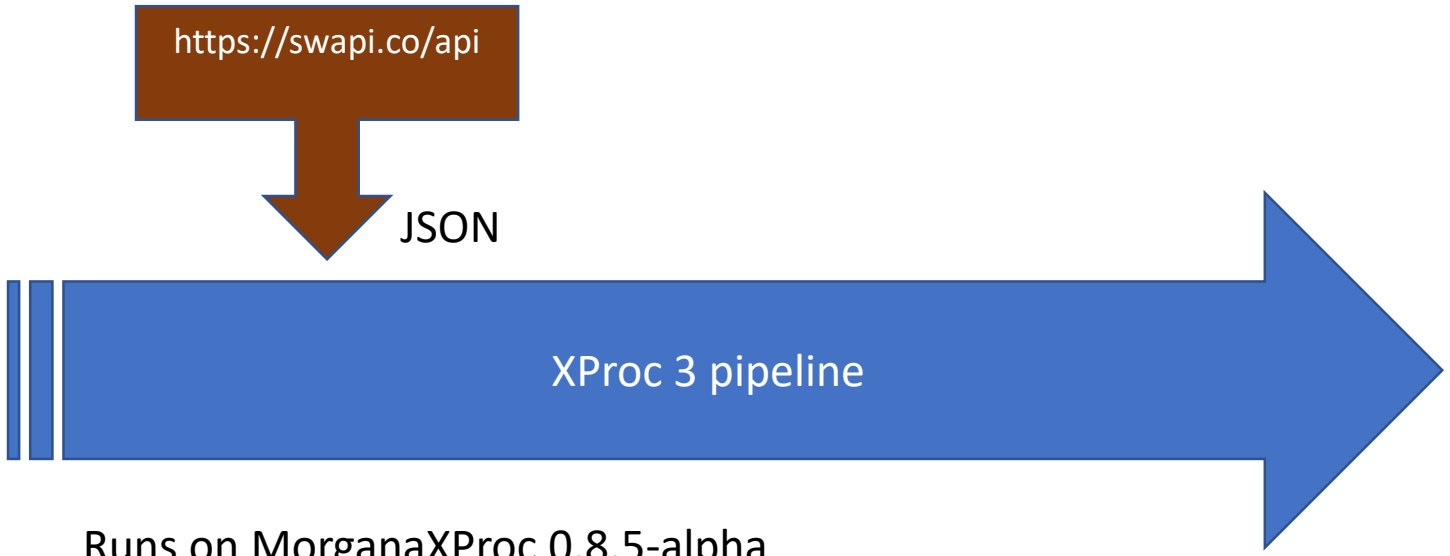
Crash Course XProc fundamentals - end

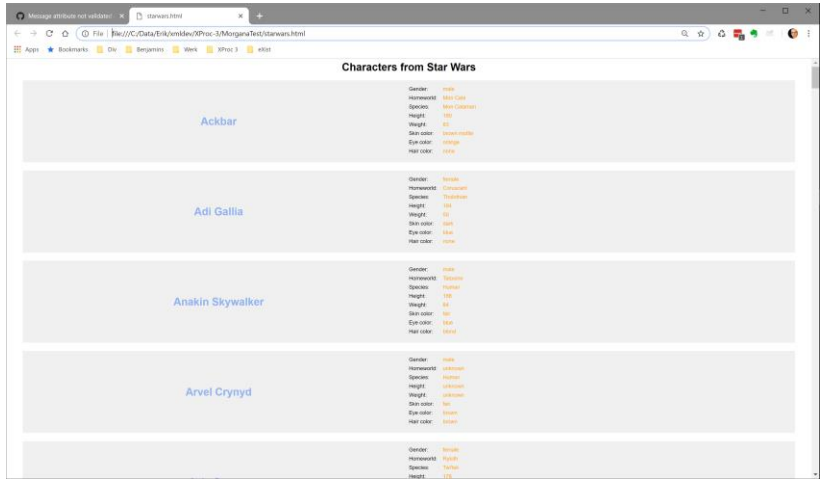


DEMO



MorganaXProc meets Star Wars using JSON





Runs on MorganaXProc 0.8.5-alpha

Native JSON

Convert JSON to XML

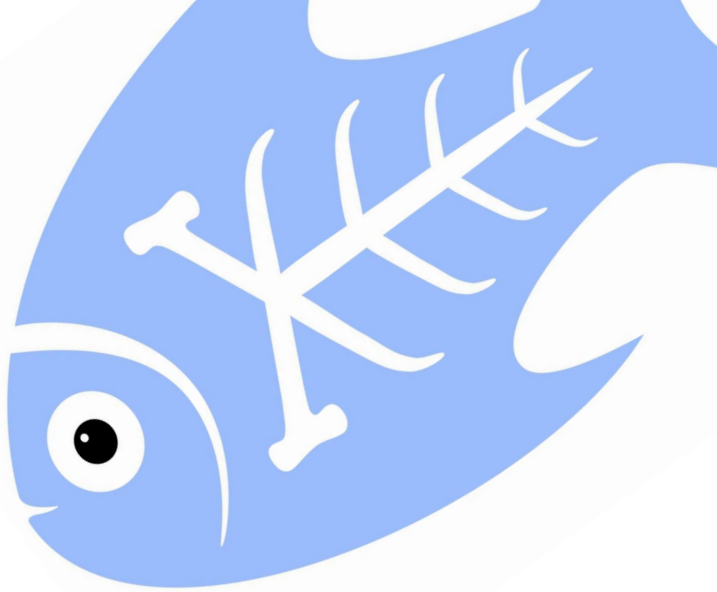
XML

```
<demo:loader p:message="Starting ...">
  <p:with-input href="https://swapi.co/api/people/"/>
</demo:loader>

<p:for-each>
  <mox:select-from-json selector="results"/>
  <p:identity>
    <p:with-input select="array:flatten(.)"/>
  </p:identity>
</p:for-each>

<p:for-each name="looper">
  <p:variable name="homeworld" select=".">
    <p:document href="{map:get(., 'homeworld')}"/>
  </p:variable>

  <mox:replace-in-json selector="homeworld" value="{map:get($homeworld, 'name')}"
    p:message="Processing for {map:get(.,'name')}"/>

  <p:choose>
    <p:when test="array:size(map:get(.,'species')) > 0">
      <p:variable name="species" select=".">
        <p:document href="{array:get(map:get(., 'species'), 1)}"/>
      </p:variable>
      <mox:replace-in-json selector="species" value="{map:get($species, 'name')}"/>
    </p:when>
    <p:otherwise>
      <mox:replace-in-json selector="species" value="unknown"/>
    </p:otherwise>
  </p:choose>
</p:for-each>

<mox:join-JSON/>

<p:cast-content-type content-type="application/xml"/>

<p:xslt message="Formatting...">
  <p:with-input port="stylesheet" href="format-sw.xsl"/>
</p:xslt>
```

Wrap up

- XProc 3.0 is a more concise and easier-to-use version than 1.0
  - Updated underlying standards
  - Support for non-XML documents
  - Lots of syntactic sugar
  - Value templating {…}
- You can help us:
  - By joining the discussions
  - By reviewing the specification
- Processors and learning materials are on their way

XProc 3.0



- Specification: **http://spec.xproc.org/master/head/**
- XProc on github: **https://github.com/xproc/**
- XProc on W3C: **https://www.w3.org/community/xproc-next/**

- Use the issues list on github
- Contact: **xproc-dev@w3.org**