

XProc 3.0 series



#### XProc 3.0 series

# Text documents in XProc



#### What is a text document?

- + Every document with content-type "text/\*"
  - Except "text/xml" → XML document
  - Except "text/html" → HTML document
- + application/relax-ng-compact-syntax
- + application/xquery
- + application/javascript
- + Implementation defined content-types.



#### What is a text document?

Dear Kate,

Here's to the crazy ones. The mis rebels. The troublemakers. The re the square holes. The ones who st things differently.

They're not fond of rules. And they have no respect for the status quo.

You can praise them, disagree with them, quote them, disbeleve them, glorify or villify them. About the only thing you can't do is ignore them. Because they change things.

They invent. They imagine. They heal. They explore. They create. They inspire. They push the human race forward.

Maybe they have to be crazy. How else can you stare at an empty canvas and see a work of art? Or sit in silence and hear a song that's never been written? Or gaze at a red planet and see a laboratory on wheels? We make tools for these kinds of people.

While some see them as the crazy ones, we see genius. Because the people who are crazy enough to think they can change the world, are the ones who do.

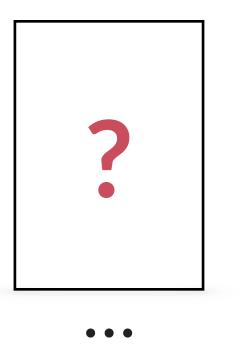
Take care,
John Appleseed

plain text



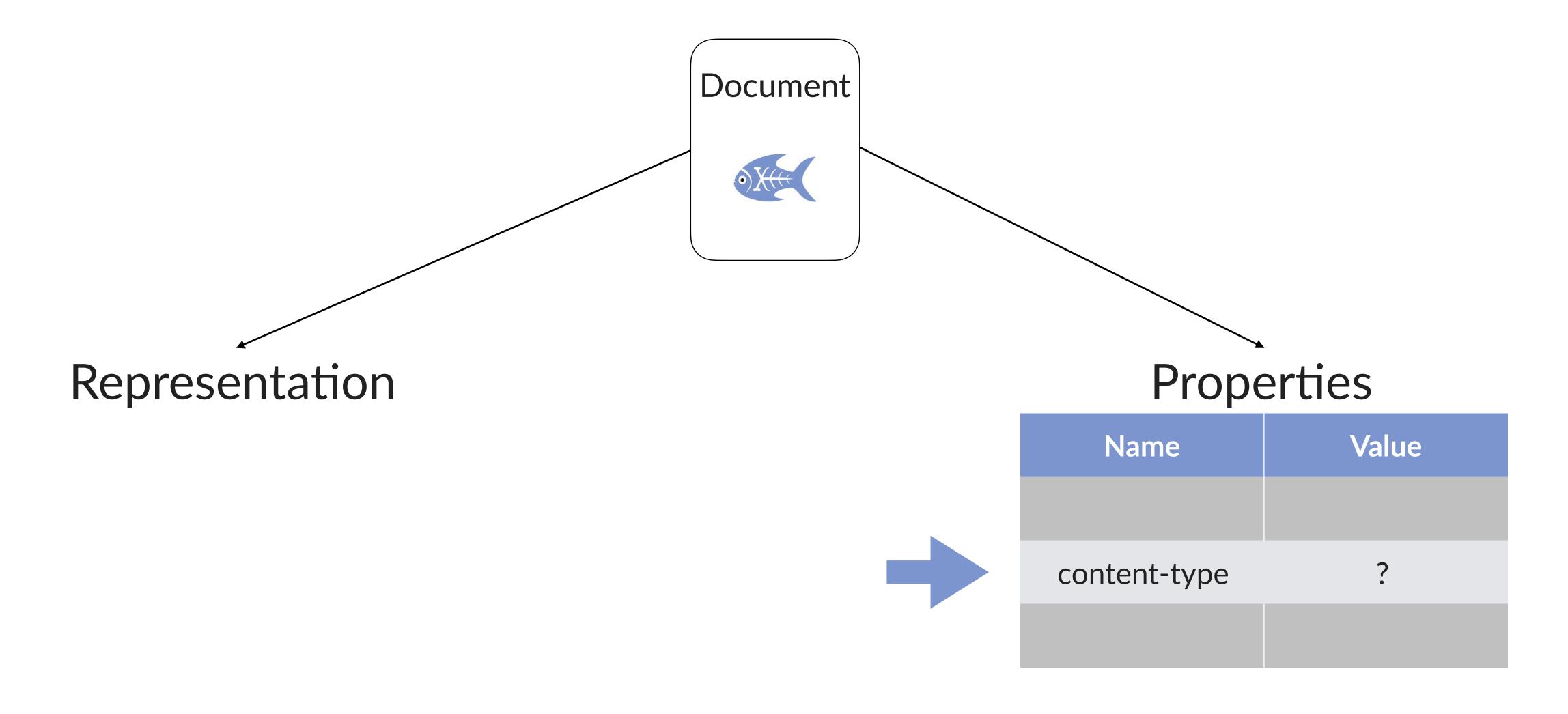






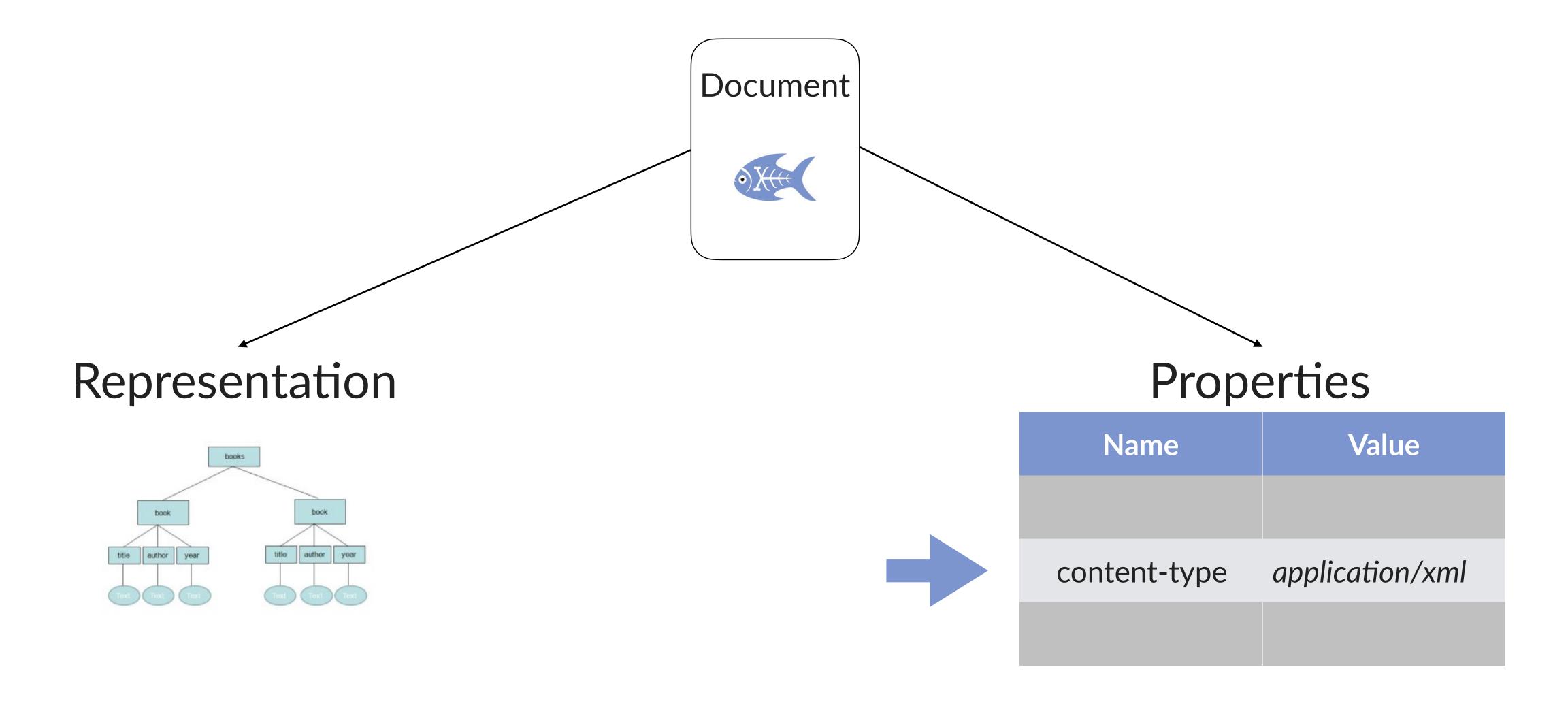


#### Adocument in XProc



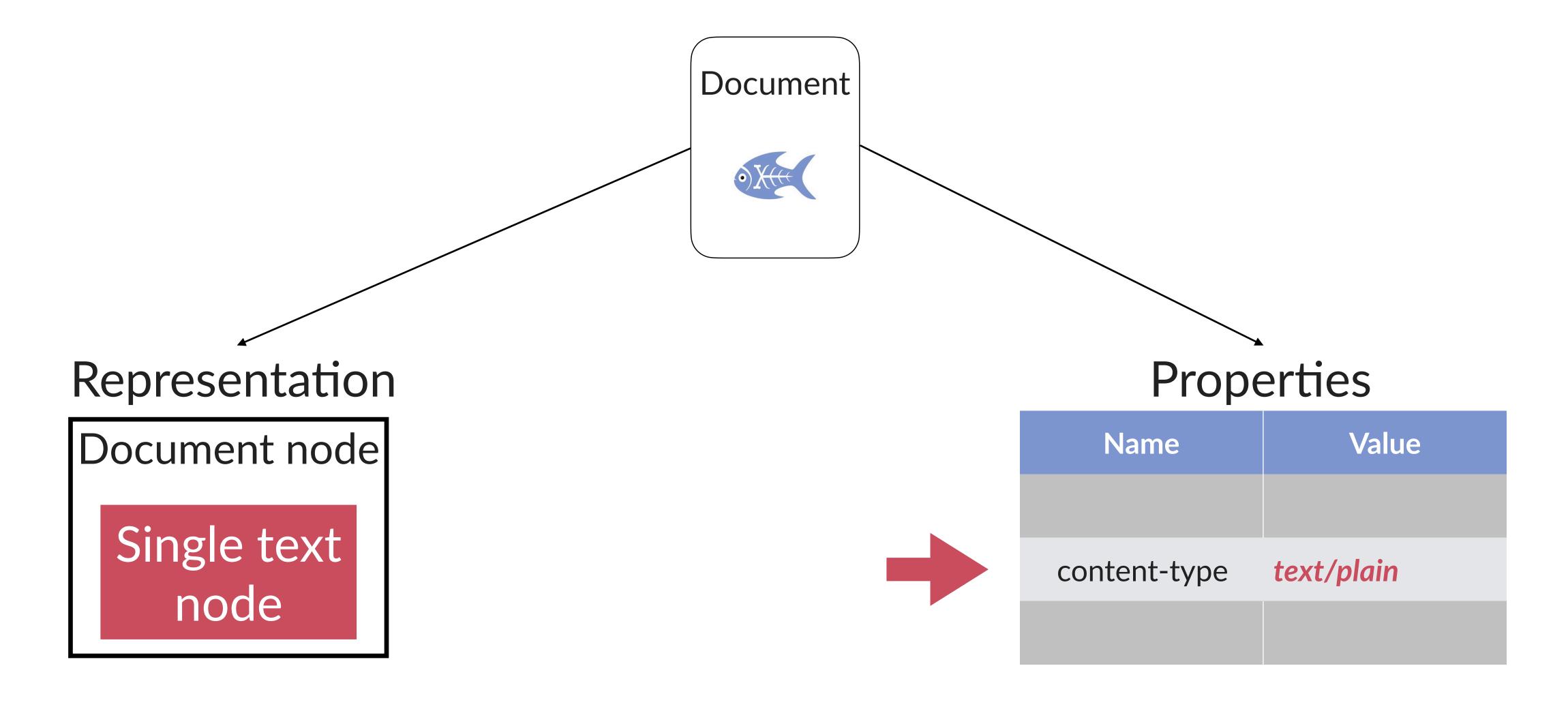


#### Adocument in XProc





#### Adocument in XProc







Creating a text document:

```
<p:identity>
  <p:with-input>
    <p:inline content-type="text/plain">This is a simple text
    containing multiple lines. Since it is inside an XML document
    it is UTF-8.</p:inline>
        </p:with-input>
    </p:identity>
```

Hint: Everything inside p:inline is the text, so be careful with indents!



