

# Why our HTML Docs Don't Just Print and What to Do About It

---



Nikolaj Potashnikov

PhD in Economics, Solution architect, Course-IT

✉ consulting@yandex.ru ➬ @nmpotashnikoff



“

*The principal task of a conductor is not to put himself  
in evidence but to disappear behind his functions*

– Liszt Ferenc

Why our HTML Docs Don't  
Just Print and What to Do  
About It



Print, Export to Word, Export to PDF are very often just a trap

---

What to do with a long line in a listing?

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Print, Export to Word, Export to PDF are very often just a trap

---

## What to do with a long line in a listing?

- We may scale



# Print, Export to Word, Export to PDF are very often just a trap

---

## What to do with a long line in a listing?

- We may scale
- Or use landscape orientation



# Print, Export to Word, Export to PDF are very often just a trap

---

## What to do with a long line in a listing?

- We may scale
- Or use landscape orientation
- Or both, but would it be enough?



# Print, Export to Word, Export to PDF are very often just a trap

---

## What to do with a long line in a listing?

- We may scale
- Or use landscape orientation
- Or both, but would it be enough?
- If not, we may fire error for long lines or wrap them



# Print, Export to Word, Export to PDF are very often just a trap

---

## What to do with a long line in a listing?

- We may scale
- Or use landscape orientation
- Or both, but would it be enough?
- If not, we may fire error for long lines or wrap them
- With linefeed and spaces? And how to copy?



# Print, Export to Word, Export to PDF are very often just a trap

---

## What to do with a long line in a listing?

- We may scale
- Or use landscape orientation
- Or both, but would it be enough?
- If not, we may fire error for long lines or wrap them
- With linefeed and spaces? And how to copy?
- With indents? Still impossible to copy from PDF



# Print, Export to Word, Export to PDF are very often just a trap

---

## What to do with a long line in a listing?

- We may scale
- Or use landscape orientation
- Or both, but would it be enough?
- If not, we may fire error for long lines or wrap them
- With linefeed and spaces? And how to copy?
- With indents? Still impossible to copy from PDF
- And on the web we can just add horizontal scroll bar.



# Print, Export to Word, Export to PDF are very often just a trap

---

## What to do with a long line in a listing?

- We may scale
- Or use landscape orientation
- Or both, but would it be enough?
- If not, we may fire error for long lines or wrap them
- With linefeed and spaces? And how to copy?
- With indents? Still impossible to copy from PDF
- And on the web we can just add horizontal scroll bar.



There is a core mismatch: semantic markup meets the rigid world of print

---



## Iterations in converting simple text markup to print formats

---

- <https://github.com/CourseOrchestra/course-doc>: XSL-FO templates for Asciidoctor → DocBook backend
- <https://github.com/CourseOrchestra/asciidoc-open-document>: Open Document Converter for Asciidoc
- <https://github.com/fiddlededee/unidoc-publisher>: UniDoc Publisher – any markup to any printing rendering engine



# Main formats for printing

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Main formats for printing

---

## 1. PDF

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Main formats for printing

---

1. PDF
2. Text processing formats (Open XML – MS Office, Open Document – LibreOffice)

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Main formats for printing

---

1. PDF
2. Text processing formats (Open XML – MS Office, Open Document – LibreOffice)
3. HTML?

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Main formats for printing

---

1. PDF
  2. Text processing formats (Open XML – MS Office, Open Document – LibreOffice)
  3. HTML?
- 



CSS Paged Media – CSS extension, defining style specific for printing

---



# Most widespread rendering approaches

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Most widespread rendering approaches

---

- PDF ← native PDF-generating libraries

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Most widespread rendering approaches

---

- PDF ← native PDF-generating libraries
- PDF ← XSL-FO with FOP-processors

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Most widespread rendering approaches

---

- PDF ← native PDF-generating libraries
- PDF ← XSL-FO with FOP-processors
- PDF ← via TeX



