



# What will it take to get (end user) XML editors that people will use?

Norman Walsh  
MarkLogic Corporation  
2 August 2011





# What will it take to get (end user) XML editors that *don't suck*?

Norman Walsh  
MarkLogic Corporation  
2 August 2011



**Any questions?**

**A miracle**



# But, seriously...

---

Let's take a survey.



# Survey question: 1 of 9

Have you ever purchased something over the web?

- ☒ All
- ☐ Almost all
- ☐ Most
- ☐ Some
- ☐ A few
- ☐ None



# Survey question: 2 of 9

Do you regularly drive an automobile, motorcycle, etc.?

- ☐ All
- ☒ Almost all
- ☐ Most
- ☐ Some
- ☐ A few
- ☐ None



# Survey question: 3 of 9

Have you used Emacs?

- ☐ All
- ☐ Almost all
- ☐ Most
- ☒ Some
- ☐ A few
- ☐ None



# Survey question: 4 of 9

Have you used that other editor?

- ☐ All
- ☐ Almost all
- ☐ Most
- ☒ Some
- ☐ A few
- ☐ None



# Survey question: 5 of 9

Have you used oXygen?

- ☐ All
- ☐ Almost all
- ☒ Most
- ☐ Some
- ☐ A few
- ☐ None



# Survey question: 6 of 9

Have you used Notepad?

- ☐ All
- ☐ Almost all
- ☐ Most
- ☐ Some
- ☒ A few
- ☐ None



# Survey question: 7 of 9

Have you used Arbortext Editor?

- ☐ All
- ☐ Almost all
- ☐ Most
- ☒ Some
- ☐ A few
- ☐ None



# Survey question: 8 of 9

Have you used a recent version of Microsoft Word?

- ☐ All
- ☒ Almost all
- ☐ Most
- ☐ Some
- ☐ A few
- ☐ None



# Survey question: 9 of 9

Have you commented on a weblog?

- ☐ All
- ☒ Almost all
- ☐ Most
- ☐ Some
- ☐ A few
- ☐ None



# What do we really mean?

*What will it take to get (end user) XML editors that people will use?*

- What do we mean by XML?
- What do we mean by editor?
- What do we mean by people?



# What do we mean by XML?

`<purchase/>`



# What do we mean by XML?

```
<purchase/>  
<drive/>  
<emacs/>
```

## Sidebar

- Edit only well-formed XML documents?
- Edit any XML fragment?
- Edit XML-like documents that aren't well-formed or are momentarily not well-formed?
  - Is ~/.bashrc or c:\autoexec.bat a not-well formed XML file?



# What do we mean by XML?

```
<purchase>all</purchase>  
<drive>almost</drive>  
<emacs>some</emacs>
```



# What do we mean by XML?

```
<survey>  
  <purchase>all</purchase>  
  <drive>almost</drive>  
  <emacs>some</emacs>  
  ...
```

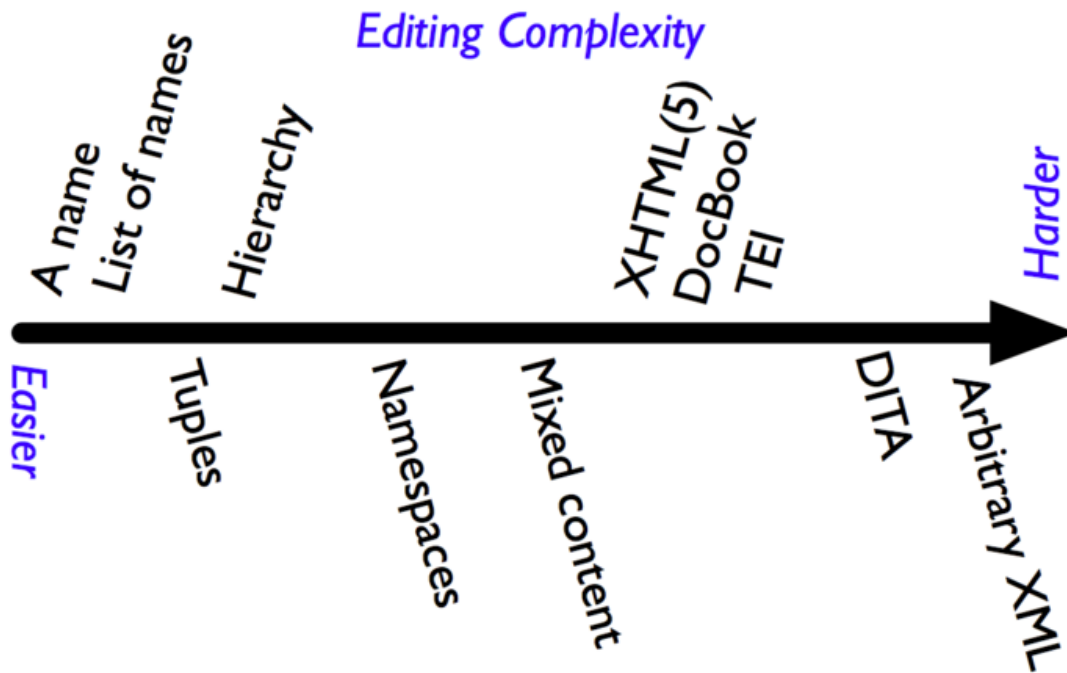


# What do we mean by XML?

- Namespaces ...
- Mixed content ...
- HTML, DocBook, TEI, DITA, ...
- ...
- Arbitrary XML



# What do we mean by XML?





# What do we mean by editor?

Survey says!

```
<survey>
  <purchase>all</purchase>
  <drive>almost</drive>
  <emacs>some</emacs>
  <other>some</other>
  <oxygen>most</oxygen>
  <notepad>few</notepad>
  <epic>some</epic>
  <msword>almost</msword>
  <comment>almost</comment>
</survey>
```



Editing XML – Wikipedia, the free encyclopedia

en.wikipedia.org/w/index.php?title=XML&action=edit

Nwalsh My talk My preferences My watchlist My contributions Log out

Article Discussion Read Edit View history Search

## Editing XML

From Wikipedia, the free encyclopedia

**B I** **Advanced** **Special characters** **Help** **Cite**

"Extensible Markup Language" ("XML") is a set of rules for encoding documents in [[machine-readable]] form. It is defined in the XML 1.0 Specification<ref>{{cite web|url=http://www.w3.org/TR/REC-xml |title=XML 1.0 Specification |publisher=W3.org |date= |accessdate=2010-08-22}}</ref> produced by the [[W3C]], and several other related specifications, all [[gratis]] [[open standard]]s.<ref>{{cite web|title=W3C DOCUMENT LICENSE|url=http://www.w3.org/Consortium/Legal/2002/copyright-documents-20021231}}</ref>

The design goals of XML emphasize simplicity, generality, and usability over the [[Internet]].<ref name="XML Goals">{{cite web|title=XML 1.0 Origin and Goals|url=http://www.w3.org/TR/REC-xml/#sec-origin-goals|accessdate=July 2009}}</ref> It is a textual data format with strong support via [[Unicode]] for the languages of the world. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary [[data structures]], for example in [[web service]]s.

Many [[application programming interfaces]] (APIs) have been developed that software developers use to process XML data, and several [[XML schema|schema systems]] exist to aid in the definition of XML-based languages.

{{As of|2009}}, hundreds of XML-based languages have been developed,<ref name="Cover pages list">{{cite web|url=http://xml.coverpages.org/xmlApplications.html|title=XML Applications and Initiatives}}</ref> including [[RSS]], [[Atom (standard)|Atom]], [[SOAP]], and [[XHTML]]. XML-based formats have become the default for most office-productivity tools, including [[Microsoft Office]] ([[Office Open XML]]), [[OpenOffice.org]] ([[OpenDocument]]), and [[Apple Computer|Apple]]'s [[iWork]].<ref>{{cite web|title=Introduction to iWork Programming Guide. Mac OS X Reference Library|publisher=Apple|url=http://developer.apple.com/mac/library/documentation/AppleApplications/Conceptual/iWork2-0\_XML/Chapter01/Introduction.html}}</ref>

==Key terminology==

Content that violates any copyrights will be deleted. Encyclopedic content must be **verifiable**.

