Accessibility

what it is about

context **2024** meeting

What is accessibility

Accessible pdf documents are a somewhat hot topic (for a while). Here are some definitions:

- Greenwich: An accessible document is a document created to be as easily readable by a low vision or non-sighted reader as a sighted reader.
- Harvard: Accessible documents are easier to understand and read for all of your users, not just users with disabilities.
- University of California San Francisco: An accessible digital document is well-structured, providing visual information in a non-visual format.
- Carlton: Accessible documents provide all text and other elements in an accessible format, so that everyone can access the information in the documents in some manner.

Such definitions are often followed by a similar list of suggestions, likely taken from some (government) directive.

Note: This talk is a variant on the one done at BachoTEX 2024 but most examples are the same!

What is tagging

Tagging adds information to a pdf file so that:

- **content can be extracted**: apart from basic copying we're not interested in this
- **the text can reflow:** use an other format is that is needed
- **text can be spoken:** to some extend that can be useful

But it comes a at cost:

- There are **no real good free tools** that handle it and validation, fixing, standards with respect to pdf has always been a somewhat commercial enterprise.
- The standard is a **confusing**, and interpretation gets debated: it looks like little research went ahead of it.

- So we can best just start from **common sense and usage** and also realize that in the end (future) demands are different anyway (compare book printing).
- Nevertheless, we always end up with a **bloated** pdf file, which kind of contradicts other efforts to be lean and mean.

And so . . .

- We basically end up implementing a feature for the sake of the feature that might be useful in the future.
- And that in the end might not work out as intended as it might be suboptimal.
- And we can not check its usability so it's mostly about conformance and playing safe.
- Also, we operate in a fast moving world when it comes to demands, presentation models, usage and maybe coming technologies that might make this obsolete.

Examples

So what are the consequences of tagging for a pdf file? Let's have a look at some simple examples.

untagged: test: $\boxed{x^2 = 4}$!

tagged: $x^2 = 4$

tagged: test: $x^2 = 4$

tagged: test: $\boxed{x^2 = 4}$!

```
stream
0 g 0 G
BT
/F1 10 Tf
1.195517 0 0 1.195517 3.941792 7.979264 Tm [<000100020003000100040005>] TJ
/F2 10 Tf
1.195517 0 0 1.195517 33.563574 7.979264 Tm [<0001>] TJ
0.836858 0 0 0.836858 40.398174 12.914036 Tm [<0002>] TJ
1.195517 0 0 1.195517 48.485127 7.979264 Tm [<0003>-278<0004>] TJ
ET
q
1 0 0 1 32.076871 2.455088 cm
[] 0 d 0 J 0.3985 w 0 0 36.486178 17.507437 re
                                                                S

Q
BT
/F1 10 Tf
1.195517 0 0 1.195517 72.741182 7.979264 Tm [<0006>] TJ
ET
0 g 0 G
endstream
```

test: $x^2 = 4$!

```
stream
0 g 0 G
/math <</MCID 1>> BDC
BT
/F1 10 Tf
1.195517 0 0 1.195517 3.941792 4.073226 Tm [<0001>] TJ
0.836858 0 0 0.836858 10.776392 9.007999 Tm [<0002>] TJ
1.195517 0 0 1.195517 18.863344 4.073226 Tm [<0003>-278<0004>] TJ
ET
EMC
0 g 0 G
endstream
```

$x^2 = 4$

```
stream
0 g 0 G
/documentpart <</MCID 1>> BDC
BT
/F1 10 Tf
1.195517 0 0 1.195517 3.941792 4.073226 Tm [<0001000200030010004>] TJ
ET
EMC
/math <</MCID 2>> BDC
BT
/F2 10 Tf
1.195517 0 0 1.195517 31.877621 4.073226 Tm [<0001>] TJ
0.836858 0 0 0.836858 38.712221 9.007999 Tm [<0002>] TJ
1.195517 0 0 1.195517 46.799174 4.073226 Tm [<0003>-278<0004>] TJ
ET
EMC
0 g 0 G
endstream
```

