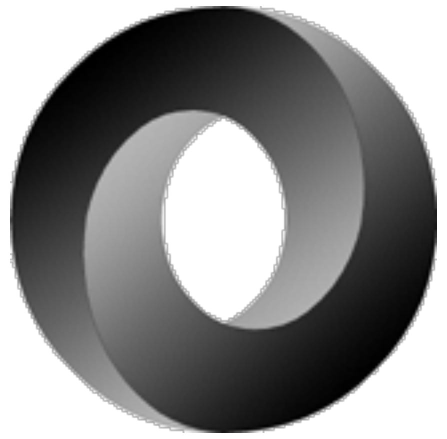
Markup UK 2020

Webinar

XProc 3.0 series

XProc 3.0 series

JSON in XProc



meets

Why bother about JSON?

JSON is the new XML!

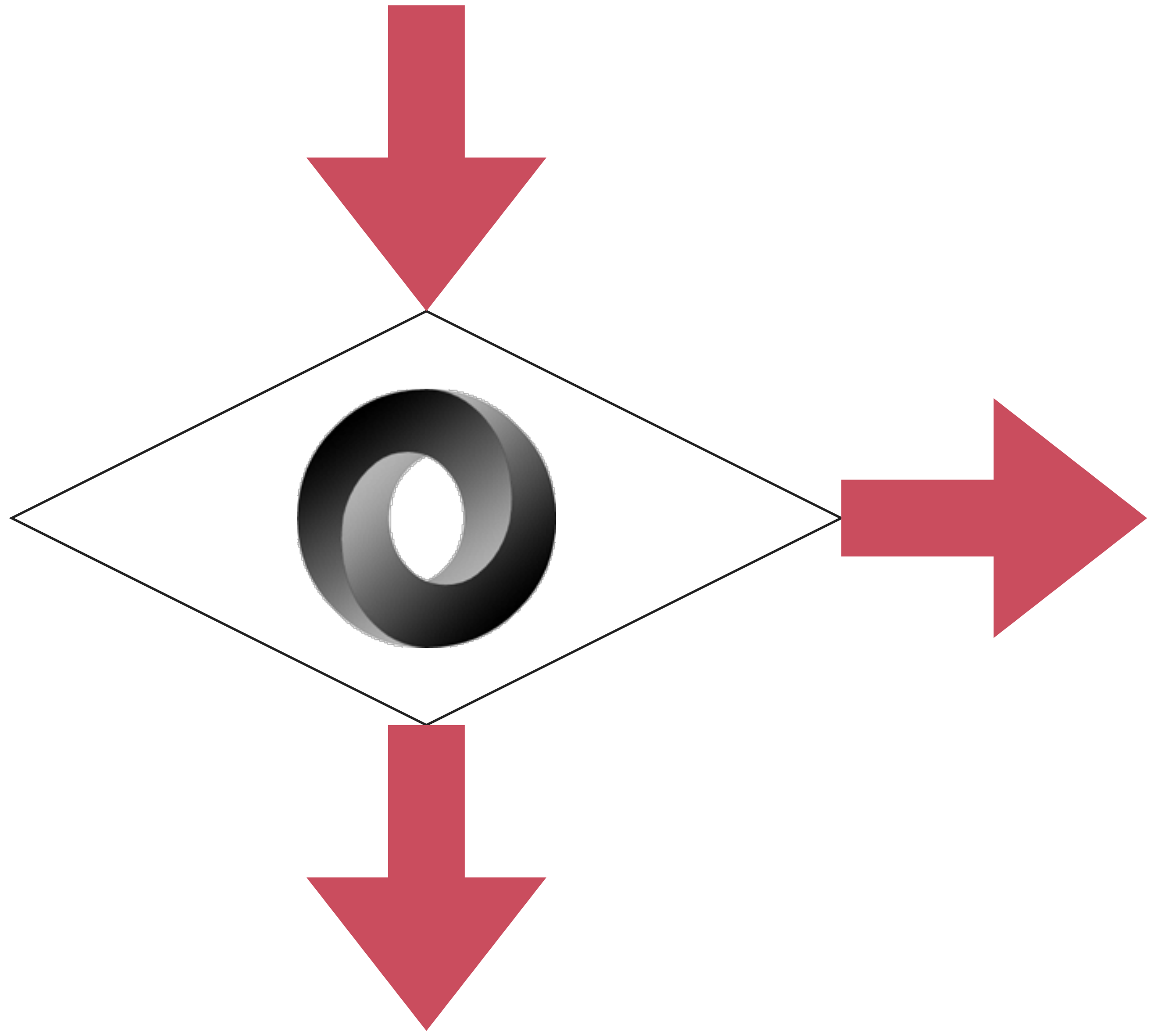
?

