OST
Ostschweizer
Fachhochschule

Markua Support for Pandoc

Department of Computer Science
OST – University of Applied Sciences
Campus Rapperswil-Jona

Autumn Term 2021

Author(s): Samuel Lemmenmeier, Tim Wisotzki
Advisor: Prof. Dr. Farhad D. Mehta

Contents

Project Documentation

Abstract

The purpose of this semester thesis was to add Markua version 0.10 support to Pandoc.

Pandoc is an open-source project that can read and write different markup languages and convert between them. It supports many different formats but the possiblity to convert to Markua was not a feasable option. Markua is a markup language, developed by Peter Armstrong, that is mainly used on Leanpub for writing books. So this new functionality was requested from the community and especially from authors and publishers who are writting in Markua and would like to publish their work on Leanpub.

We accomplished the task by writing a Markua writer for Pandoc, which we then integrated within the existing Markdown writer as a new variant.

The main project phase ended with the successful integration of our Markua 0.10 writer to the official Pandoc code base.

By integrating the Markua writer into the existing Markdown writer two advantages arose. Firstly it decreases duplicated code because the similarities between Markua and Markdown. Secondly it is much more likely that the Markua variant will be maintained in the future since the Markdown writer is one of the main supported and used writers in Pandoc.

Introduction

The following text documents the SA “Markua Support for Pandoc”.

It contains everything regarding the project and is suplemented by the product documentation. First it starts with the projectplan to set the stage and give an overview timewise. Then possible risks are assessed and requirements are recorded. After that a more personal review about our project approach will follow and after that some technical things are listed as well as all references and links. Finally in the appendix additional files and artifacts can be found.

Projektplan

Zweck des Projektplans

Der Projektplan dient der Strukturierung des Projektes “Markua Support for Pandoc” und liefert eine Uebersicht ueber die noetigen Arbeiten, die zur Erfuellung der Projektziele, ausgefuehrt werden.

Gueltigkeitsbereich

Der Projektplan bezieht sich auf die SA “Markua Support for Pandoc” von Samuel Lemmenmeier und Tim Wisotzki im HS21 an der FH OST Standort Rapperswil-Jona und ist waehrend der Laufzeit dieses Projektes gueltig.

Der Projektplan kann waehrend des Projektes angepasst werden, um den aktuellen Stand der Arbeiten zu reflektieren.

Projekt Uebersicht

Das Hauptziel der SA ist es das Format Markua in dem Open-Source Projekt Pandoc zu realisieren und zu implementieren.

Zweck und Ziele des Projektes Zweck des Projektes ist es nebst neuen Erfahrungen in einem Software-Projekt und mit FP in Haskell zu sammeln, auch einen Beitrag an die Gemeinschaft zu leisten. Letzter Punkt wird erzielt indem das Open-Source Projekt Pandoc erweitert wird.

- MVP ist ein Markua Writer fuer die Version 0.10

Definitionen und Begriffe Fuer das Projekt werden Abkuerzungen benutzt, diese sind im Glossar zu finden. Gewisse Begriffe werden in der ursprungssprache Englisch belassen, um Verwirrungen zu vermeiden.

Lieferumfang

- Quellcode welcher die Implementation von Markua in Pandoc bereitstellt. Zudem wird dieser Quellcode auch noch an das Open-Source Github Repository geschickt.
- Produktdokumentation in Englisch, diese soll dem Verstaendnis des Produktes dieser SA dienen sowie eine spaetere Weiterentwicklung dieses Produktes unterstuetzen.
- Projekt Dokumentation in Deutsch oder Englisch. Diese wird verschiedene Dokumente beinhalten, z.B. diesen Projektplan.
- Weitere Artefakten die im Zusammenhang mit diesem Projekt entstehen.

Projektorganisation

Das Projekt wird gemaess dem RUP in die vier Phasen **Inception**, **Elaboration**, **Construction** und **Transition** unterteilt. Gearbeitet wird schliesslich agil anhand von Scrum. Die Sprints sollen, wenn moeglich und konform mit den vier oben genannten Phasen, je zwei Wochen dauern.

Im ganzen Projekt wird iterativ in agiler Arbeitsweise vorgegangen, d.h. grundsaetzlich sind alle Dokumente Momentaufnahmen und koennen bei Bedarf angepasst werden. Diese Anpassungen koennen auch auf definierte Tools, Arbeitsprodukte, Zwischenziele sowie auf den definitiven Umfang des Endproduktes Auswirkungen haben.

Organisationsstruktur Das Team besteht aus folgenden Mitgliedern:

- Tim Wisotzki tim.wisotzki@ost.ch
- Samuel Lemmenmeier samuel.lemmenmeier@ost.ch

Externe Schnittstellen

- Projektbetreuung: Prof. Dr. Farhad D. Mehta

Management Ablaeufe

Kostenvoranschlag Das Projekt beginnt am 20.9.2021 (Kalenderwoche 38) und endet am 24.12.2021 (Kalenderwoche 51). Insgesamt stehen laut ECTS ca. 480 Arbeitsstunden zur Verfuegung.