By clicking the "Save Page" button, you agree to the [Terms of Use](#), and you irrevocably agree to release your contribution under the [CC-BY-SA 3.0 License](#) and the [GFDL](#). You agree that a hyperlink or URL is sufficient attribution under the Creative Commons license.

**Edit summary** (Briefly describe the changes you have made)

☐ This is a minor edit ([what's this?](#)) ☐ Watch this page

[Cancel](#) | [Editing help](#) (opens in new window)

If you do not want your writing to be edited, used, and redistributed at will, then do not submit it here. All text that you did not write yourself, except brief excerpts, must be available under terms consistent with Wikipedia's [Terms of Use](#) before you submit it.

— <sup>12</sup> <sup>13</sup> <sup>14</sup> <sup>15</sup> <sup>16</sup> <sup>17</sup> <sup>18</sup> <sup>19</sup> <sup>20</sup> <sup>21</sup> <sup>22</sup> <sup>23</sup> <sup>24</sup> <sup>25</sup> <sup>26</sup> <sup>27</sup> <sup>28</sup> <sup>29</sup> <sup>30</sup> <sup>31</sup> <sup>32</sup> <sup>33</sup> <sup>34</sup> <sup>35</sup> <sup>36</sup> <sup>37</sup> <sup>38</sup> <sup>39</sup> <sup>40</sup> <sup>41</sup> <sup>42</sup> <sup>43</sup> <sup>44</sup> <sup>45</sup> <sup>46</sup> <sup>47</sup> <sup>48</sup> <sup>49</sup> <sup>50</sup> <sup>51</sup> <sup>52</sup> <sup>53</sup> <sup>54</sup> <sup>55</sup> <sup>56</sup> <sup>57</sup> <sup>58</sup> <sup>59</sup> <sup>60</sup> <sup>61</sup> <sup>62</sup> <sup>63</sup> <sup>64</sup> <sup>65</sup> <sup>66</sup> <sup>67</sup> <sup>68</sup> <sup>69</sup> <sup>70</sup> <sup>71</sup> <sup>72</sup> <sup>73</sup> <sup>74</sup> <sup>75</sup> <sup>76</sup> <sup>77</sup> <sup>78</sup> <sup>79</sup> <sup>80</sup> <sup>81</sup> <sup>82</sup> <sup>83</sup> <sup>84</sup> <sup>85</sup> <sup>86</sup> <sup>87</sup> <sup>88</sup> <sup>89</sup> <sup>90</sup> <sup>91</sup> <sup>92</sup> <sup>93</sup> <sup>94</sup> <sup>95</sup> <sup>96</sup> <sup>97</sup> <sup>98</sup> <sup>99</sup> <sup>100</sup> <sup>101</sup> <sup>102</sup> <sup>103</sup> <sup>104</sup> <sup>105</sup> <sup>106</sup> <sup>107</sup> <sup>108</sup> <sup>109</sup> <sup>110</sup> <sup>111</sup> <sup>112</sup> <sup>113</sup> <sup>114</sup> <sup>115</sup> <sup>116</sup> <sup>117</sup> <sup>118</sup> <sup>119</sup> <sup>120</sup> <sup>121</sup> <sup>122</sup> <sup>123</sup> <sup>124</sup> <sup>125</sup> <sup>126</sup> <sup>127</sup> <sup>128</sup> <sup>129</sup> <sup>130</sup> <sup>131</sup> <sup>132</sup> <sup>133</sup> <sup>134</sup> <sup>135</sup> <sup>136</sup> <sup>137</sup> <sup>138</sup> <sup>139</sup> <sup>140</sup> <sup>141</sup> <sup>142</sup> <sup>143</sup> <sup>144</sup> <sup>145</sup> <sup>146</sup> <sup>147</sup> <sup>148</sup> <sup>149</sup> <sup>150</sup> <sup>151</sup> <sup>152</sup> <sup>153</sup> <sup>154</sup> <sup>155</sup> <sup>156</sup> <sup>157</sup> <sup>158</sup> <sup>159</sup> <sup>160</sup> <sup>161</sup> <sup>162</sup> <sup>163</sup> <sup>164</sup> <sup>165</sup> <sup>166</sup> <sup>167</sup> <sup>168</sup> <sup>169</sup> <sup>170</sup> <sup>171</sup> <sup>172</sup> <sup>173</sup> <sup>174</sup> <sup>175</sup> <sup>176</sup> <sup>177</sup> <sup>178</sup> <sup>179</sup> <sup>180</sup> <sup>181</sup> <sup>182</sup> <sup>183</sup> <sup>184</sup> <sup>185</sup> <sup>186</sup> <sup>187</sup> <sup>188</sup> <sup>189</sup> <sup>190</sup> <sup>191</sup> <sup>192</sup> <sup>193</sup> <sup>194</sup> <sup>195</sup> <sup>196</sup> <sup>197</sup> <sup>198</sup> <sup>199</sup> <sup>200</sup> <sup>201</sup> <sup>202</sup> <sup>203</sup> <sup>204</sup> <sup>205</sup> <sup>206</sup> <sup>207</sup> <sup>208</sup> <sup>209</sup> <sup>210</sup> <sup>211</sup> <sup>212</sup> <sup>213</sup> <sup>214</sup> <sup>215</sup> <sup>216</sup> <sup>217</sup> <sup>218</sup> <sup>219</sup> <sup>220</sup> <sup>221</sup> <sup>222</sup> <sup>223</sup> <sup>224</sup> <sup>225</sup> <sup>226</sup> <sup>227</sup> <sup>228</sup> <sup>229</sup> <sup>230</sup> <sup>231</sup> <sup>232</sup> <sup>233</sup> <sup>234</sup> <sup>235</sup> <sup>236</sup> <sup>237</sup> <sup>238</sup> <sup>239</sup> <sup>240</sup> <sup>241</sup> <sup>242</sup> <sup>243</sup> <sup>244</sup> <sup>245</sup> <sup>246</sup> <sup>247</sup> <sup>248</sup> <sup>249</sup> <sup>250</sup> <sup>251</sup> <sup>252</sup> <sup>253</sup> <sup>254</sup> <sup>255</sup> <sup>256</sup> <sup>257</sup> <sup>258</sup> <sup>259</sup> <sup>260</sup> <sup>261</sup> <sup>262</sup> <sup>263</sup> <sup>264</sup> <sup>265</sup> <sup>266</sup> <sup>267</sup> <sup>268</sup> <sup>269</sup> <sup>270</sup> <sup>271</sup> <sup>272</sup> <sup>273</sup> <sup>274</sup> <sup>275</sup> <sup>276</sup> <sup>277</sup> <sup>278</sup> <sup>279</sup> <sup>280</sup> <sup>281</sup> <sup>282</sup> <sup>283</sup> <sup>284</sup> <sup>285</sup> <sup>286</sup> <sup>287</sup> <sup>288</sup> <sup>289</sup> <sup>290</sup> <sup>291</sup> <sup>292</sup> <sup>293</sup> <sup>294</sup> <sup>295</sup> <sup>296</sup> <sup>297</sup> <sup>298</sup> <sup>299</sup> <sup>300</sup> <sup>301</sup> <sup>302</sup> <sup>303</sup> <sup>304</sup> <sup>305</sup> <sup>306</sup> <sup>307</sup> <sup>308</sup> <sup>309</sup> <sup>310</sup> <sup>311</sup> <sup>312</sup> <sup>313</sup> <sup>314</sup> <sup>315</sup> <sup>316</sup> <sup>317</sup> <sup>318</sup> <sup>319</sup> <sup>320</sup> <sup>321</sup> <sup>322</sup> <sup>323</sup> <sup>324</sup> <sup>325</sup> <sup>326</sup> <sup>327</sup> <sup>328</sup> <sup>329</sup> <sup>330</sup> <sup>331</sup> <sup>332</sup> <sup>333</sup> <sup>334</sup> <sup>335</sup> <sup>336</sup> <sup>337</sup> <sup>338</sup> <sup>339</sup> <sup>340</sup> <sup>341</sup> <sup>342</sup> <sup>343</sup> <sup>344</sup> <sup>345</sup> <sup>346</sup> <sup>347</sup> <sup>348</sup> <sup>349</sup> <sup>350</sup> <sup>351</sup> <sup>352</sup> <sup>353</sup> <sup>354</sup> <sup>355</sup> <sup>356</sup> <sup>357</sup> <sup>358</</sup>



# What do we mean by editor?

← → ↺

https://wiki.marklogic.com/pages/editpage.action?pagelId=9511858


☆ 🔑 🗨️ 🔄 🛑 🚫

Dashboard > Mark Logic Wiki > ...