- There is a lot of useful JSON out there.
- Sometimes JSON and XML make a perfect combination.

JSON is a first class citizen in XProc

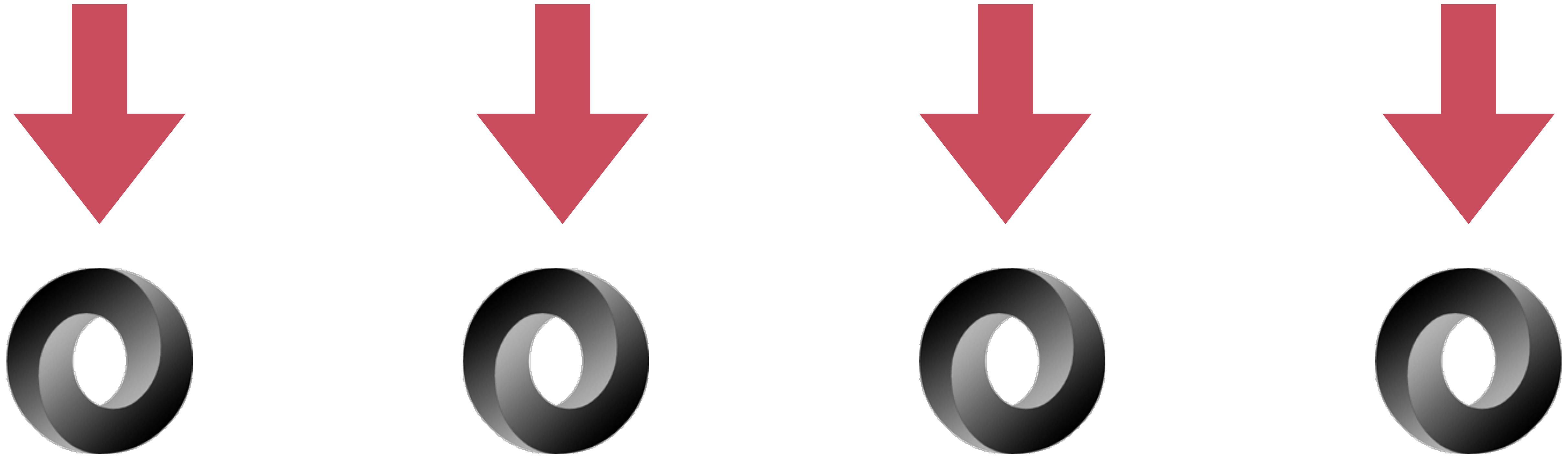
JSON flowing in and/or out of steps.

JSON is a first class citizen in XProc



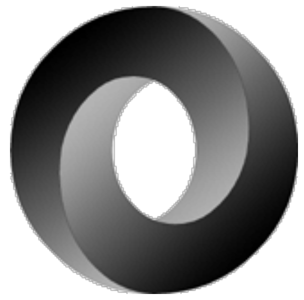
Making decisions based on (content of) JSON document

JSON is a first class citizen in XProc



Iterating over a sequence of JSON documents.

JSON is a first class citizen in XProc



I am first class!

