Valentin$E$-typo

from AsciiDoc to ConTeXt

Götz SCHNELL

2024-08-21

18th International ConTeXt Meeting

Contents

Goal of Valentin*E*-typo

$\implies$ lower the entry level into the world of ConT$_E$Xt.

Two use cases:

1. facilitate the creation of ConTeXt files (templates).
   - templates for different applications.
   - can be compiled without errors (working example).
   - pretty-printing of ConTeXt code is not a priority.
2. produce PDF output in typographic quality from AsciiDoc files.
   - AsciiDoc is a relatively complete and versatile lightweight markup language.
   - alternative approaches show difficulties.

Why AsciiDoc?

"Write as code"

extension of "Documentation as Code"

Pros

⟶ syntax of DocBook, suitable for writing technical books,

⟶ quite many syntax elements, for documentation see https://docs.asciidoctor.org/asciidoc/latest/ ,

⟶ standardized: https://asciidoc-wg.eclipse.org/ ,

⟶ human readable, similarities with Markdown,

⟶ can be converted in other formats without great effort.

Other approaches

- AsciiDoc converter for PDF exists: *Asciidoctor-pdf*:
  - limited typographic quality,
  - limited theming possibilities,
  - fast, good solution for quick reports in PDF.
- AsciiDoc converter for LaT$_E$X exists: *Asciidoctor-Latex*:
  - difficulties in installation,
  - limits for different use cases?
- Pandoc can convert AsciiDoc files into ConT$_E$Xt
  - limits for special text elements
  - different with configuration variants

$\longrightarrow$ So, why not trying with ConT$_E$Xt?

Concept of Valentin*E*-typo

- AsciiDoc file can be converted with different backends:
  - HTML5, DocBook5, Epub, asciidoctor-pdf , . . .
- Monolithic library:
  - one style for distinct document type (exception: Asciidoc doctypes),
  - no distraction by "fiddling" with parameters,
  - variants with AsciiDoc document attributes and options.
- transparency, repeatbility (PDF metadata).
- Library styles with different document types.

⟶ see online repository:

https://gitlab.com/valentine4743416/ValentinE-typo

Steps to write a document:

Authoring

1. Write a file in format AsciiDoc
   - with any text editor
   - on any device

Converting

2. Convert AsciiDoc file into a ConTeXt file
   - with a specific typo-theme library file
   - maybe on another device

Typesetting

3. Compile the ConTeXt file into a PDF file
   - maybe on another device

Challenges in development

- Big variety in AsciiDoc syntax, for example:
  - styles for bullets,
  - design of tables.
- "Special characters" in AsciiDoc and ConTeXt:
  - ({, [, &).
- Decisions on styling possibilities in AsciiDoc or in ConTeXt, for example:
  - with document attributes like `sectnums`.
  - **decision**: priority for AsciiDoc attributes

Additional ConTeXt content

1. Pass macro
   - depends on backend, deprecated
2. Roles
   - maybe intrepretated by every backend

Example 1. Use of a ConTeXt command as role

```
The velocity is [.unit]#800 kilometer per hour#.
```

is interpretated as:

The velocity is 800 km/h.

3. `docinfo` file
   - include ConT$_E$Xt content into the header of the ConT$_E$Xt files
   - you may define new roles

Example 2. Use of a command defined in the docinfo file

```
[.textorange]
This paragraph has a nice color.
```
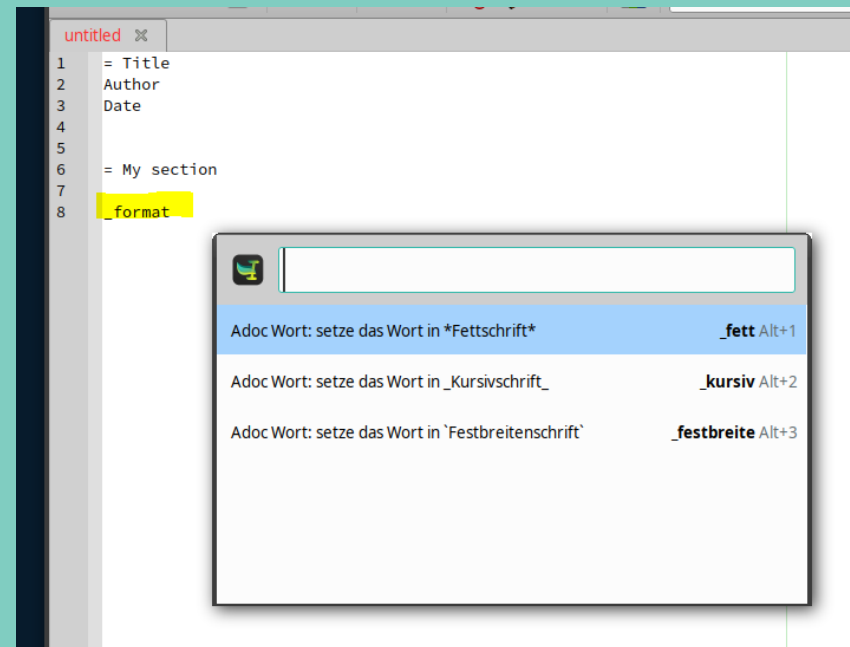
is rendered as

This paragraph has a nice color.

Helpers

Editing (Ascii,Doc syntax)

- AsciiDoc Live: https://asciidoclive.com/ ,
- AsciiDoc FX: https://asciidocfx.com/ ,
- Syntax highlighting and helpers in text editors
- Browser extensions (Chrome, Firefox, Safari, Opera, Edge).

- Espanso (text expander)
  - like "snippets" in text editors
  - Selection box

Rendering (AsciiDoc syntax and final result)

- Reference manual
  - for each library files
  - in mother language
  - AsciiDoc Syntax elements and their interpretation by the specific library file