Browse ▾ Norman Walsh ▾

🔍 Search Confluence












> XSLT and XQuery > XSLT > Edit Page



XSLT

Rich Text Wiki Markup Preview

Save Cancel


Paragraph **B** *I* U ABC           Insert 

## What is XSLT?

[XSL Transformations](#) (XSLT 2.0) is a language for transforming XML documents into other XML documents, HTML documents. You might want to format chapters of a book using XSL-FO, or you might want to take a database and process it into a report, but on anything that can be made to look like XML: database tables, geographical information systems, file systems, anything from which your XSLT processor instance. In some cases an XSLT 2.0 processor might also be able to work directly from a database of XDM ability to operate on multiple input files in multiple formats, and to treat them all as if they were XML files, is v shared with [XQuery](#), and with anything else using XPath 2.0

## Specialist

List of specialists for this domain:

 ([Markspace](#))

## Maintainer

Hint: press **⌘M** to open the image browser.

What did you change?

☐ Minor change? (no notifications will be sent)

Location: Mark Logic Wiki > XSLT and XQuery [Edit](#)

Restrictions: [Edit](#)



# What do we mean by editor?



```
ch01.xml
<?xml version="1.0" encoding="UTF-8" ?>
<chapter xmlns="http://docbook.org/ns/docbook"
  xmlns:cx="http://xmlcalabash.com/ns/extensions"
  xmlns:xi="http://www.w3.org/2001/XInclude"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:p="http://www.w3.org/ns/xproc"
  version="5.0-xproc" xml:id="introduction">
  <info>
    <title>Introduction</title>
  </info>

  <para>In this chapter, we'll fill in some background about pipelines
  and get you up to speed on the concepts before we dive into the actual
  features of XProc. We're going to assume that you're familiar with
  XML: you're comfortable the vocabulary of tags and attributes, parsing
  and validation, and that you've used an XML application or two: XSLT,
  for example.</para>

  <section xml:id="whatis">
    <title>What's a pipeline?</title>

    <para>At a high level of abstraction, a pipeline is what you get
    whenever the output of one process feeds into the input of another.
    If you work with XML, you've probably already used pipelines
    even if you didn't think about them in those terms. If you validate
    a document and then transform it, or if you apply XInclude and then
    validate, or if you use XSLT to run two different transformations,
    those are all simple "pipeline" operations.</para>

    <para>In another context, pipelines are a common feature of Unix
    command lines, for example:</para>

    <screen>cat ch01.xml | grep "<para" | wc -l</screen>

    <para>What that says is:</para>

    <orderedlist>
      <listitem>
        <para>Run the <command>cat</command> command over
        <filename>ch01.xml</filename>. That will display the contents of the file.
      </para>
      </listitem>
      <listitem>
        <para>The "|" symbol means that instead of displaying those lines in
        the shell window, that output will become the input to the
        <command>grep</command> command. What <command>grep</command> will do
        is display all the lines that contain "<literal><para></literal>"
        and discard all the rest.</para>
      </listitem>
      <listitem>
        <para>Finally, with another "|", the output of <command>grep</command> will
        become the input to <command>wc</command>. The <command>wc</command> command
        counts the words in a file, but we gave it the <option>-l</option> option, so
        it will count the lines instead.</para>
      </listitem>
    </orderedlist>

    <para>What this pipeline does is print the number of paragraphs in the
    document:</para>

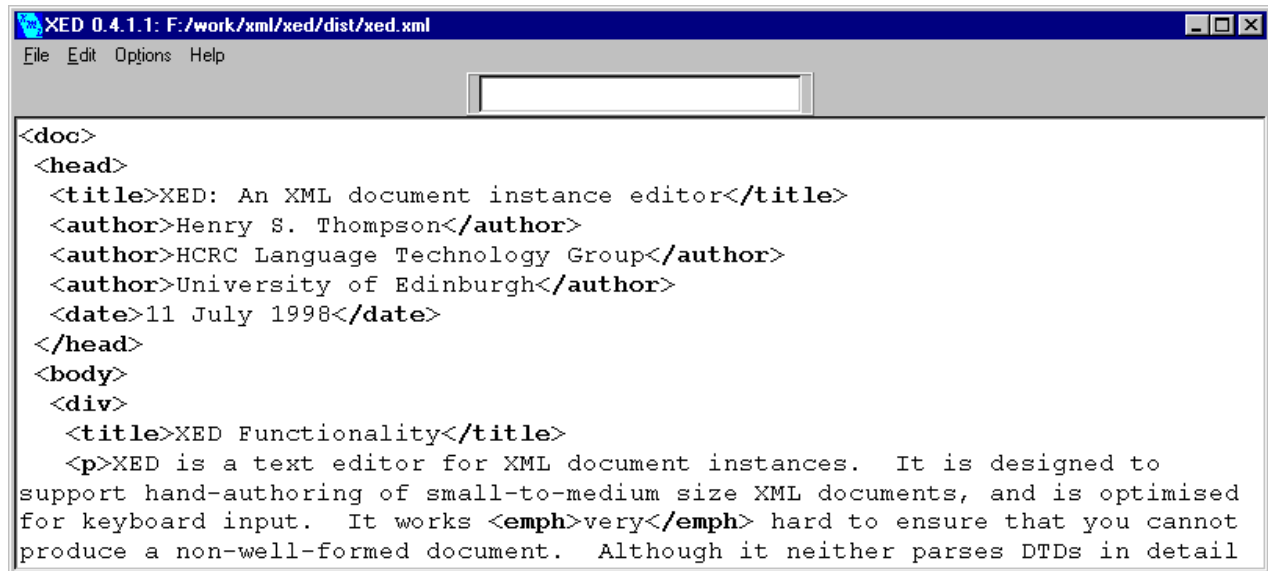
    <screen>??FIXME:??</screen>

    <para>By using a pipeline we've taken three useful utilities and
    composed them together to create a pipeline that will count the
    approximate number of paragraphs we've used in all the XML documents
    in the current directory. I say approximate because those are all line-based
  </chapter>
</?xml>
```