test: $x^2 = 4$

```
stream
0 g 0 G
/documentpart <</MCID 1>> BDC
BT
/F1 10 Tf
1.195517 0 0 1.195517 3.941792 7.979264 Tm [<0001000200030010004>] TJ
ET
EMC
/math <</MCID 2>> BDC
BT
/F2 10 Tf
1.195517 0 0 1.195517 33.563574 7.979264 Tm [<0001>] TJ
0.836858 0 0 0.836858 40.398174 12.914036 Tm [<0002>] TJ
1.195517 0 0 1.195517 48.485127 7.979264 Tm [<0003>-278<0004>] TJ
ET
EMC
/Artifact BMC
q
1 0 0 1 32.076871 2.455088 cm
[] 0 d 0 J 0.3985 w 0 0 36.486178 17.507437 re
                                                              S
Q
EMC
```

test: $\boxed{x^2 = 4}$!

```
/documentpart <</MCID 3>> BDC
BT
/F1 10 Tf
1.195517 0 0 1.195517 72.741182 7.979264 Tm [<0005>] TJ
ET
EMC
0 g 0 G
endstream
```

We need a lot so tracing options to figure out possible issues, like:

```
backend         > tags > begin page
backend         > tags >
backend         > tags > P    11 document>1 documentpart>1 navigationpage>1 : 1
backend         > tags > T     2 document>1 documentpart>1 : test:
backend         > tags > T     3 document>1 documentpart>1 math>1 : [2] = 4
backend         > tags >   -----
backend         > tags > T     2 document>1 documentpart>1 : !
backend         > tags >
backend         > tags > end page
backend         > tags >

backend         > tags >     1    1  document>1 (content)
backend         > tags >     2    1  document>1 documentpart>1 (content)
backend         > tags >     3    1  document>1 documentpart>1 navigationpage>1 (content)
backend         > tags >     4    1  document>1 documentpart>1 math>1 (content)
```

But we also have visual clues: tag labels, suspects, etc.

Checking if we're okay

- We can look at the file and if it opens in viewers we know that we didn't mess up too badly. Looking at the pdf in an editor also works.
- The VeraPDF checker can be used but it's not always reliable. The order of reported issues can differ per run and when you fixed the last issue, suddenly a new one can be shown. (There are two parsers to choose from and results can differ.)
- The PAC 2021 checker is more powerful but hasn't been updated to handle pdf 2.0 (we can hack around that) an dit doesn't handle the role maps. But it has a nice preview, shows a tag tree, etc. It's a bit slow in analyzing.
- We're only interested in the file being okay because there is not way to know what is needed. We don't relate to pseudo html but users can do that if they want. We don't want to cook up something sub-optimal.

- As long as we add meaningful tags, we can expect future document analyzer to do a decent job, after all a ‘section’ says what it is.