Zeitliche Planung Da es in unserem Team noch keine grosse Projekt Erfahrung, insbesondere mit Haskell gibt, ist es zum momentanen Zeitpunkt schwierig abzuschaetzen, welche Aufgaben wie viel Zeit beanspruchen. Insbesondere das Einarbeiten in die FP sowie in Pandoc koennen zeitintensiv ausfallen. Daher werden wir agil Planen und zuerst den Fokus auf einen Writer legen.

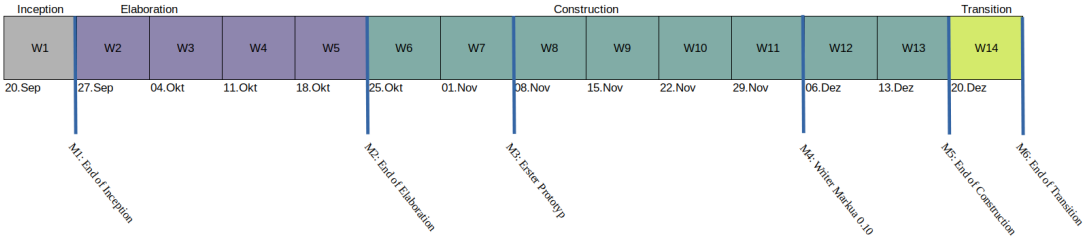

Figure 1: Grafik Projektuebersicht

Phasen / Iterationen

Inception Diese Phase laeuft vom 20.9.2021 bis zum 26.9.2021.

In dieser machen wir eine grobe zeitliche Planung fuer das Projekt. Zudem werden erste Produkte (Dokumentation und Code) sowie Ziele festgehalten. Tools werden vorlaeufig definiert und eingerichtet. Ebenfalls findet ein erster Kontakt mit dem Github Repository von Pandoc statt. Zum Schluss wird noch eine erste Risikoanalyse durchgefuehrt.

Elaboration Diese Phase laeuft vom 27.9.2021 bis zum 24.10.2021.

In dieser Phase wird die grobe zeitliche Planung verfeinert und weitere Arbeitspakete definiert. Ziel dieser Phase ist es einen zeitlichen Ueberblick zu erhalten, bis wann die wichtigsten Arbeitsprodukte in welchem Umfang fertiggestellt werden sollten. Weiteres Hauptziel ist es, sich mit Haskell vertrauter zu machen und sich gruendlich in Pandoc einzuarbeiten. In anderen Worten bedeutet dies, dass nach dieser Phase das Wissen ueber Haskell, sowie das Verstaendnis ueber Aufbau und Funktionsweise von Pandoc auf einem Niveau sein sollte, dass der Einstieg in die Construction Phase so reibungslos wie moeglich verlaeuft.

Construction Diese Phase laeuft vom 25.10.2021 bis zum 19.12.2021.

Waehrend dieser Phase wird iterativ an der Fertigstellung des Projektes gearbeitet. Schwerpunkt und damit erstes grosses Zwischenziel ist ein Markua Writer.

Transition Diese Phase laeuft vom 20.12.2021 bis zum 24.12.2021 17:00 Uhr

In der Transition werden die letzten Tests durchgefuehrt und das Projekt fuer die Abgabe vorbereitet. Insbesondere wird diese Zeit noch fuer das Zusammenstellen der Projekt -und Produktdokumentation genutzt.

Iterationen Wir verwenden 2-Wochen Iterationen, sofern diese konform zu den vier Phasen sind (d.h. ausser fuer die erste und letzte Iteration). Diese werden laufend geplant und wenn noetig auch anpegasst.

Meilensteine Die Meilensteine koennen in Zukunft noch angepasst werden. Die Meilensteine dienen in erster Linie als Zwischenziele, so kann auch der Fortschritt des Projektes ueberprueft werden.

M1 End of Inception Datum: Ende SW 1 -> 27.9.21, 24:59

Fuer diesen Meilenstein erstellen wir einen ersten Projektplan, der die grobe Struktur erfasst. Erste Risiken werden in der Risikoanalyse festgehalten. Ziel ist es ebenfalls, die ersten Tools bestimmt und eingerichtet zu haben.