Slide 22 Copyright © 2011 MarkLogic Corporation



# What do we mean by editor?

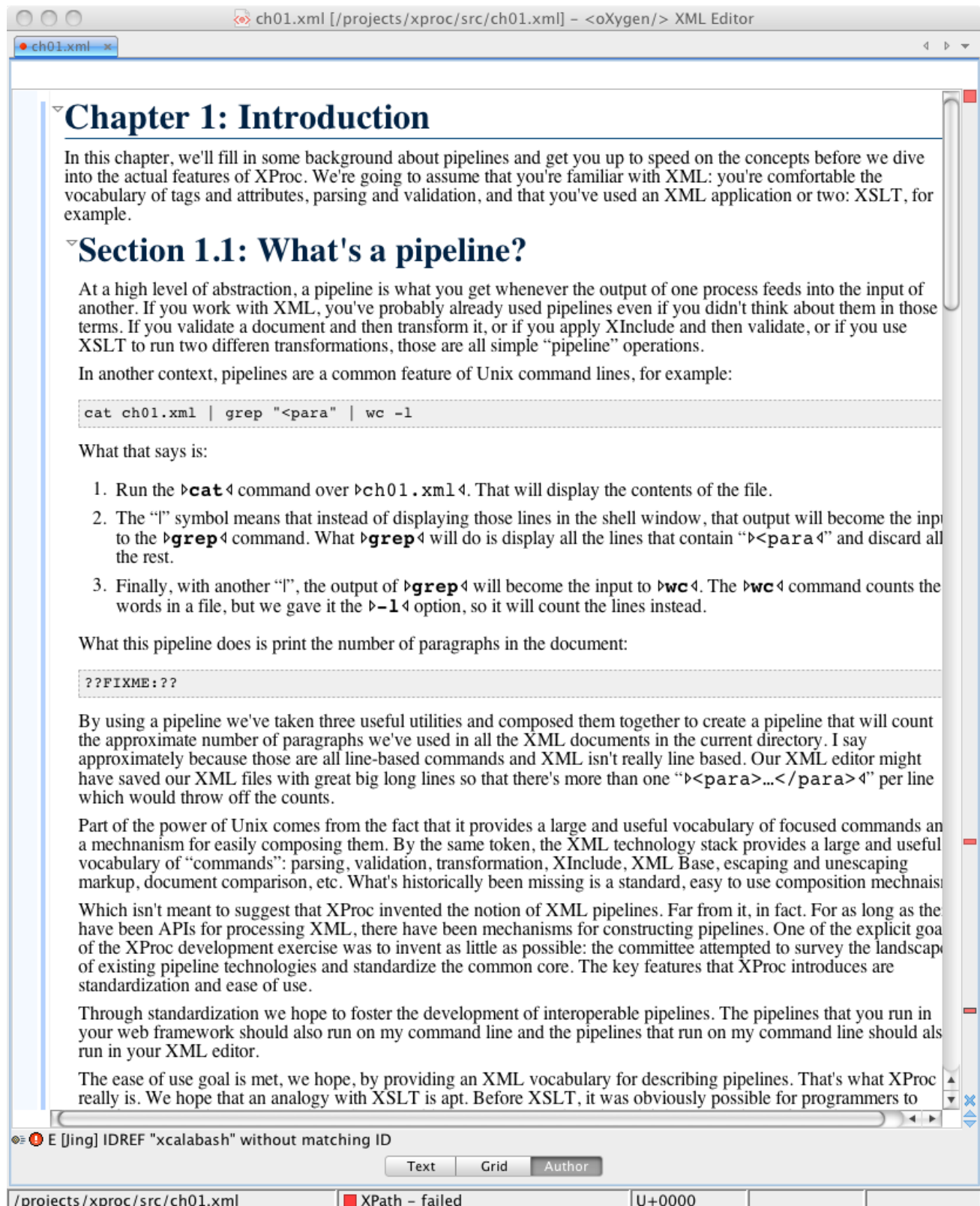


The screenshot shows a window titled "XED 0.4.1.1: F:/work/xml/xed/dist/xed.xml". The menu bar includes "File", "Edit", "Options", and "Help". The main text area contains the following XML document:

```
<doc>
  <head>
    <title>XED: An XML document instance editor</title>
    <author>Henry S. Thompson</author>
    <author>HCRC Language Technology Group</author>
    <author>University of Edinburgh</author>
    <date>11 July 1998</date>
  </head>
  <body>
    <div>
      <title>XED Functionality</title>
      <p>XED is a text editor for XML document instances. It is designed to
support hand-authoring of small-to-medium size XML documents, and is optimised
for keyboard input. It works <emph>very</emph> hard to ensure that you cannot
produce a non-well-formed document. Although it neither parses DTDs in detail
```



# What do we mean by editor?



The screenshot shows the oXygen XML Editor interface. The title bar indicates the file is `ch01.xml` located at `[/projects/xproc/src/ch01.xml]` and the application is `<oXygen/> XML Editor`. The editor window displays the following content:

## Chapter 1: Introduction

In this chapter, we'll fill in some background about pipelines and get you up to speed on the concepts before we dive into the actual features of XProc. We're going to assume that you're familiar with XML: you're comfortable the vocabulary of tags and attributes, parsing and validation, and that you've used an XML application or two: XSLT, for example.

### Section 1.1: What's a pipeline?

At a high level of abstraction, a pipeline is what you get whenever the output of one process feeds into the input of another. If you work with XML, you've probably already used pipelines even if you didn't think about them in those terms. If you validate a document and then transform it, or if you apply XInclude and then validate, or if you use XSLT to run two different transformations, those are all simple "pipeline" operations.

In another context, pipelines are a common feature of Unix command lines, for example:

```
cat ch01.xml | grep "<para" | wc -l
```

What that says is:

1. Run the `cat` command over `ch01.xml`. That will display the contents of the file.
2. The `|` symbol means that instead of displaying those lines in the shell window, that output will become the input to the `grep` command. What `grep` will do is display all the lines that contain `<para` and discard all the rest.
3. Finally, with another `|`, the output of `grep` will become the input to `wc`. The `wc` command counts the words in a file, but we gave it the `-l` option, so it will count the lines instead.

What this pipeline does is print the number of paragraphs in the document:

```
??FIXME??
```

By using a pipeline we've taken three useful utilities and composed them together to create a pipeline that will count the approximate number of paragraphs we've used in all the XML documents in the current directory. I say approximately because those are all line-based commands and XML isn't really line based. Our XML editor might have saved our XML files with great big long lines so that there's more than one `<para>...</para>` per line which would throw off the counts.

Part of the power of Unix comes from the fact that it provides a large and useful vocabulary of focused commands and a mechanism for easily composing them. By the same token, the XML technology stack provides a large and useful vocabulary of "commands": parsing, validation, transformation, XInclude, XML Base, escaping and unescaping markup, document comparison, etc. What's historically been missing is a standard, easy to use composition mechanism.

Which isn't meant to suggest that XProc invented the notion of XML pipelines. Far from it, in fact. For as long as there have been APIs for processing XML, there have been mechanisms for constructing pipelines. One of the explicit goals of the XProc development exercise was to invent as little as possible: the committee attempted to survey the landscape of existing pipeline technologies and standardize the common core. The key features that XProc introduces are standardization and ease of use.

Through standardization we hope to foster the development of interoperable pipelines. The pipelines that you run in your web framework should also run on my command line and the pipelines that run on my command line should also run in your XML editor.

The ease of use goal is met, we hope, by providing an XML vocabulary for describing pipelines. That's what XProc really is. We hope that an analogy with XSLT is apt. Before XSLT, it was obviously possible for programmers to

At the bottom of the editor, there is a status bar showing the file path `/projects/xproc/src/ch01.xml`, a red X icon with the text `XPath - failed`, and the position `U+0000`. A toolbar at the bottom includes buttons for `Text`, `Grid`, and `Author`. A message bar at the very bottom displays an error: `E [jing] IDREF "xcalabash" without matching ID`.



# What do we mean by editor?



```
1 <chapter xmlns="http://docbook.org/ns/docbook"
2       xmlns:cx="http://xmlcalabash.com/ns/extensions"
3       xmlns:xi="http://www.w3.org/2001/XInclude"
4       xmlns:xlink="http://www.w3.org/1999/xlink"
5       xmlns:p="http://www.w3.org/ns/xproc"
6       version="5.0-xproc" xml:id="introduction">
7   <info>
8     <title>Introduction</title>
9   </info>
10
11  <para>In this chapter, we'll fill in some background about pipelines
12  and get you up to speed on the concepts before we dive into the actual
13  features of XProc. We're going to assume that you're familiar with
14  XML: you're comfortable the vocabulary of tags and attributes, parsing
15  and validation, and that you've used an XML application or two: XSLT,
16  for example.</para>
17
18  <section xml:id="whatis">
19    <title>What's a pipeline?</title>
20
21    <para>At a high level of abstraction, a pipeline is what you get
22    whenever the output of one process feeds into the input of another.
23    If you work with XML, you've probably already used pipelines
24    even if you didn't think about them in those terms. If you validate
25    a document and then transform it, or if you apply XInclude and then
26    validate, or if you use XSLT to run two different transformations,
27    those are all simple "pipeline" operations.</para>
28
29    <para>In another context, pipelines are a common feature of Unix
30    command lines, for example:</para>
31
32    <screen>cat ch01.xml | grep "<para" | wc -l</screen>
33
34    <para>What that says is:</para>
35
36    <orderedlist>
37      <listitem>
38        <para>Run the <command>cat</command> command over
39        <filename>ch01.xml</filename>. That will display the contents of the file.
40      </para>
41    </listitem>
42    <listitem>
43      <para>The "|" symbol means that instead of displaying those lines in
44      the shell window, that output will become the input to the
45      <command>grep</command> command. What <command>grep</command> will do
46      is display all the lines that contain "<literal><para></literal>"
47      and discard all the rest.</para>
48    </listitem>
49    <listitem>
50      <para>Finally, with another "|", the output of <command>grep</command> will
51      become the input to <command>wc</command>. The <command>wc</command> command
52      counts the words in a file, but we gave it the <option>-l</option> option, so
53      it will count the lines instead.</para>
54    </listitem>
55  </orderedlist>
56</chapter>
```