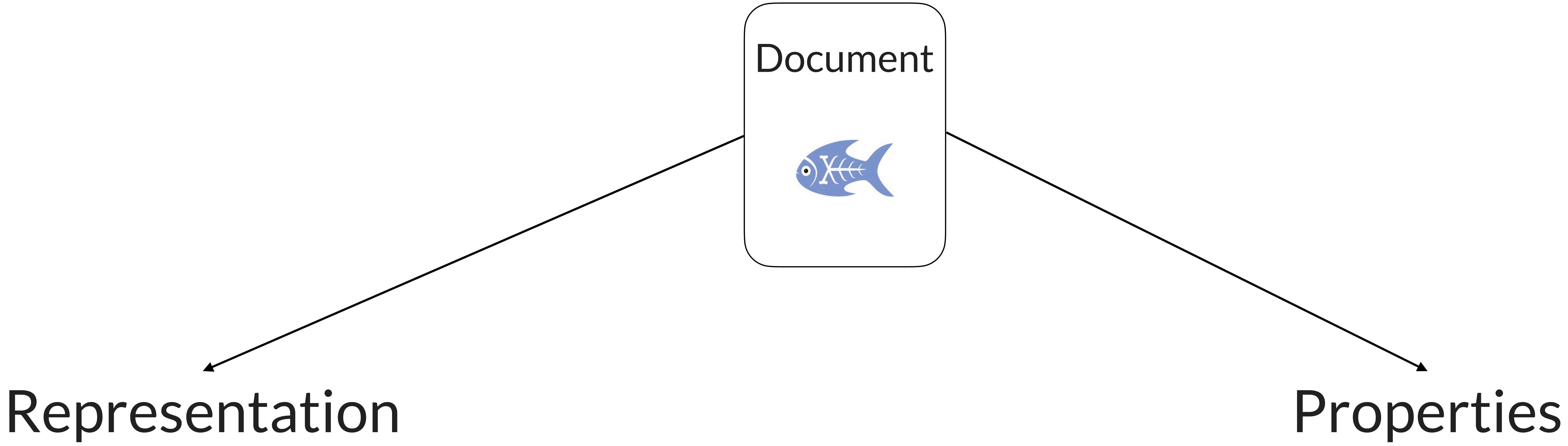
But we are first class too!

XML HTML Text

Not so sure about me!

Others (binary)

A document in XProc

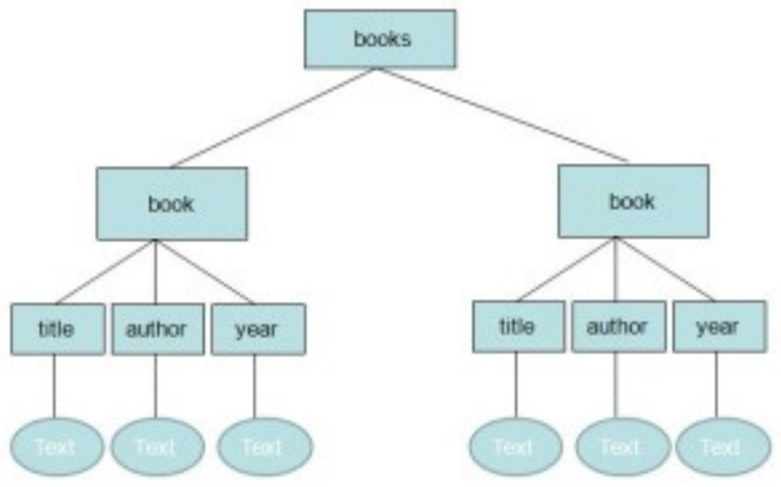


Name	Value
content-type	?