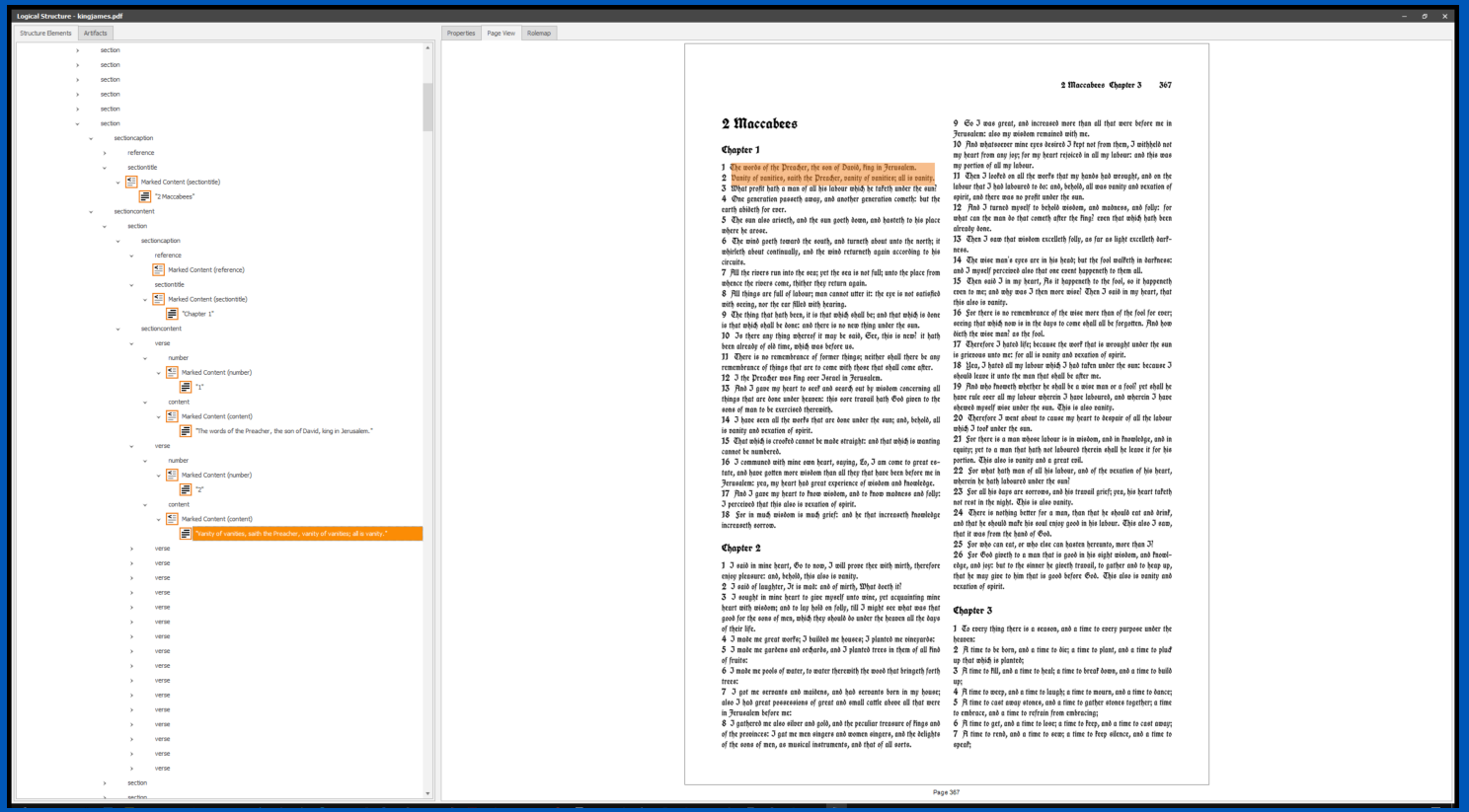
Structure, meaning and rolemaps (1)

Old Testament

sectiontitle
reference

Genesis 5
listcontent listpage
Exodus 36
listcontent listpage
Leviticus 62
listcontent listpage
Numbers 81
listcontent listpage
Deuteronomy 108
listcontent listpage
Joshua 130
listcontent listpage
Judges 145
listcontent listpage
Ruth 160
listcontent listpage
1 Samuel 162

Structure, meaning and rolemaps (2)



cessibility — context **2024** meeting — Structure, meaning and rolemaps

Structure, meaning and rolemaps (3)

Let's get an idea what we're dealing with. You can forget about it after seeing it. The real content is this, when untagged we also have more efficient text streams (here between <>):

```
stream
0 g 0 G
BT
/F1 10 Tf
0.996264 0 0 0.996264 549.598217 791.184973 Tm [<0001>] TJ
2.066252 0 0 2.066252 42.097049 741.603508 Tm
                                        [<0002000300040005000600070008000900 0A000B000C>] TJ
/F2 10 Tf
0.996264 0 0 0.996264 42.097049 710.081548 Tm
                                    [<0001000200030004000500060007000800090 00A0006000B0008>] TJ
0.996264 0 0 0.996264 548.192356 710.081548 Tm [<000C>] TJ
0.996264 0 0 0.996264 42.097049 689.160004 Tm
                                    [<000D0006000E0004000500060007000800090 00A0006000B0008>] TJ
0.996264 0 0 0.996264 541.114406 689.160004 Tm [<000F00100010>] TJ
ET
```

0 g 0 G
endstream

cessibility — context **2024** meeting — Structure, meaning and rolemaps

When we tag we get entries like this in the page stream:

```
0 g 0 G
/Artifact BMC
BT
/F1 10 Tf
0.996264 0 0 0.996264 549.598217 791.184973 Tm [<0001>]
                                                    TJ
ET
EMC
/documentpart <</MCID 1>> BDC
BT
/F1 10 Tf
2.066252 0 0 2.066252 42.097049 741.603508 Tm
    [<000200030004000500060007000800090000A000B000C>] TJ
ET
EMC
/link <</MCID 2>> BDC
EMC
/listcontent <</MCID 3>> BDC
BT
/F2 10 Tf
0.996264 0 0 0.996264 42.097049 710.081548 Tm
[<000100020003000400050006000700080009000A0006000B0008>]
                                                    TJ
ET
EMC
/listpage <</MCID 4>> BDC
BT
/F2 10 Tf
0.996264 0 0 0.996264 548.192356 710.081548 Tm [<000C>]
                                                    TJ
ET
EMC
/link <</MCID 5>> BDC
EMC
/listcontent <</MCID 6>> BDC
BT
/F2 10 Tf
0.996264 0 0 0.996264 42.097049 689.160004 Tm
[<000D0006000E000400050006000700080009000A0006000B0008>]
                                                    TJ
ET
EMC
/listpage <</MCID 7>> BDC
BT
/F2 10 Tf
0.996264 0 0 0.996264 541.114406 689.160004 Tm
                                        [<000F00100010>] TJ
ET
EMC
0 g 0 G
endstream
```