Arbeitsprodukte M1:

Folgende Arbeitsprodukte werden zum Meilenstein 1 erstellt:

- Projektplan
- Risiken

M2 End of Elaboration Datum: Ende SW 5 -> 24.10.21, 24:59

Zu diesem Meilaenstein hin erarbeiten wir sowohl funktionale, als auch nicht funktionale Anforderungen. In anderen Worten bedeutet dies, wir untersuchen die genauen Anforderungen die an einen Writer im Projekt Pandoc erwartet werden koennen. Ebenfalls wird der Zeitplan und somit auch der Projektplan verfeinert und spezifischer gestaltet. Schliesslich sollten die Tools, die wir verwenden werden, fertig eingerichtet sein. Ein weiterer Schwerpunkt wird auf Aufarbeitung von Haskell-Defiziten liegen, sowie das Verstaendnis und Wissen ueber Pandoc und dessen Funktionsweise zu erlangen.

Arbeitsprodukte M2

Folgende Arbeitsprodukte werden zum Meilenstein 2 erstellt:

- Aktualisierter Projektplan
- Anforderungsspezifikationen
- Aktualisierte Risikoanalyse
- Skizze und Erklaerung die zum Verstaendnis des Ablaufes und Funktionsweise von Pandoc beitragen, siehe Pandoc translation process.

M3 Prototyp Datum: Ende SW 7 - 7.11.21, 23:59

Ziel ist es, eine erste, bis zu einem gewissen Grad funktionierende, Version eines Markua Writers zu haben. Der Grad muss aber noch definiert werden. Dieses Zwischenprodukt sollte helfen Risiken zu minimieren.

Arbeitsprodukte M3

Folgende Arbeitsprodukte werden zum Meilenstein 3 erstellt:

- Aktualisierter Projektplan
- Anforderungsspezifikationen
- Prototyp der MVP beinhaltet

M4 Writer Markua 0.10 Datum: Ende SW 11 - 05.12.21, 23:59

Bei diesem Meilenstein soll der Prototyp die Anforderungen des Markua Writers ohne grosse Fehler erfuellen koennen. Das heisst der Pandoc Writer fuer Markua 0.10 ist implementiert. Ein weiteres Ziel dieses Meilensteines ist es, bereits einen ersten Pull Request an Pandoc gemacht zu haben.

Arbeitsprodukte M4

Folgende Arbeitsprodukte werden zum Meilenstein 4 erstellt:

- Writer Markua 0.10
- Pull Request
- Zugehoerige Artefakten fuer Dokumentation erweitert

M5 End of Construction Datum: Ende SW 13 - 19.12.21, 23:59

Hauptziel dieser Phase ist Funktionalalitaet eines Writer fuer Markua 0.10 zu haben. Es werden Tests, die die Funktionsweise des Writers sicherstellen sollen, implementiert. Ziel dieser Phase ist es moegliche Verbesserungsvorschlaege von Pandoc bereits angepasst zu haben, so dass unser Code gemerged wird.

Arbeitsprodukte M5

Folgende Arbeitsprodukte werden zum Meilenstein 5 erstellt:

- Funktionierender Writer fuer Markua 0.10
- Weitere Artifakten die zum Verstaendnis fuer Code und Produkt beitragen
- Angepasster Projektplan, Risikoanalyse etc.

M6 End of Transition Datum: Ende SW 14 - 24.12.21, 17:00 Die Projekt -und Produktdokumentation wird fertig gestellt. Fuer die Abgabe werden alle entsprechenden Files sowie der Code zusammengestellt.

Arbeitsprodukte M6

Folgende Arbeitsprodukte werden zum Meilenstein 6 erstellt:

- Quellcode
- Projekt -und Produktdokumentation
- Weitere Artifakten die die Projekt -und Produktdokumentation ergaenzen

Besprechungen Besprechungen finden in regelmaessigen Abstaenden statt. Der naechste Termin wird aber jeweils erst bei der letzten Besprechung definiert. Bei Bedarf koennen auch mehr oder weniger Besprechungen geplant werden.

Infrastruktur

- MS Teams fuer Meeting
- Github fuer Code, Documentation
- Jira fuer Project Management und Zeiterfassung
- Eigene Editorwahl zum Programmieren

Risikomanagement

Das Projet enthaelt verschiedene Risiken. Diese veraendern sich im Verlauf des Projektes. Eine Uebersicht der Risiken kann aus der Tabelle im Appendix und Risiken gefunden werden. Die bestehenden Risiken werden bei der Sprintplanung beruecksichtigt und entsprechende Puffer eingebaut. Im Verlauf des Projektes sollte das Gesamtrisiko abnehmen, da durch die geleistete Arbeit bereits Teile bestehen und die Unsicherheit reduziert wird.

Anforderungen

Die Anforderungsspezifikationen werden in Anforderungsspezifikationen beschrieben.

Qualitaetsmassnahmen

Um die Qualitaet des Projektes sicherzustellen, sehen wir folgende Massnahmen vor:

Projekt Das Projekt wird agil entwickelt. Durch ein regelmaessiges Abgleichen mit Projektplan und Timeline, sowie mehrere woechentliche Meetings koennen Abweichungen vom Plan fruehzeitig bemerkt werden.