A document in XProc

Document

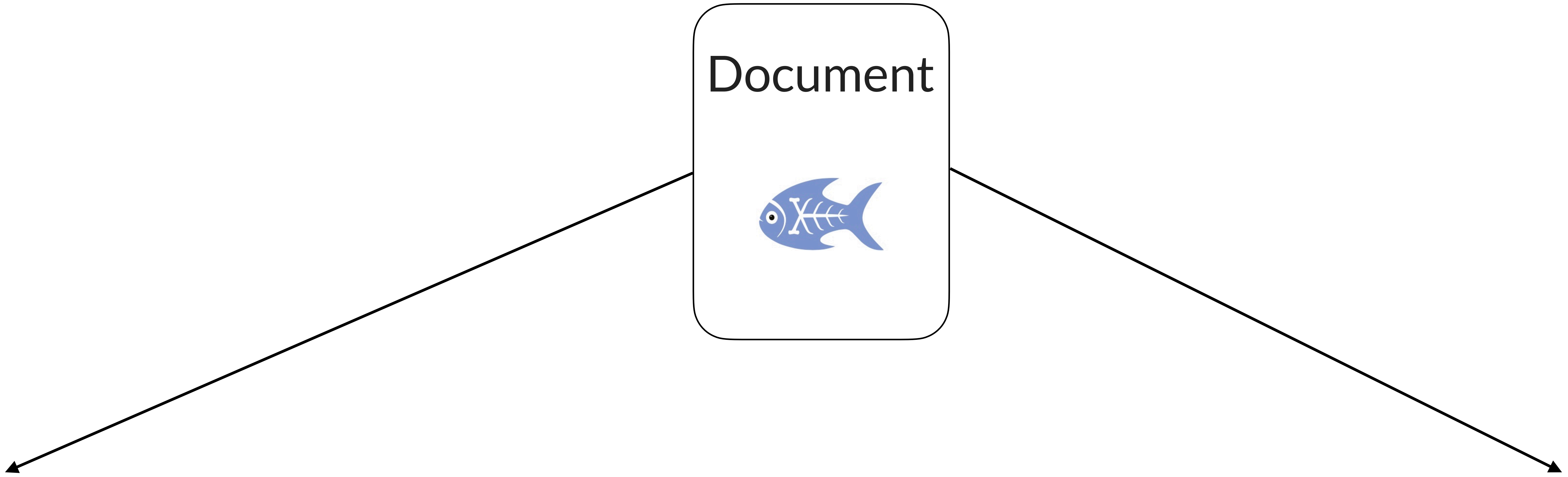
Representation



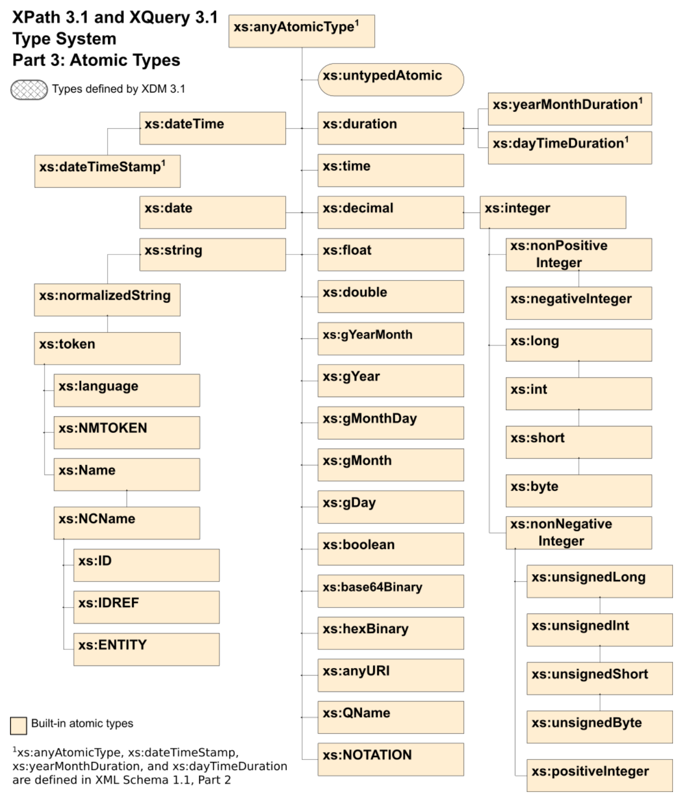
Properties

Name	Value
content-type	*application/xml*

A document in XProc



Representation

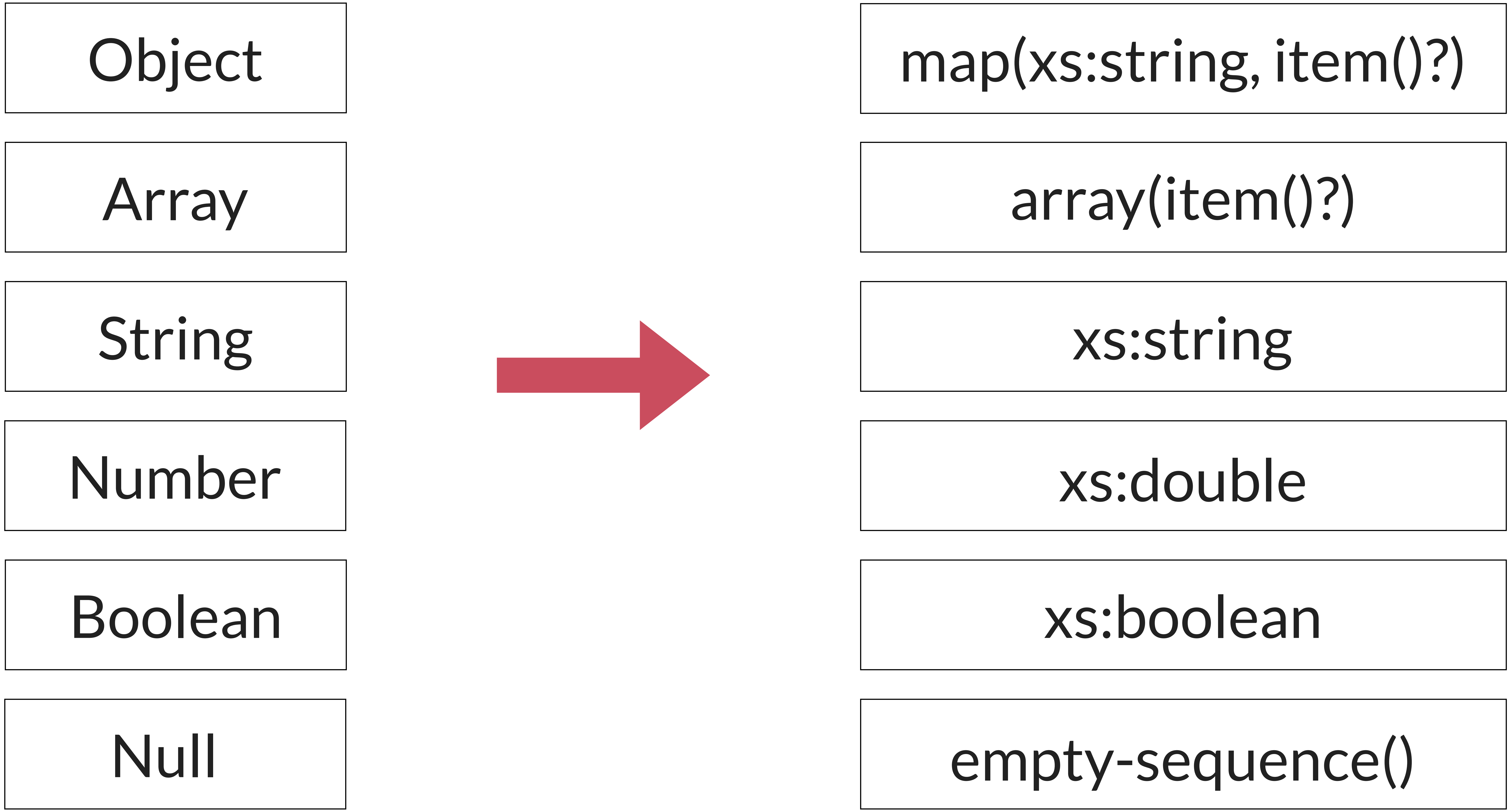

XPath 3.1 and XQuery 3.1
Type System
Part 3: Atomic Types

[1] xs:anyAtomicType, xs:dateTimeStamp, xs:yearMonthDuration, and xs:dayTimeDuration are defined in XML Schema 1.1, Part 2

Properties

Name	Value
content-type	*application/json*

JSON to XDM mapping



Object	map(xs:string, item()?)
Array	array(item()?)
String	xs:string
Number	xs:double
Boolean	xs:boolean
Null	empty-sequence()

Basic use patterns for JSON

Basic use patterns for JSON

Creating a JSON document:

```
<p:identity>
 <p:with-input>
  <p:inline content-type="application/json">
    {{
     "key" : [1, 2, "text", {{"inner" : "val", "second" : null}}]
    }}
   </p:inline>
  </p:with-input>
</p:identity>
```

Basic use patterns for JSON

Creating a JSON document:

```
<p:identity>
 <p:with-input>
  <p:inline content-type="application/json">
    {{
     "key" : [1, 2, "text", {{"inner" : "val", "second" : null}}]
    }}
   </p:inline>
  </p:with-input>
</p:identity>
```

Important: Double "{" and "}" to distinct object limits from TVTs.

Basic use patterns for JSON

Creating a JSON document: Alternative

```
<p:identity>
 <p:with-input>
  <p:inline content-type="application/json" expand-text="false">
     {
      "key" : [1, 2, "text", {"inner" : "val", "second" : null}]
     }
   </p:inline>
  </p:with-input>
</p:identity>
```

Basic use patterns for JSON

Loading a JSON document:

```
<p:load href="doc.json" />

<!-- or, if you want to be sure -->
<p:load href="doc.json" content-type="application/json" />

<!-- directly binding a port -->
<p:identity>
 <p:with-input>
  <p:document href="doc.json" content-type="application/json" />
 </p:with-input>
</p:identity>
```

Basic use patterns for JSON

Loading a JSON document:

```
<!-- Using XPath function -->
<p:identity>
 <p:with-input select="json-doc('doc.json')" />
</p:identity>


<!-- from the web -->
<p:http-request href="http://somewhere/resource.json"
    parameters="map{'override-content-type' : 'application/json'}">
 <p:with-input><p:empty/></p:with-input>
</p:http-request>
```