E [Jing] element "att" not allowed anywhere; expected the element end-tag, text or element "abbrev", "accel", "acronym", "address"

Text Grid Author

/projects/xproc/src/ch01.xml XPath - failed U+003C 1:1 Modified



# What do we mean by editor?

The screenshot shows the oXygen XML Editor interface. The main window displays a hierarchical tree view of an XML document. The tree structure is as follows:

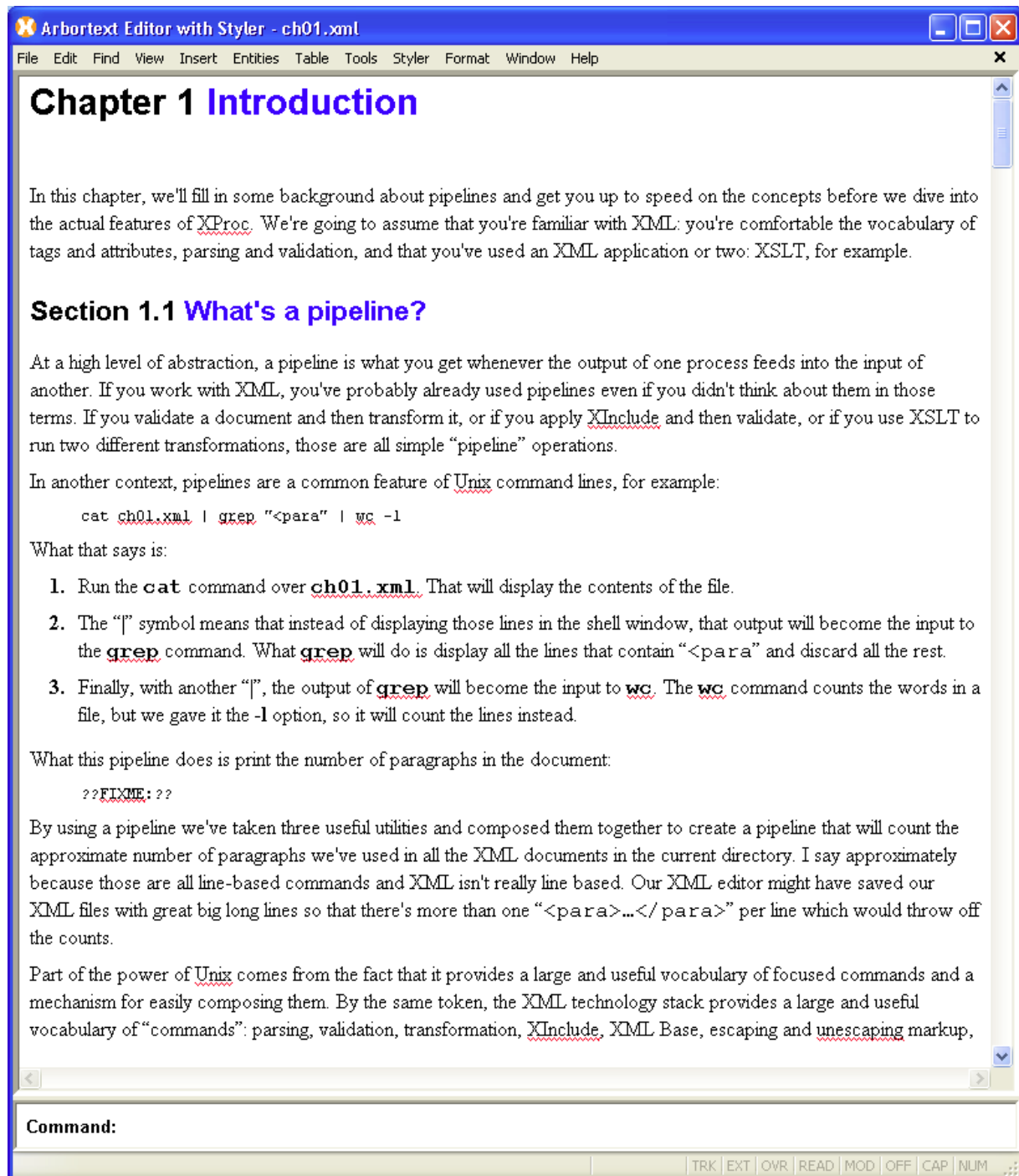
- chapter**
  - @xmlns: http://docbook.org/ns/docbook
  - @xmlns:cx: http://xmlcalabash.com/ns/extensions
  - @xmlns:xi: http://www.w3.org/2001/XInclude
  - @xmlns:xlink: http://www.w3.org/1999/xlink
  - @xmlns:p: http://www.w3.org/ns/xproc
  - @version: 5.0-xproc
  - @xml:id: introduction
  - info**
    - title: Introduction
  - para**

In this chapter, we'll fill in some background about pipelines and get you up to speed on the concepts before we dive into the actual features of XProc. We're going to assume that you're familiar with XML: you're comfortable the vocabulary of tags and attributes, parsing and validation, and that you've used an XML application or two: XSLT, for example.
  - section (7 rows)**

@xml:id	title	para	screen	ite		
1 whatis	What's a pipeline?	<b>para (2 rows)</b> <table border="1"><thead><tr><th>#text</th></tr></thead><tbody><tr><td>1 At a high level of abstraction, a pipeline is what you get whenever the output of one process feeds into the input of another. If you work with XML, you've probably already used pipelines even if you didn't think about them in those terms. If you validate a document and then transform it, or if you apply XInclude and then validate, or if you use XSLT to run two different transformations, those are all simple "pipeline"</td></tr></tbody></table>	#text	1 At a high level of abstraction, a pipeline is what you get whenever the output of one process feeds into the input of another. If you work with XML, you've probably already used pipelines even if you didn't think about them in those terms. If you validate a document and then transform it, or if you apply XInclude and then validate, or if you use XSLT to run two different transformations, those are all simple "pipeline"	cat ch01.xml   grep "<para"   wc -l	
#text						
1 At a high level of abstraction, a pipeline is what you get whenever the output of one process feeds into the input of another. If you work with XML, you've probably already used pipelines even if you didn't think about them in those terms. If you validate a document and then transform it, or if you apply XInclude and then validate, or if you use XSLT to run two different transformations, those are all simple "pipeline"						



# What do we mean by editor?



Arbortext Editor with Styler - ch01.xml

File Edit Find View Insert Entities Table Tools Styler Format Window Help

## Chapter 1 Introduction

In this chapter, we'll fill in some background about pipelines and get you up to speed on the concepts before we dive into the actual features of XProc. We're going to assume that you're familiar with XML: you're comfortable the vocabulary of tags and attributes, parsing and validation, and that you've used an XML application or two: XSLT, for example.

### Section 1.1 What's a pipeline?

At a high level of abstraction, a pipeline is what you get whenever the output of one process feeds into the input of another. If you work with XML, you've probably already used pipelines even if you didn't think about them in those terms. If you validate a document and then transform it, or if you apply XInclude and then validate, or if you use XSLT to run two different transformations, those are all simple "pipeline" operations.