The /MCID 3 points into an array related to the page. Let's start at the top parent (676):

```
676 0 obj
    <<
        /K          103359 0 R
        /Namespaces [ 678 0 R 681 0 R 682 0 R
                                                                    ]
        /ParentTree 677 0 R
        /Type       /StructTreeRoot
    >>
endobj
```

The top level kids array (103359) is

```
103359 0 obj
[ 683 0 R ]
endobj
```

The first entry (683) brings us to the document level

```
683 0 obj
    <<
        /K  [ 684 0 R ]
```

```
        /NS 678 0 R
        /P  676 0 R
        /Pg 1 0 R
        /S  /document
    >>
endobj
```

This element has only one kid (684) and sits in a name space (678). The parent is (676) a way to get back, the page object is also references (1).

```
678 0 obj
    <<
        /LMTXNameSpace /context
        /NS            <feff.....>>
        /RoleMapNS     103357 0 R
        /Type          /Namespace
    >>
endobj
```

The name space points to a role map (103357, we have many objects here) so we can use nice names as we like. We map most on the default `NonStruct` as the regular subset makes little sense for us.

```
103357 0 obj
    <<
        /document     [ /Document  681 0 R ]
        /documentpart [ /NonStruct 681 0 R ]
        /link         [ /Link      681 0 R ]
        /list         [ /NonStruct 681 0 R ]
        /listcontent  [ /NonStruct 681 0 R ]
```

```
        /listitem     [ /NonStruct 681 0 R ]
        ...
    >>
endobj
```

The mapped ones come from, a default set defines in (681):

```
681 0 obj
    <<
        /LMTXNameSpace /ua2
        /NS            <feff....>
        /Type          /Namespace
    >>
endobj
```

Back to the mapping from elements on the page to real ones:

```
677 0 obj
<<
    /Nums [
          0 [ 685 0 R 684 0 R 688 0 R 689 0 R 690 0 R 692 0 R 693 0 R 694 0 R ]
          1 [ 704 0 R ]
          2 [ .... ]
        ...
        738 77343 0 R
        739 77347 0 R
    ]
>>
endobj
```

The second element on the page (684) is:

```
684 0 obj
<<
    /K [ 685 0 R 1 686 0 R ... ]
    /NS 678 0 R
    /P 683 0 R
    /Pg 1 0 R
    /S /documentpart
>>
endobj
```

The kids can be followed (from 676) to (684):

```
684 0 obj
<<
    /K [ 685 0 R 1 686 0 R .... ]
    /NS 678 0 R
    /P 683 0 R
    /Pg 1 0 R
    /S /documentpart
>>
endobj
```

We go all the way down to:

```
686 0 obj
<< /K [ 687 0 R 691 0 R ] /NS 678 0 R /P 684 0 R /Pg 1 0 R /S /list >>
endobj
687 0 obj
<< /K [ 688 0 R 689 0 R 690 0 R ] /NS 678 0 R /P 686 0 R /Pg 1 0 R /S /listitem /T (chapter)
                                                                                >>
endobj
688 0 obj
<< /K [ 2 ] /NS 678 0 R /P 687 0 R /Pg 1 0 R /S /link >>
endobj
689 0 obj
<< /K [ 3 ] /NS 678 0 R /P 687 0 R /Pg 1 0 R /S /listcontent >>
endobj
690 0 obj
<< /K [ 4 ] /NS 678 0 R /P 687 0 R /Pg 1 0 R /S /listpage >>
endobj
691 0 obj
<< /K [ 692 0 R 693 0 R 694 0 R ] /NS 678 0 R /P 686 0 R /Pg 1 0 R /S /listitem /T (chapter)
                                                                                >>
endobj
```

And so on. Keep in mind that in the page stream we see the endpoints and in order to see where they come from one has to follow the chain back!