Code Um die Code Qualitaet zu gewaehrleisten wird der code moeglichst im Style von Pandoc gehalten. Um dies automatisch zu verifiziren, verwenden wir einen Linter, welcher ebenfalls in Pandoc genutzt wird. Zudem wird regelmaessig Output mit dem Markua Writer generiert und dieser wird wiederum manuell auf Markua-Konformitaet ueberprueft. Besonders bei complexeren Funktionen wird zudem das vier Augenprinzip verwendet um allfaellige Mangel schnell zu bemerken.

Weiter werden bei Bedard und wichtigem Code explizite Code-reviews durchgefuehrt.

Technologien

Projektmanagement Wir benutzen Jira als Projektmanagement Tool, Issue Tracker und fuer die Zeiterfassung.
Neue Work-Items werden ueber Issues erstellt und in Stories zusammengefasst, die wiederum in Sprints abgearbeitet werden.

Die Definition of Done entspricht dem Zustand: Funktioniert und wurde durch alle SA-Teammitglieder fuer in Ordnung befunden.

Die Dokumentation wird in Markdown geschrieben und via Github gemanaged.

Entwicklung Entsprechend zu Pandoc werden wir Haskell als Progammiersprache verwenden. Ebenfalls verwenden wir mit hlint den gleichen Linter.

Der Code wird ebenfalls mittels Github verwaltet.

Our approach

The following chapter describes our process in an informal and retrospective fashion.

The goal of our SA was to add the support of Markua to Pandoc. To use all features of Pandoc, a reader and writer would be required. Since we did not have any experience with either Pandoc or Markua and only a limited knowledge of the Haskell language, our main target was to build a writer for the current Markua version of 0.10.

Preparation & mapping

Before any work on the actual project could be done we had to familiarise ourself with the involved languages and tools. We started analysing the Markua manual, which is the primary description of Markua to get an overview of the language. Simultaneously we looked into Pandoc, mainly the internal representation of the conversion-data. While getting to know the two formats, we continuously worked on a mapping from Pandoc's AST to Markua. At the same time we identified parts, that did not have a clear counterpart or lacked proper specification, in order to further research them.

Prototype

To implement a writer that can be integrated into the Pandoc project, it is necessary to understand its architecture and mechanisms. Because Pandoc is a rather large project with ten thousands of lines of code we early on started to read it's documentation and look through the different parts of the source code. After gaining a rough overview of the relevant parts we started to create a prototype writer with very limited functionality. The main goal of this writer was to integrate our writer within the existing project and to insure proper interaction with all necessary internal mechanisms.

Initial writer

While approaching the implementation we oriented us on the architecture of existing writers. Besides acting as an example this insured, that our structure would fit into the established style of the Pandoc project. We started implementing the translation with a divide and conquer approach, starting with simpler text based types and then preceded with the more advanced structures. The internal datatype, the Pandoc AST is a nested tree of different blocks, subtypes and metadata. This proved to be more complex than we initialy thought, which forced us to do some siginificant restructuring of the code. The change mainly affected the interaction of nested blocks and their attributes. Especially blocks that require a differnt output depending on their parent and the level of recursion require intricate interaction between all elements. In order to get the writer "pull request ready" we had to additionaly verify and complement the internal documentation as well as create and run tests. Lastly it had to be insured, that all administrative requirements for the merge were met, before the pull request was created.

Our individual writer at this stage can be found in the repository under initial_writer.

Pull request & improvements

After opening the pull request we got our first feedback through the review of the Pandoc maintainers. The main suggestion was to integrate the Markua writer as a variant of Markdown to improve the maintainability and reduce code. We subsequently estimated the feasibility, since a major change would be required, but concluded, that the benefits prevail. The following integration with the existing Markdown writer demanded an intense effort in order to meet our time constraints. Given that we already had a working solution, we were able to adopt many functions and thereby saving a lot of time. The second major benefit was our by now greatly improved knowledge of Pandoc, Markua and Haskell.

After our updated pull request we got detailed feedback from Pandocs maintainer John MacFarlane. We discussed and improved the code accordingly until everyone was satisfied and the pull request got accepted. The final code can now be found in the official Pandoc repository.

Finishing the project

After our main objective was reached with the integration of Markua support, we were able to prepare the final presentation, show casing our work. At the same time we worked on finishing the documentation and administrative parts of the project.

Technical description

Intuitive presentation of how Pandoc is working

Pandoc is build quite modular. This means a reader takes over the task to parse an input file into the internal representation, called AST. And a writer handles the translation from the AST to the desired output format. Because of this architecture Pandoc is able to support many different input and output formats and it is also possible to add new functionality to Pandoc by just adding a single reader or a single writer. The following graphic visualizes the process input -> AST -> output.