In another context, pipelines are a common feature of Unix command lines, for example:

```
cat ch01.xml | grep "<para" | wc -l
```

What that says is:

1. Run the `cat` command over `ch01.xml`. That will display the contents of the file.
2. The `|` symbol means that instead of displaying those lines in the shell window, that output will become the input to the `grep` command. What `grep` will do is display all the lines that contain `<para` and discard all the rest.
3. Finally, with another `|`, the output of `grep` will become the input to `wc`. The `wc` command counts the words in a file, but we gave it the `-l` option, so it will count the lines instead.

What this pipeline does is print the number of paragraphs in the document:

```
??FIXME:??
```

By using a pipeline we've taken three useful utilities and composed them together to create a pipeline that will count the approximate number of paragraphs we've used in all the XML documents in the current directory. I say approximately because those are all line-based commands and XML isn't really line based. Our XML editor might have saved our XML files with great big long lines so that there's more than one `<para>...</para>` per line which would throw off the counts.

Part of the power of Unix comes from the fact that it provides a large and useful vocabulary of focused commands and a mechanism for easily composing them. By the same token, the XML technology stack provides a large and useful vocabulary of "commands": parsing, validation, transformation, XInclude, XML Base, escaping and unescaping markup,

Command:

TRK | EXT | OVR | READ | MOD | OFF | CAP | NUM



# What do we mean by editor?

Arbortext Editor with Styler - ch01.xml

File Edit Find View Insert Entities Table Tools Styler Format Window Help

chapter xmlns:cx="http://xmlcalabash.com/ns/extensions" xmlns:xi="http://www.w3.org/2001/XInclude" xmlns:xlink="http://www.w3.org/1999/xlink" x

**Chapter 1 Introduction**

In this chapter, we'll fill in some background about pipelines and get you up to speed on the concepts before we dive into the actual features of XProc. We're going to assume that you're familiar with XML: you're comfortable the vocabulary of tags and attributes, parsing and validation, and that you've used an XML application or two: XSLT, for example.

**Section 1.1 What's a pipeline?**

At a high level of abstraction, a pipeline is what you get whenever the output of one process feeds into the input of another. If you work with XML, you've probably already used pipelines even if you didn't think about them in those terms. If you validate a document and then transform it, or if you apply XInclude and then validate, or if you use XSLT to run two different transformations, those are all simple "pipeline" operations.

In another context, pipelines are a common feature of Unix command lines, for example:

```
cat ch01.xml | grep "<para" | wc -l
```

What that says is:

- Run the `cat` command over `ch01.xml`. That will display the contents of the file.
- The `|` symbol means that instead of displaying those lines in the shell window, that output will become the input to the `grep` command. What `grep` will do is display all the lines that contain `<para` and discard all the rest.
- Finally, with another `|`, the output of `grep` will become the input to `wc`. The `wc` command counts the words in a file, but we gave it the `-l` option, so it will count the lines instead.

What this pipeline does is print the number of paragraphs in the document:

```
??FIXME: ??
```

By using a pipeline we've taken three useful utilities and composed them together to create a pipeline that will count the approximate number of paragraphs we've used in all the XML documents in the current directory. I say approximately because those are all line-based commands and XML isn't really line based. Our XML editor might have saved our XML files with great big long lines so that there's more than one `<para>...</para>` per line which would throw off the counts.

Part of the power of Unix comes from the fact that it provides a large and useful vocabulary of focused

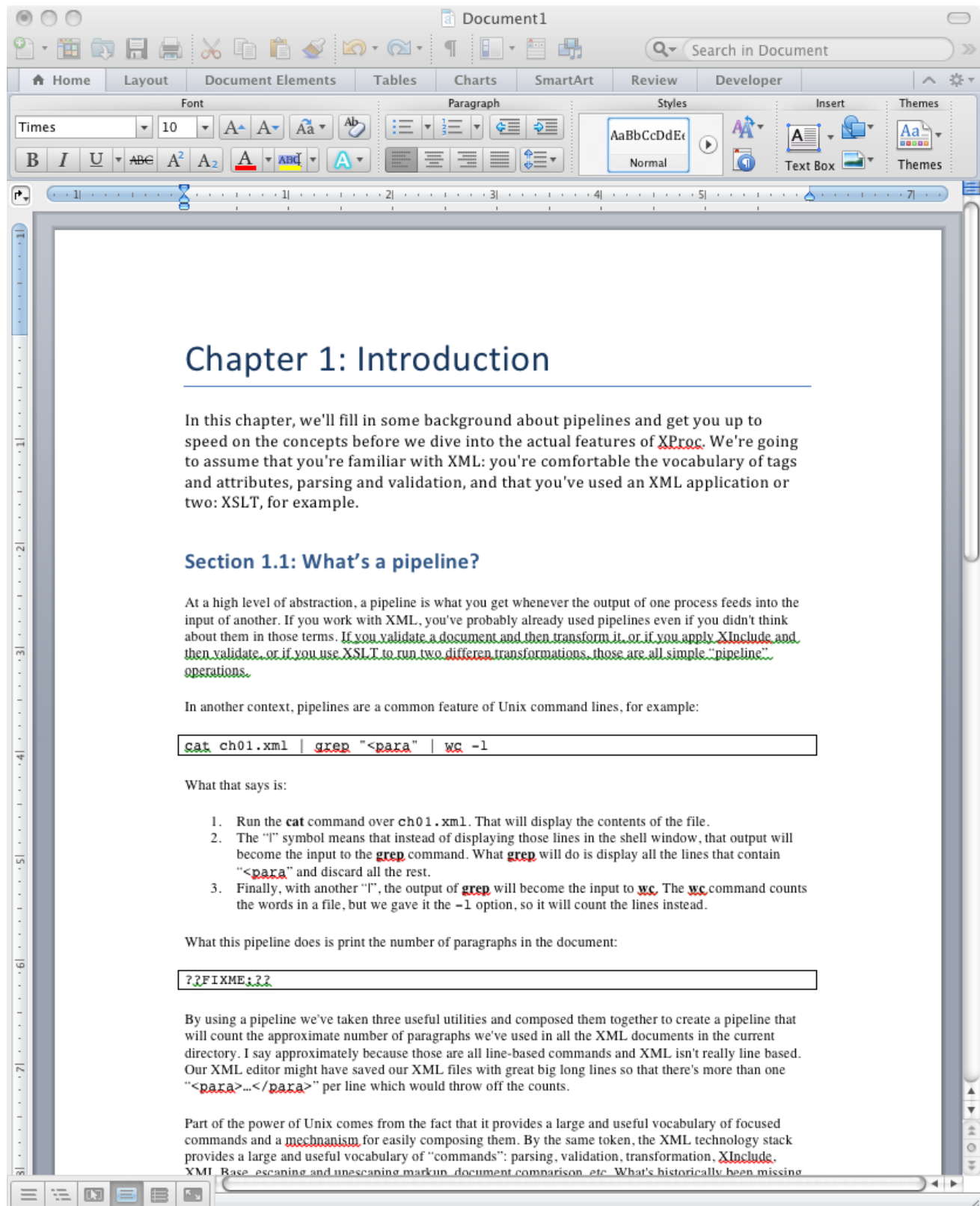
Command:

chapter | info title

TRK | EXT | OVR | READ | MOD | OFF | CAP | NUM



# What do we mean by editor?





# Aside: WordML

---

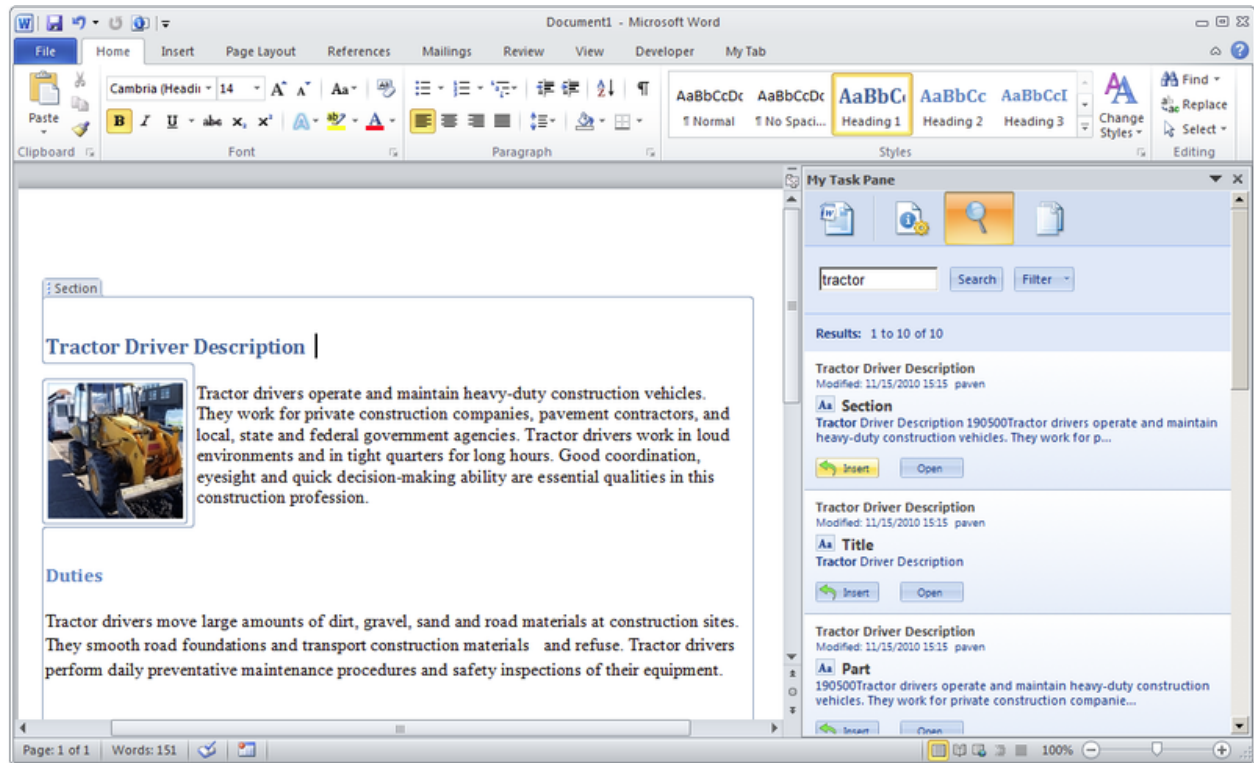
Or, “is this what we mean by XML!”



/projects/presentations/2011/08-balisa...