# Most widespread rendering approaches

---

- PDF ← native PDF-generating libraries
- PDF ← XSL-FO with FOP-processors
- PDF ← via TeX
- PDF ← HTML + Paged Media CSS

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Most widespread rendering approaches

---

- PDF ← native PDF-generating libraries
- PDF ← XSL-FO with FOP-processors
- PDF ← via TeX
- PDF ← HTML + Paged Media CSS
- DOCX/ODT, PDF ← +/- text processors (MS Word, LO Writer)

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Most widespread rendering approaches

---

- PDF ← native PDF-generating libraries
  - PDF ← XSL-FO with FOP-processors
  - PDF ← via TeX
  - PDF ← HTML + Paged Media CSS
  - DOCX/ODT, PDF ← +/- text processors (MS Word, LO Writer)
- 

These technologies are not aligned in a great number of details like:



- Apache FOP has problems with Leader alignment (dots in a table of contents)
  - LO Writer doesn't support typography (like keep with next) within table cells
  - Microsoft doesn't recommend running automation tasks (like saving PDF) on a server
- 



# Some brief conclusions

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Some brief conclusions

---



Feel like speleologist?

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Some brief conclusions

---



Feel like speleologist?

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Some brief conclusions

---



Feel like speleologist?

---

- The world of printing is the world of constraints

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Some brief conclusions

---



Feel like speleologist?

---

- The world of printing is the world of constraints
- And those constraints differ for each technology, you often need to support several chains (exquisitely looking PDF with TeX and LibreOffice for coordination)



## Some brief conclusions

---



Feel like speleologist?

---

- The world of printing is the world of constraints
- And those constraints differ for each technology, you often need to support several chains (exquisitely looking PDF with TeX and LibreOffice for coordination)
- With no universal solutions



## UniDoc Publisher approach suits best if at least **one** of

---

- You don't prepare documentation especially for printing purposes
- You are automating documentation generation and hope it will look good, no matter what will be generated
- Your output format is one of the text processing format

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# In search for flexibility: Asciidoctor open document

---

**Automation on the writer side**

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# In search for flexibility: Asciidoctor open document

---

## Automation on the writer side

1. Asciidoctor parses markup into AST (Abstract Syntax Tree)

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# In search for flexibility: Asciidoctor open document

---

## Automation on the writer side

1. Asciidoctor parses markup into AST (Abstract Syntax Tree)
2. You may transform AST with Asciidoctor tree processor

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# In search for flexibility: Asciidoctor open document

---

## Automation on the writer side

1. Asciidoctor parses markup into AST (Abstract Syntax Tree)
2. You may transform AST with Asciidoctor tree processor
3. Asciidoctor runs writer template for each AST node recursively

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# In search for flexibility: Asciidoctor open document

---

## Automation on the writer side

1. Asciidoctor parses markup into AST (Abstract Syntax Tree)
2. You may transform AST with Asciidoctor tree processor
3. Asciidoctor runs writer template for each AST node recursively
4. You may override writer with pure Ruby or with special Slim templates



# A simplified processing AST example

---

```
- !<OrderedList>
  roles:
    - "arabic"
    id: "ol-1"
    captioned_title:
      children:
        - !<Text>
          text: "Automation"
    children:
- !<ListItem>
  children:
    - !<Paragraph>
      children:
        - !<Text>
          text: "Asciidoctor..."
- !<ListItem>
  ...
- list_style = "#{get_basic_style} ordered-list"
- if captioned_title?
  text:p text:style-name="#{list_style}"
    text:bookmark text:name="#{id}"
    =captioned_title
  text:list text:style-name="#{list_style}"
    - items.each_with_index do |item, index|
      ...
  ...
```

---



# Great, but

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Great, but

---

- You can't override part of a template

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Great, but

---

- You can't override part of a template
- You should invent something for styling



## Great, but

---

- You can't override part of a template
- You should invent something for styling