Creating a text document:

```
<p:identity>
  <p:with-input>
   <p:inline content-type="text/plain">This is a simple text
   containing multiple lines. Since it is inside an XML document
   it is UTF-8.</p:inline>
     </p:with-input>
  </p:identity>
```

Curly brackets in text document start/end text value templates. When you need them as text, double them. Or use @expand-text=,false()"



Creating a text document containing text value templates:

```
<p:identity>
  <p:with-input>
   <p:inline content-type="text/plain">Today is: {current-date()}.
   The time is {current-time()}.</p:inline>
   </p:with-input>
  </p:identity>
```

Of course you can use XProc's variables, options and functions.



Creating a text document: Not!

```
<p:identity>
  <p:with-input>
    <p:inline content-type="text/plain">XML works with "<" and ">".
    p:inline>
    </p:with-input>
</p:identity>
```



Creating a text document: Not!

```
<p:identity>
  <p:with-input>
    <p:inline content-type="text/plain">XML works with "<" and ">".</p:inline>
    </p:with-input>
  </p:identity>
```

Remember it's XML: You have to escape angle brackets.



Creating a text document:

```
<p:identity>
  <p:with-input>
    <p:inline content-type="text/plain">XML works with "&lt;" and "&gt;".</p:inline>
    </p:with-input>
</p:identity>
```

Remember it's XML: You have to escape angle brackets.



Creating a text document: (NOT!)

```
<p:identity>
  <p:with-input>
   <p:inline>This is not a text document.</p:inline>
   </p:with-input>
</p:identity>
```

The content-type attribute is **not optional** here: A p:inline without content-type is an XML document by default. (= application/xml)



Loading a text document:

```
<p:load href="note.txt" />
<!-- or, if you want to be sure -->
<p:load href="note.txt" content-type="text/plain" />
<!-- directly binding a port -->
<p:identity>
<p:with-input>
 <p:document href="note.txt" content-type="text/plain" />
</p:identity>
```



Loading a text document:

```
<!-- from the web -->
<p:http-request href="http://somewhere/text-resource"</pre>
    parameters="map{'override-content-type' : 'text/plain;
    charset=iso-8859-1'}">
<p:with-input><p:empty/></p:with-input>
</p:http-request>
```



Loading a text document:

```
<!-- Using XPath function -->
<p:identity>
<p:with-input>
    <p:inline content-type="text/plain">{unparsed-text('text.txt')
}</p:inline>
</p:identity>
```



Loading a text document: (NOT!)

```
<!-- Using XPath function -->
<p:identity>
  <p:with-input select="unparsed-text('text.txt')" />
        <p:empty />
  </p:identity>
```

This will NOT create a text document, but a JSON document: The selection result is an atomic value and the specs says those become JSON docs.



Check whether a document is a text document:

```
<p:if test="p:document-properties(., 'content-type')</pre>
  = 'text/plain'">
  <p:wrap-sequence wrapper="text" />
</p:if>
<!-- Pure XPath to be used in step context -->
<p:if test="count(//node())=1 and ./node() instance of text()">
 <p:wrap-sequence wrapper="text">
</p:if>
<!-- But remember: The test is also true for some XML documents. -->
```



## Build in steps for text documents

- p:text-count
- p:text-head
- p:text-tail
- p:text-join
- p:text-replace
- p:text-sort
- p:markdown-to-html (Text step library)

Want more text related step? Tell us!



# Let's go for live coding now...

#### Time to answer your questions!

Contact: achim.berndzen@xml-project.com

Slides and examples: https://github.com/xml-project/markup-uk-2020