Annoyances

- One has to mark everything. There is no default to artifact, which would save a lot of (time and) file size as well as checking.
- Unicode lacks a code point that represents “no character, just ignore me when copying or speaking” so one has to mark private slots as artifact which is pain and dirties the backend.
- There are no code points that can help the speech engine, like pauses. One can argue that this should not be in Unicode but we do have linguistic and plenty odd symbols anyway.
- Often a nice looking and educational rich document has more than just text, otherwise one could as well emulate a typewriter. It's also about motivating and attraction. So there might be hard to catch artifacts.

- Validating can be fragile, so one never knows for sure if what is okay or bad today is bad or okay tomorrow. But we can decide to ignore some warnings, especially when it hard to explain why it matters.
- There are some weird demands. Why should for instance a hyperlink mark as artifact still resolve to a destination. Also, one assumes viewers to to not adapt so there are redundant entries (for no real reason like `/D` and `/SD` in destinations).

Math

- Math tagging is somewhat complex and often **domain dependent** the current state made us decide to just do what we think is best.
- As with math fonts it's not the TEX community that drives it (although of course there has been early adoption and feedback, e.g. by Ross Moore). We just have to **follow the trends**.
- We **always** had some kind of support for tagged math, not that there were applications out there that we could check it with.
- At EuroBachoTEX 2017 there has been **ambitious plans** for future projects with respect to tagged pdf (mentioning involvement of publishers and substantial funding) but if that happens it is outside the ConTEXt community scope.

- So . . . we just **go our own way** and ‘ritmik’ is what we came up with, which actually is a side track of our math upgrading project.

- Sidenote: we do the same with bibliographies but that is much simpler: **serialize** citations and embed bibTEX data.

How

- We decided to go for what we call “meaningful math”: instead of relying on unknown technology we make sure that when gets ‘read out’ reflects our intentions: we provide **serialized math** in addition to **embedded MathML**.
- We have **quite some structure** in ConTEXt and math is no exception. When we add features we normally also take care of tagging.
- We already had a way to extract MathML from formulas, but with (presentation) MathML being **unstable** (dropping features, support comes and goes) we have to adapt and anticipate the worst.
- We can now actually make use of the already present **dictionary** mechanisms and carry a bit more information around with symbols. This saves some extra processing and serves serializing well.

- We could actually remove some rendering related output (alignments using tables) by **more natural** solutions.
- But . . . we need some information from **users**, like usage patterns, specific support for ‘fields’, and translations.
- We don't want to adapt the engine because it's very **macro package dependent** and it's also more flexible.

Tests

- A university **math book** of some 300 pages with 3500 formulas, and a lot of (educational) structure.
- The upcoming **math manual** with many examples, fancy features, specific control, symbols, different structures, etc.
- For performance tests we use relatively simple **text only** documents, like the King James bible, novels from the Gutenberg project, etc.
- For meaningful math we have a (growing) document that shows **examples in various languages** as well as MathML from ConTEXt input.

We can show some examples.

Impact: King James Bible

from xml, two columns, using unifraktur:

fitclasses	passes	tagging	pages	runtime	uncompressed	runtime
default		no	670	14.2		
default	quality	no	670	14.2		
granular	quality	no	672	14.3	24.999	14.5
granular	quality	yes	672	17.5	39.660	18.4

Impact: Math in ConTEXt

all bells and whistles, interactive, screen, menus, many math fonts:

tagging	pages	runtime	uncompressed	runtime	compressed
no	433	15.8	43.467	15.9	7.204
yes	433	18.5	52.842	18.7	8.648

Impact: Infinitesimalkalkyl

a lot of structure, granular, passes, interactive, thousands of formulas, graphics:

synctex	tagging	pages	runtime	uncompressed	runtime	compress
no	**no**	292			9.3	3.645
yes	**no**	292	9.7	17.379	9.8	3.645
yes	**yes**	292	15.3	27.652	15.8	5.815

April 25, 2024
Dell 7220 Laptop: Intel(R) Xeon(R) CPU E3-1505M v6 @ 3.00GHz, 48.0 GB, 2TB Samsung Pro SSD
Windows 10 Pro for Workstations
LuaMetaTeX 2.11.02 / 20240425 (MingW64)