**Styling? But text processors do support styling!**

---



---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Great, but

---

- You can't override part of a template
- You should invent something for styling

**Styling? But text processors do support styling!**

---



- `<span class="bold green">bold, green</span>` – impossible to apply two styles to one element
- 



## Great, but

---

- You can't override part of a template
- You should invent something for styling

**Styling? But text processors do support styling!**

---



- `<span class="bold green">bold, green</span>` – impossible to apply two styles to one element
- 



- Asciidoc Open Document uses slightly extended intermediary OD format (to preserve AST attributes)
  - It uses special functions that check, if style should be applied. It doesn't know styling attributes but forces the Open Document style structure
- 



# And still

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## And still

---

- Unexpectedly transforming this extended Open Document format became one of the most used features of Asciidoctor Open Document

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## And still

---

- Unexpectedly transforming this extended Open Document format became one of the most used features of Asciidoctor Open Document
- Styling as separate task of writing proved also to be useful

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## And still

---

- Unexpectedly transforming this extended Open Document format became one of the most used features of Asciidoctor Open Document
- Styling as separate task of writing proved also to be useful
- Gradle was magnificent in gluing all parts together

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Thoughts before the second step

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Thoughts before the second step

---

- If creating universal converter is impossible...

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Thoughts before the second step

---

- If creating universal converter is impossible...
- We should create **meta converter** – platform for building converters

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Thoughts before the second step

---

- If creating universal converter is impossible...
- We should create **meta converter** – platform for building converters

## Estimated requirements

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Thoughts before the second step

---

- If creating universal converter is impossible...
- We should create **meta converter** – platform for building converters

## Estimated requirements

- Native converter as a reader



# Thoughts before the second step

---

- If creating universal converter is impossible...
- We should create **meta converter** – platform for building converters

## Estimated requirements

- Native converter as a reader
- Sound ways of transforming AST



# Thoughts before the second step

---

- If creating universal converter is impossible...
- We should create **meta converter** – platform for building converters

## Estimated requirements

- Native converter as a reader
- Sound ways of transforming AST
- A good approach for styling as a separate focus



# Thoughts before the second step

---

- If creating universal converter is impossible...
- We should create **meta converter** – platform for building converters

## Estimated requirements

- Native converter as a reader
- Sound ways of transforming AST
- A good approach for styling as a separate focus
- 99% generic writer



# Thoughts before the second step

---

- If creating universal converter is impossible...
- We should create **meta converter** – platform for building converters

## Estimated requirements

- Native converter as a reader
- Sound ways of transforming AST
- A good approach for styling as a separate focus
- 99% generic writer
- Good integration with CI/CD with a focus on homogeneity



# Native converter as a reader?

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Native converter as a reader?

---



Each converter outputs HTML. HTML is quite semantic, why shouldn't we use it?

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Let's convert this presentation to LO Writer

The slide has a purple border and a purple gear icon at the top center. The title is in bold black font. Below the title is a quote by Liszt Ferenc. On the left, there is a photo of the speaker and his contact information. A list of bullet points follows, along with a 'List of slides' section. The slide number is 6.

Why our HTML Docs Don't Just **Print** and What to Do About It

*The principal task of a conductor is not to put himself in evidence but to disappear behind his functions*

— Liszt Ferenc

Nikolaj Potashnikov  
PhD in Economics, Solution architect, Course-IT  
[consulting@yandex.ru](mailto:consulting@yandex.ru) @nmpotashnikoff

- As a solution architect, I want to share them? Via email?
- [REF] Why this epigraph?
  - It's FOSDEM, and lots of people
  - Value is in the result
  - A conductor should disappear behind us, no matter what format, rendering tec

List of slides

1. Print, Export to Word, Export
2. Iterations in converting simple
3. Main formats for printing....
- 4.1. Most widespread rendering
5. Some brief conclusions.....
6. UniDoc Publisher approach
7. In search for flexibility: Ascii
- 7.1. A simplified processing A
- 7.2. Great, but.....