Basic use patterns for JSON

Loading a JSON document:

```
<!-- Using XPath function -->
<p:identity>
 <p:with-input select="json-doc('doc.json')" />
</p:identity>

<!-- from the web -->
<p:http-request href="http://somewhere/resource.json"
    parameters="map{'override-content-type' : 'application/json'}">
 <p:with-input><p:empty/></p:with-input>
</p:http-request>
```

Basic use patterns for JSON

Check whether a document is a JSON document:

```
<p:if test="p:document-properties(., 'content-type')
  = 'application/json'">
  <p:cast-content-type content-type="application/xml" />
</p:if>


<!-- Pure XPath to be used in step context -->
<p:if test="not (. instance of node())">
  <p:cast-content-type content-type="application/xml" />
</p:if>
```

Basic use patterns for JSON

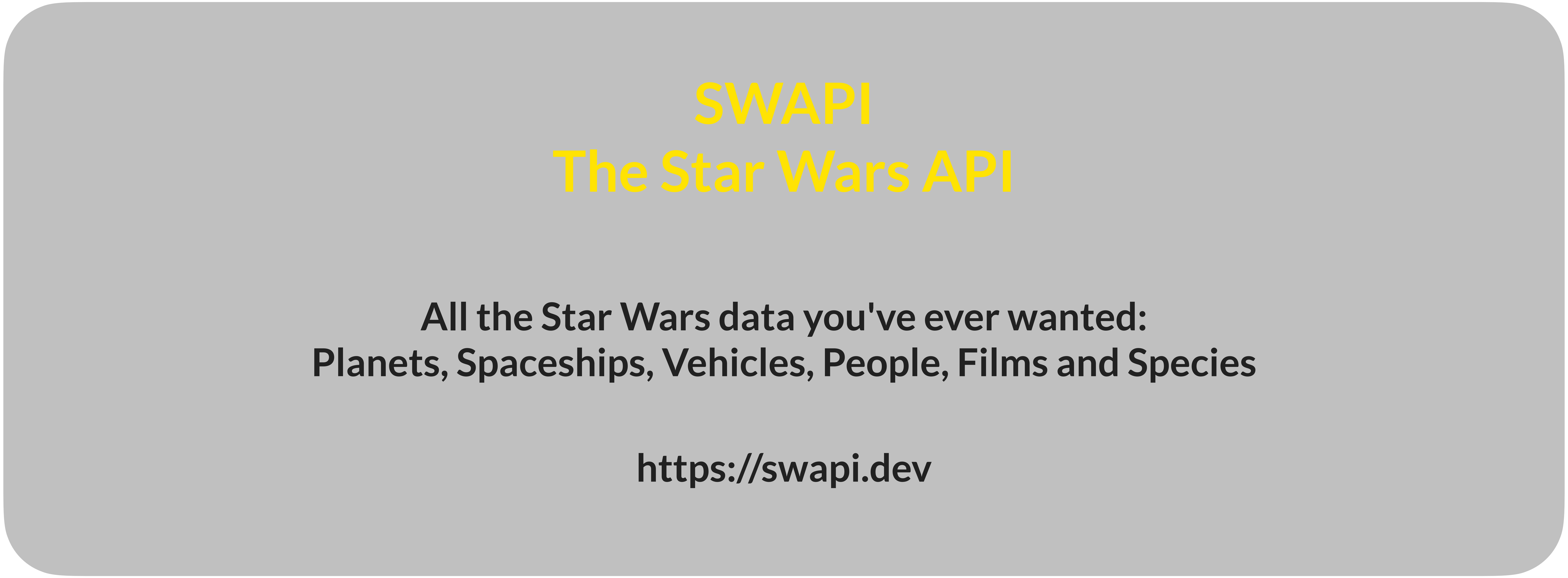
XProc steps designed especially for JSON:

```
<p:json-join />
<!-- Produces one JSON array from all documents port 'source'. -->


<p:json-merge />
<!-- Produces one JSON map from all documents port 'source'. -->


<!-- In the future: -->
<p:json-validate />
```

Let’s go for live coding now…

There is a lot of useful JSON out there!

e.g. this:



SWAPI
The Star Wars API

All the Star Wars data you've ever wanted:
Planets, Spaceships, Vehicles, People, Films and Species

https://swapi.dev

Time to answer your questions!

Contact:	achim.berndzen@xml-project.com
Slides and examples:	https://github.com/xml-project/markup-uk-2020