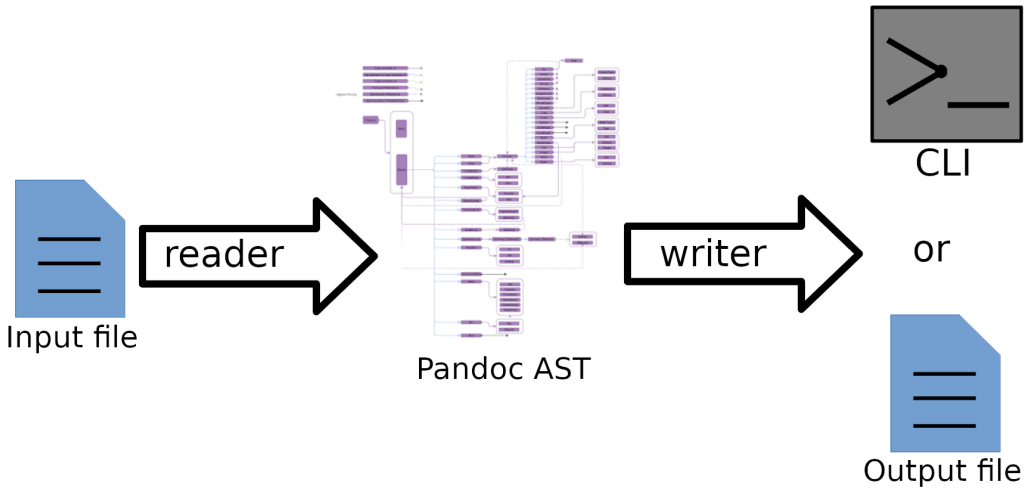

Figure 2: Pandoc translation process

Product description

A full description of the created writer can be found in the product-descriptoin

Conclusion

We started of the project with little knowledge about many of the involved parts. The seemingly simple task description turned out to be much more complex then anticipated and we had to navigate through many difficulties. Finally though we managed so solve all of them and were able to extend a very useful tool with a much requested feature and thereby creating something truely applicable.

Along the way we learned a lot about Pandoc, Haskell and a variety of markup languages. We gathered a lot of experiance not only working on an open-source project, but one of the biggest Haskell ones. Given our limited familiarity in the beginning, we can certainly count the project as a success.

References and links

- Code Repository, Markua Writer 0.10 as a standalone
- Code Repository, Markua Writer 0.10 integrated as a Markdown variant

Appendix

Glossary

abreviation	meaning
AST	Abstract Syntax Tree
FH	Fachhochschule
FP	functional programming
GFM	GitHub Flavored Markdown
HS21	autumn semester 2021
MS	Microsoft
MVP	Minimal Viable Product
Nr	Nummer
R#	risk number
RUP	Rational Unified Process
SA	semester-thesis
SW	semester-week

Risks

The following list contains possible project risks and was updated continuously during the project:

Nr	Titel	Beschrieb	max. Schaden	W.keit	Risiko
R1	Zeit Management	Zeitaufwand unterschaetzt	200 h	10-20 %	5
R2	Wieder kehrende Bugs	Wiederauftauchen von bereits korrigierten Fehlern	20 h	10 %	2
R3	Tool Ausfall	Ausfall von Github, Jira etc	30 h	0-5 %	4
R4	Knowhow Defizit	Teammitglied hat Knowhow Defizite	50 h	20-50 %	3
R5	Duplikat	Markua writer wurde in Zwischenzeit bereits erstellt	480 h	1-5 %	3
R6	Performance	Code braucht zu lange um Format zu konvertieren	50 h	20 %	5
R7	nicht Zufriedenstel-lend	Code funkioniert, aber Konvertierung genuegt den Anspruechen nicht	100 h	50 %	50
R8	Umfang zu gross	Erwarteter Umfang uebersteigt einsetzbare Arbeitsstunden	50 h	-	5
R9	falsch Uebersetz	Falsche Annahmen bez. der Herangehensweise sowie der Form der Umsetzung	200 h	20-100 %	5

Nr	Vorbeugung	Verhalten beim Eintreten
R1	Milestones und Fahrplan definieren sowie regelmaessig ueberpruefen ob auf Kurs	agil arbeiten, Zeitschaetzungen anpassen
R2	Testcases	Codereview
R3	keine Vorbeugung moeglich	offline weiter arbeitent
R4	genuegend Zeit einplanen um Wissensluecken auf zu arbeiten	Rat bei Experten einholen
R6	fragen	am Projektanfang: neues Thema, am Projektende trotzdem weiter arbeiten
R7	regelmaessige Tests, Performance ueberpruefen, Codereview	optimieren
R7	regelmaessig Testfile durchlaufen lassen, Anforderungsspezifikationen definieren	Codereviews, Anforderungsspezifikationen anpassen
R8	regelmaessig Testfile durchlaufen lassen, Anforderungsspezifikationen definieren	Codereviews, Anforderungsspezifikationen anpassen
R9	moeglichst frueh einen Pull Request machen	Projekt verstehen

General risks like hardware failure, unavailability of team-members, etc. are further risks, but are not explicitly noted here.