6

Why our HTML Docs Don't Just Print and What to Do About It



# Notes on this demo

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Notes on this demo

---

- All conversion settings are written in Kotlin

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Notes on this demo

---

- All conversion settings are written in Kotlin
- Everything is in a single Gradle script (`build.gradle.kts`)

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Notes on this demo

---

- All conversion settings are written in Kotlin
- Everything is in a single Gradle script (`build.gradle.kts`)
- The following code listings are excerpts from this script



# Boilerplate

---

```
FodtConverter {
    html = AsciiDocHtmlFactory()
        .getHtmlFromFile(File("${project.projectDir}/$presentationFile.adoc"), true)
    template = File("${project.projectDir}/template-1.fodt").readText()
    adaptWith(AsciidocToOdtAdapter)
    unknownTagProcessingRule = unknownTagProcessingRuleRevealJs()
    parse() // Processing AST
    ast2fodt()
}
```

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Processing AST

---

```
ast().descendant { section ->
    section.sourceTagName == "section" &&
        section.descendant { it is Heading && it.level == 1 }
            .isNotEmpty()
}.first().also { it.insertBefore(makeTitle(it)) }.remove()
```

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Rearranging title (Asciidoc source)

---

```
| ====
a|
[.title-photo]
image::images/nmp1.jpg[]
a|
[.full-name]
Nikolaj Potashnikov

[.bio]
PhD in Economics, Solution architect, Course-IT
.2+>.>a|{nbsp}
[.logo]
image::images/fosdem-logo.svg[]
2+a|
[.contact]
icon:envelope[] consulting@yandex.ru icon:telegram[]{nbsp}@nmpotashnikoff
| ====

```

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Rearranging title, extracting semantics

---

```
val title = titleSlideSection.descendant { it is Heading && it.level == 1 }.first()
val notes = titleSlideSection.descendant { it.sourceTagName == "aside" }.first()
val (fullName, bio, photo, contact, logo) =
    arrayOf("full-name", "bio", "title-photo", "contact", "logo")
    .map { role -> titleSlideSection.descendant { it.roles.contains(role) }.first()

logo.descendant { it is Image }.first().let { it as Image }
    .width = Length(1000F, LengthUnit.cmm)
photo.descendant { it is Image }.first().let { it as Image }
    .width = Length(1500F, LengthUnit.cmm)
```

---



# Rearranging title, constructing title

---

```
appendChild(logo)
appendChild(title)
table {
  col(Length(18F)); col(Length(152F))
  roles("about-me")
  tableRowGroup(TRG.body) {
    tr {
      td { appendChild(photo) }
      td { arrayOf(fullName, bio, contact).forEach { appendChild(it) } }
    }
  }
}
appendChild(notes)
appendChild(Toc(2, "List of slides"))
normalizeImageDimensions()
```

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Back return to the result



## Why our HTML Docs Don't Just Print and What to Do About It

*The principal task of a conductor is not to put himself in evidence but to disappear behind his functions*

— Liszt Ferenc

 Nikolaj Potashnikov  
PhD in Economics, Solution architect, Course-IT  
[consulting@yandex.ru](mailto:consulting@yandex.ru) @nmpotashnikoff

- As a solution architect, I want to share them? Via email?
- [REF] Why this epigraph?
  - It's FOSDEM, and lots of people
  - Value is in the result
  - A conductor should guide us, not show us, no matter what format, rendering techniques

**List of slides**

- Print, Export to Word, Export
- Iterations in converting simple
- Main formats for printing...
- Most widespread rendering engines
- Some brief conclusions.....
- UniDoc Publisher approach
- In search for flexibility: AsciiDoc
1. A simplified processing API
- Great, but...