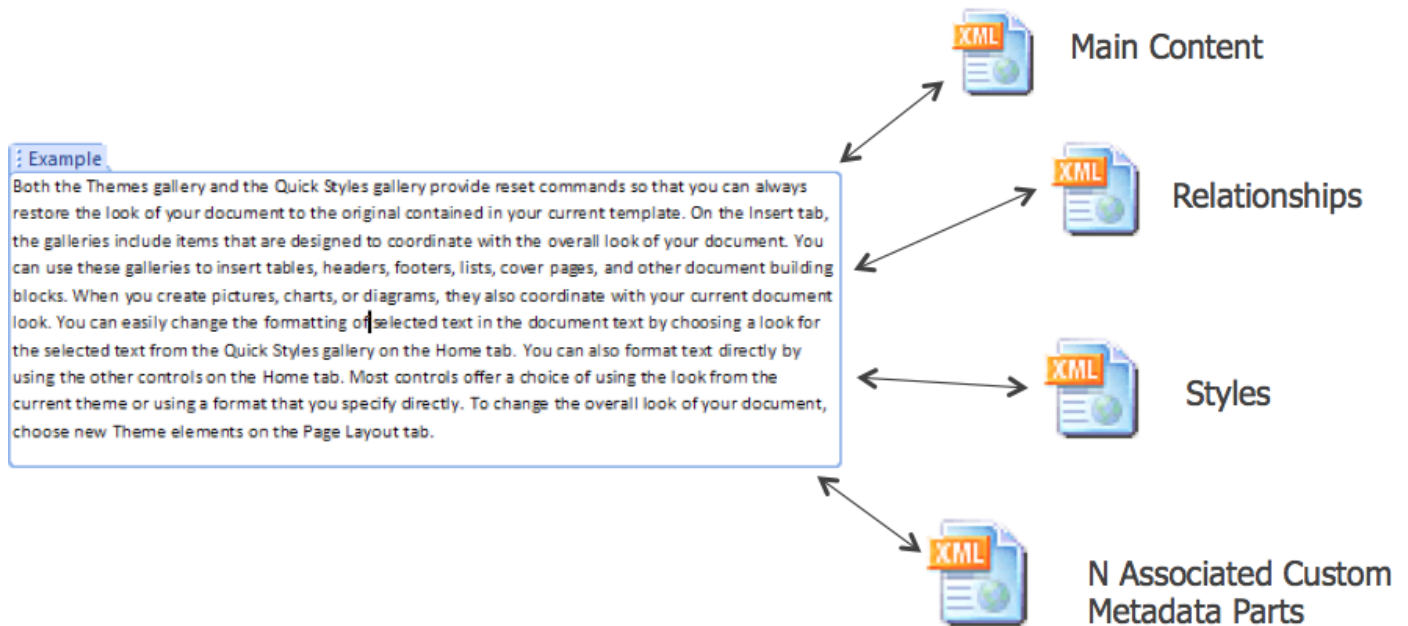


# MarkLogic Toolkit for Word





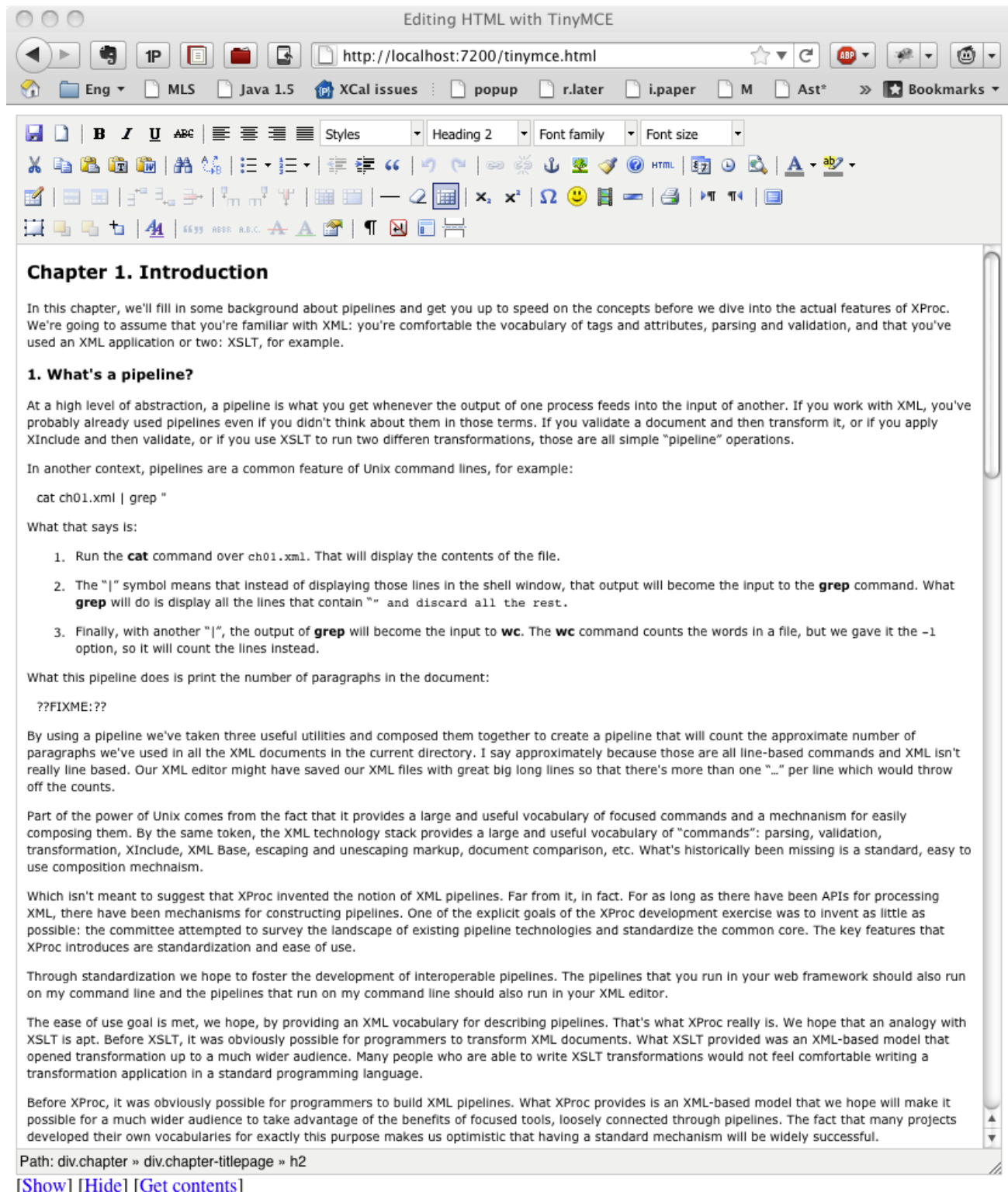
# Word Components



Try it [yourself](#)



# What do we mean by editor?





# Aside: (X)HTML(5)

---

Or, “is this what we mean by XML!”



```
chapter1.html x
view-source:file:///tmp/chapter1.html

15 <div class="chapter" id="introduction">
16 <div class="chapter-titlepage">
17 <h2>Chapter 1. Introduction</h2>
18 </div>
19 <p>In this chapter, we'll fill in some background about pipelines
20 and get you up to speed on the concepts before we dive into the actual
21 features of XProc. We're going to assume that you're familiar with
22 XML: you're comfortable the vocabulary of tags and attributes, parsing
23 and validation, and that you've used an XML application or two: XSLT,
24 for example.
25 </p>
26 <div class="section" id="whatis">
27 <div class="section-titlepage">
28 <h3>1. What's a pipeline?</h3>
29 </div>
30 <p>At a high level of abstraction, a pipeline is what you get
31 whenever the output of one process feeds into the input of another.
32 If you work with XML, you've probably already used pipelines
33 even if you didn't think about them in those terms. If you validate
34 a document and then transform it, or if you apply XInclude and then
35 validate, or if you use XSLT to run two different transformations,
36 those are all simple "pipeline" operations.
37 </p>
38 <p>In another context, pipelines are a common feature of Unix
39 command lines, for example:
40 </p>
41 <div class="screen"><pre xml:space="preserve"><span class="linenumber">
1</span><span class="linenumber-separator"> </span>cat ch01.xml | grep "<para" | wc -
1</pre></div>
42 <p>What that says is:</p>
43 <div class="orderedlist">
44 <ol style="list-style: decimal;">
45 <li>
46 <p>Run the <strong class="command">cat</strong> command over
47 <tt class="filename">ch01.xml</tt>. That will display the contents of the
file.
48 </p>
49 </li>
50 </ol>
51 <li>
52 <p>The "|" symbol means that instead of displaying those lines in
53 the shell window, that output will become the input to the
54 <strong class="command">grep</strong> command. What <strong
class="command">grep</strong> will do
55 is display all the lines that contain "<tt class="literal"><para</tt>"
56 and discard all the rest.
57 </p>
58 </li>
59 <li>
60 <p>Finally, with another "|", the output of <strong
class="command">grep</strong> will
61 become the input to <strong class="command">wc</strong>. The <strong
class="command">wc</strong> command
62 counts the words in a file, but we gave it the <tt class="option">-l</tt>
option, so
63 it will count the lines instead.
64 </p>
65 </li>
66 </ol>
67 </div>
68 <p>What this pipeline does is print the number of paragraphs in the
69 document:
70 </p>
```



# What do we mean by editor?

Or, “what was that driving question all about?”



Photo credit: [Doug Wertzman](#)



# What do we mean by people?

- You and me?
- Professional writers?
- Engineers?
- Interns?
- Managers?



# What do we mean by people?

- XML experts
  - I'll see your CDATA section and raise you an external parsed entity
- XML users
  - I'll see your empty element and raise you a processing instruction
- Users being presented with XML for the first time
  - What's an XML?



# XML experts

- Understand the separation of content and presentation
- Believe in the value of structured markup
- Can leverage XML to deliver value

What editing problem?



# XML users

- Understand the separation of content and presentation
- Have been told about the value of structured markup
- May be seeing some value from structured markup
- May feel threatened by the loss of control

Possibly comfortable with oXygen/Arbortext Editor/etc.



# First time XMLers

- May have been told about the separation of content and presentation
- May not understand the value of structured markup
- May be threatened or frightened by the prospect of changing editing tools
- Probably think in terms of document presentation

Think everyone uses Microsoft Word.



# Aside: presentation

---

- “Probably think in terms of document presentation”



# The presentation problem

Fact:

- Many users consider their branded output as primary, XML as secondary

Reality:

- Rendering arbitrary XML to high-quality print is hard
- “Lights out” formatting is fundamentally different than “by hand” layout
- Automatically rendering arbitrary XML to an arbitrary design is at least hard, if not impossible



# The other presentation problem

- The rise of desktop publishing, and the WYSIWYG word processors that followed, have trained a generation (or more) of authors to believe that the important thing about a document is how it looks
- Authors believe that they add value by controlling the appearance of the document



# A tale of two engagements

In which the presenter rambles on a bit



**What will it take...**

---

# Education



# Editing XML

- Isn't harder just because the tools suck.
- It requires authors
  - to think differently about content
  - to know more about the content
  - to *unlearn* a set of skills of which they may be rightfully proud and for which they have historically been rewarded
  - to accept that they have less control over the presentation
- It requires managers
  - to change the structure of rewards for authors
  - to accept that the problems are harder
  - to understand that great rewards are possible, but they're not free



# Or did you mean...

What will it take to get

- An XML editor that provides a tailored user interface for any vocabulary
- With a high-quality preview and “tags off” editing
- That supports semantic markup without any additional effort
- And can be used out-of-the-box without any customizaiton



**That will take**

---

**A miracle**



# Conclusions

- For a lot of use cases and a lot of users, we have editors that don't suck. Or at least, don't suck as much as you might at first glance think they do.
- Tools like oXygen, Arbortext Editor (Serna, Altova, etc.) work reasonably well for many kinds of users.
- If you're willing to squint at the XML a bit, tools like Word and OpenOffice work today
- If you want to do better, it's as much about education as technology



# Q&A

Norman Walsh

<mailto:norman.walsh@marklogic.com>

tel: +1-413-624-6676

<http://www.marklogic.com/>





# Table of contents

- Any questions?
- But, seriously...
- Survey question: 1 of 9
- Survey question: 2 of 9
- Survey question: 3 of 9
- Survey question: 4 of 9
- Survey question: 5 of 9
- Survey question: 6 of 9
- Survey question: 7 of 9
- Survey question: 8 of 9
- Survey question: 9 of 9
- What do we really mean?
- What do we mean by XML?
- What do we mean by XML?
- What do we mean by XML?
- What do we mean by XML?
- What do we mean by XML?
- What do we mean by XML?
- What do we mean by editor?
- What do we mean by editor?
- What do we mean by editor?
- What do we mean by editor?
- What do we mean by editor?
- What do we mean by editor?
- What do we mean by editor?
- What do we mean by editor?
- What do we mean by editor?
- What do we mean by editor?
- What do we mean by editor?
- Aside: WordML
- MarkLogic Toolkit for Word
- Word Components



- Aside: (X)HTML(5)
- What do we mean by editor?
- What do we mean by people?
- What do we mean by people?
- XML experts
- XML users
- First time XMLers
- Aside: presentation
- The presentation problem
- The other presentation problem
- A tale of two engagements
- What will it take...
- Editing XML
- Or did you mean...
- That will take
- Conclusions
- Q&A