Requirements

General Description

Product Perspective The goal of the SA is to bring Markua support to the open-source project Pandoc. Markua is the main markdown language used by Leanpub.

Product Function In general the main goal of the SA is to add a writer for Markua version 0.10.

User Characteristics Probably the main group of people who will use the Markua support in Pandoc are authors and writers publishing on Leanpub or just who recognizes the benefits of Markua.

Use Cases An author want to convert a document to Markua.

Actors & Stakeholder Mainactors

- Authors and writers associated with Leanpub.
- Any other person who has an interest to convert some document into Markua format.

Stakeholders

- all actors
- SA-Team
- Project monitoring: Prof. Dr. Farhad D. Mehta
- Project supporter: Prof. Mirko Stocker

Functional requirements

- Conversion: The conversion to Markua and from Markua for the corresponding versions works as a user can expect, this means the conversion is in alignment with the Markua specification of the corresponding version.
- Usability: The usability of the Markua support is provided in the same way like the other writers and readers already implemented in Pandoc, this means Console based.
- Integration: The integreation with the existing Pandoc project should be seamless.

Non functional requirements

- Performance: The conversion is done in some reasonable time, compared to the other writers and readers already implemented in Pandoc.
- Maintainability: The Markua writer should be easy to maintain the future.
- Extendable: It should be simple to add additional functionality or future upgrades.

Timesheet

Project	Samuel Lemmenmeier	Tim Niklas Wisotzki	Total seconds	Total
pandocSA	303h 35m	306h 40m	2196900	610h 15m

Key	Type	Summary	Total time
SA-45	Story	Documentation	31h 15m
SA-44	Story	ASP Markua Format	4h
SA-42	Story	adjust plan to also cover Reader and Pull Request	30m
SA-41	Story	change writer to Data.Text	5h
SA-40	Story	Mini simple prototype writer	8h
SA-39	Story	Mapping of Features from Pandoc ASP to Markua	8h
SA-38	Epic	project administration	8h 30m
SA-36	Story	timetracking einladen	30m
SA-34	Story	verschiedene features durchgehen und uebersetzen	11h
SA-29	Story	Markua features auflisten	6h 15m
SA-28	Story	presentation	2h
SA-27	Story	pandoc beispiele verstehen (flags etc. beachten)	10h
SA-25	Epic	meetings	82h 35m
SA-23	Story	Haskell einarbeiten	38h 30m
SA-22	Story	pandoc, welche files sind relevant um den Writer zu integrieren	8h
SA-21	Story	pandoc internes format bestimmen	16h
SA-19	Story	mail to Mirco Stocker	40m
SA-18	Story	pandoc v0.1 Writer (MVP)	338h 15m
SA-16	Story	setup project	9h
SA-15	Story	pandoc familiarize with structure & code	14h 30m
SA-11	Story	assess risks	1h 30m
SA-10	Story	create projectplan	6h 15m
			610h 15m

Visualisation of Pandoc AST

Following graph shows an overview of Pandoc Block and Inline types:

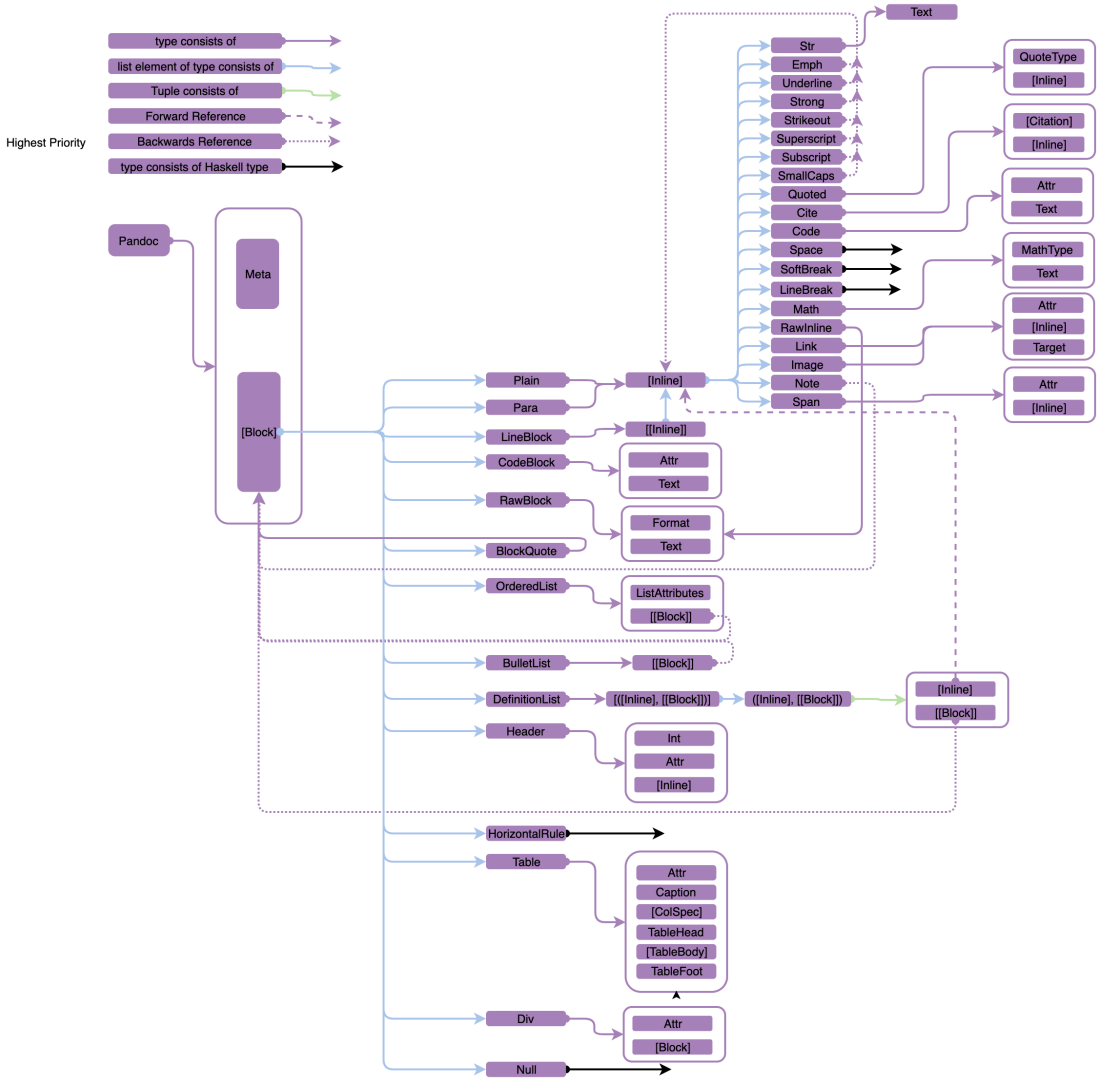

Figure 3: Nested Block and Inlines of AST

Contents

Product Documentation

Overview

A specific glossary for this document can be found at the end of this document.

Purpose of this document

The following text documents the integration of Markua 0.10 to Pandoc.

This documentation provides an overview of the final product, which consists of all newly integrated parts to the official Pandoc repository, that are related to Markua.

A general overview of the entire Pandoc project and its different functions can be found in the Pandoc User Guide.

Markua

Markua is a markup language developed by Peter Armstrong. It is based on Markdown and therefore has many similarities to it, mainly to GFM. The main focus of Markua is to optimise the writing of books and therefore targets a different audience. Accordingly there are some significant differences which will be discussed later on.

Markua is mainly used on Leanpub, where it is getting developed as an alternative and replacement for LFM. Currently version 0.10 is used, while version 0.30 is still under development.

More information can be found in the Markua Manual.

Usage

The conversion to Markua 0.10 can be done in the same way as any other transformation with Pandoc, using the following command:

```
pandoc <INPUT-FILE> -r <INPUT-FORMAT> -w markua -o <OUTPUT-FILE>
```

Options

The following options can be used when converting to Markua:

- all general pandoc options
- `--citeproc` renders citations in the file, generally used together with bibliography
- `--bibliography` is used to specify the bibliography-file
- `--columns` defines the maximum length of all lines in the output file
- `--wrap` controls the behaviour of linebreaks, when the maximum line-length is exeeded.

Special semantics

In order to create blurbs and asides in Markua the division block in Pandoc is used in combination with its class attribute. To generate a blurb in Markua the input needs to contain a division with the class "blurb". The division can still contain additional classes at the same time. The same syntax works for asides as well.

Technical description

Integration into Pandoc

Markua is implemented as a variant of Markdown in the Markdown writer and its subfiles. Whenever possible the existing functionality is used to improve maintainability and avoid duplications. For this purpose some extentions are automatically enabled in order to achieve the desired output. The user itself is not able to change or set any extensions, since the behaviour of Markua is fully defined. However the user is able to set some options, that will influence the resulting output.

Mapping AST to Markua

The following list shows all major elements involved in the mapping. The entire AST can be seen in the Pandoc Definition.