6



## Why our HTML Docs Don't Just Print and What to Do About It

Nikolaj Potashnikov  
PhD in Economics, Solution architect, Course-IT

[consulting@yandex.ru](mailto:consulting@yandex.ru) @nmpotashnikoff



Why our HTML Docs Don't Just Print and What to Do About It



# Styling

---

```
OdtStyle { p ->
    if (p !is Paragraph) return@OdtStyle
    if (p.ancestor { it.roles.contains("logo") }.isEmpty()) return@OdtStyle
        attributes("style:master-page-name" to "First_20_Page")
},
OdtStyle { tableCell ->
    if (tableCell !is TableCell) return@OdtStyle
    if (tableCell.ancestor { it.roles.contains("about-me") }.isEmpty()) return@OdtStyle
        tableCellProperties {
            arrayOf("top", "right", "bottom", "left")
                .forEach { attributes("fo:border-$it" to "none") }
        }
},
}
```

---



# Some more AST processing

---

```
ast().descendant { it.roles.contains("notes") }
    .forEach { it.insertBefore(HorizontalLine()) } ❶
ast().descendant { it is Heading && it.level > 1 }
    .forEach {
        it.insertBefore(
            Paragraph().apply { roles("slide-finish") }
        )
    } ❷
odtStyleList.add(odtStyles())
odtStyleList.add(rougeStyles()) ❸
```

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Extending AST

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Extending AST

---

```
class HorizontalLine() : NoWriterNode() {  
    override val isInline: Boolean get() = false  
}
```

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Extending AST

---

```
class HorizontalLine() : NoWriterNode() {  
    override val isInline: Boolean get() = false  
}
```

---

```
OdtCustomWriter { horizontalLine ->  
    if (horizontalLine !is HorizontalLine) return@OdtCustomWriter  
    preOdNode.apply {  
        "text:p" {  
            attributes("text:style-name" to "Horizontal Line")  
            process(horizontalLine)  
        }  
    }  
},
```

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Testing



Content type	The result
paragraph	Paragraph 1 Paragraph 2
list	1. Item 1 1. Subitem 1. Subitem 1. Subitem 2. Item 2
table	Subtable

Some paragraph after table.  
Some paragraph after table.

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Focus and trade-offs

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Focus and trade-offs

---

- CI friendly – pure Gradle (or an ordinary Kotlin project) to rule them all

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Focus and trade-offs

---

- CI friendly – pure Gradle (or an ordinary Kotlin project) to rule them all
- No declarations, everything should be programmed

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Focus and trade-offs

---

- CI friendly – pure Gradle (or an ordinary Kotlin project) to rule them all
- No declarations, everything should be programmed
- Typed AST – check before run

Why our HTML Docs Don't  
Just Print and What to Do  
About It



## Focus and trade-offs

---

- CI friendly – pure Gradle (or an ordinary Kotlin project) to rule them all
- No declarations, everything should be programmed
- Typed AST – check before run
- Clean and testable code



## Focus and trade-offs

---

- CI friendly – pure Gradle (or an ordinary Kotlin project) to rule them all
- No declarations, everything should be programmed
- Typed AST – check before run
- Clean and testable code
- One AST and one styling approach, but different styling API for each backend



# Conclusion

---

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Conclusion

---

- Treat printing as engineering: design it, test it, automate it

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Conclusion

---

- Treat printing as engineering: design it, test it, automate it
- Printing is a lossy transformation – some semantics cannot survive it

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Conclusion

---

- Treat printing as engineering: design it, test it, automate it
- Printing is a lossy transformation – some semantics cannot survive it
- Keep rendering logic programmable and under your control

Why our HTML Docs Don't  
Just Print and What to Do  
About It



# Questions?

---

- <https://github.com/fiddlededee/unidoc-publisher>: UniDoc Publisher – any markup to any printing rendering engine
- <https://github.com/fiddlededee/fosdem-printing/tree/main>: this presentation repository

Why our HTML Docs Don't  
Just Print and What to Do  
About It