AST Internal Type	Markua Version 0.10 representation
Pandoc Meta [Block]	Meta is not mapped directly but used to influence the output
Plain [Inline]	Plain text
Para [Inline]	Paragraph, like plain text but separated with newlines
LineBlock [[Inline]]	Multiple non-breaking lines
CodeBlock Attr Text	Inline as a figure with list of attributes on preceding line
RawBlock Format Text	No output
BlockQuote [Block]	Block quote
OrderedList ListAttributes [[Block]]	Numbered list
BulletList [[Block]]	Bullet list
DefinitionList [([Inline], [[Block]])]	Definition list
Header Int Attr [Inline]	Atx header with list of attributes on preceding line
HorizontalRule	Scene Break
Table	GFM Table syntax used by Markua
Div Attr [Block]	If Attr contains class aside: Aside
Div Attr [Block]	If Attr contains class blurb: Blurb
Div Attr [Block]	If Attr contains class refs and options citeproc & bibliography are set: Bibliography content

AST Internal Type	Markua Version 0.10 representation
Div Attr [Block]	Else: just the content without any div tag
Null	Nothing
Str Text	Text
Emph [Inline]	Italic text
Underline [Inline]	Underline text represented as italic text
Strong [Inline]	Bold text
Strikeout [Inline]	Strikethrough text
Superscript [Inline]	Superscripted text
Subscript [Inline]	Subscripted text
SmallCaps [Inline]	Small capitalized text
Quoted SingleQuote [Inline]	Double quoted text
Quoted DoubleQuote [Inline]	Single quoted text
Cite [Citation]	Outputs the exact input content unless a bibliography is generated
Code Attr Text	Inline code span succeeded directly by a possible attribute list
Space	Prints a Space
Softbreak	Prints a Space
Linebreak	Prints a Newline
Math DisplayMath Text	LaTeX math inserted as a figure with list of attributes on preceding line
Math InlineMath Text	LaTeX math inserted as a span
Rawline Format Text	No output
Link Attr [Inline] Target	Hyperlink as an inline link with succeeding list of attributes
Image Attr [Inline] Target	Inserted as a figure with list of attributes on preceding line
Note [Block]	Footnote
Span Attr [Inline]	Inserted as a span with succeeding list of attributes
DefaultStyle	Decimal list number
Example	Decimal list number
Decimal	Decimal list number
LowerRoman	Lower roman list number
UpperRoman	Upper roman list number
LowerAlpha	Lower alphanumeric list number
UpperAlpha	Upper alphanumeric list number
DefaultDelim	. as list delimiter
Period	. as list delimiter
OneParen	) as list delimiter
TwoParens	) as list delimiter
type Target	Possible url and title gets extracted and will be further processed
type Attr	A Markua attribute list can be created

Attributes

A list of attributes pre- or succeeding a Markua construct will only be added if the list is not empty.

Attr can contain an id, classes and key-value pairs. With these data a specific Markua attribute list can be created. The list has following structure: {id: someId, class: firstClass, class: secondClass, key1: value1, key2: value2} If an id is available, it gets integrated within the attribute list with the identifier id. For example this is always the case for headers. Some specific classes will be transformed into a key-value pair e.g. {format: "latex"} but normally classes will be denoted with the default identifier class. Whenever an attribute has a specific name or is a key-value pair it will be added to the attribue list as a key-value pair. The values of the attributes with the key alt, caption or title are double quoted.

Tests

Besides the normal Pandoc testsuite, there are some specific tests for Markua. They can be found within the writer specific tests.

Special considerations

Some syntax is quite Markua specific, for those edge cases additional functions are created. Markua provides in many cases the possiblity of adding some attributes to some Markua construct. The attribute lists are generated with the attrsToMarkua function as described in the previous chapter "Attributes". Markua does handle the escaping of many characters differently than e.g. GFM. This functionality is handled through the function escapeMarkuaString.

Limitations

As with most translations the Markua writer is not perfect. The majority of use cases should be covered but especially with special or rare cases a subpar output can result. These limitations are based on the limitations of Pandoc and its internal data representation, as well as some unspecified or unclear syntax in the Markua Manual. The later should be improved, when the specification for Markua 0.30 is finished, which contains more detailed information.

When not explicitly defined, the optimal translation was determined by the most common use case and/or the newer Markua 0.30 specification.

Further work

The current implementation provides a writer for Markua 0.10 with all standard features.

To further improve the Markua capabilities of Pandoc the following features would be suggested: - Create a Pandoc reader for Markua which can enable the need for a standalone Markua writer - when the specification for Markua 0.30 is done, update or complement the writer - improve edge case performance especially with nested blocks - possibly create new options in order give more control to the user (depends if allowed)

Glossary

- AST: Abstract Syntax Tree (Pandocs internal data format)
- GFM: Github flavored Markdown
- LFM: Leanpub flavored Markdown