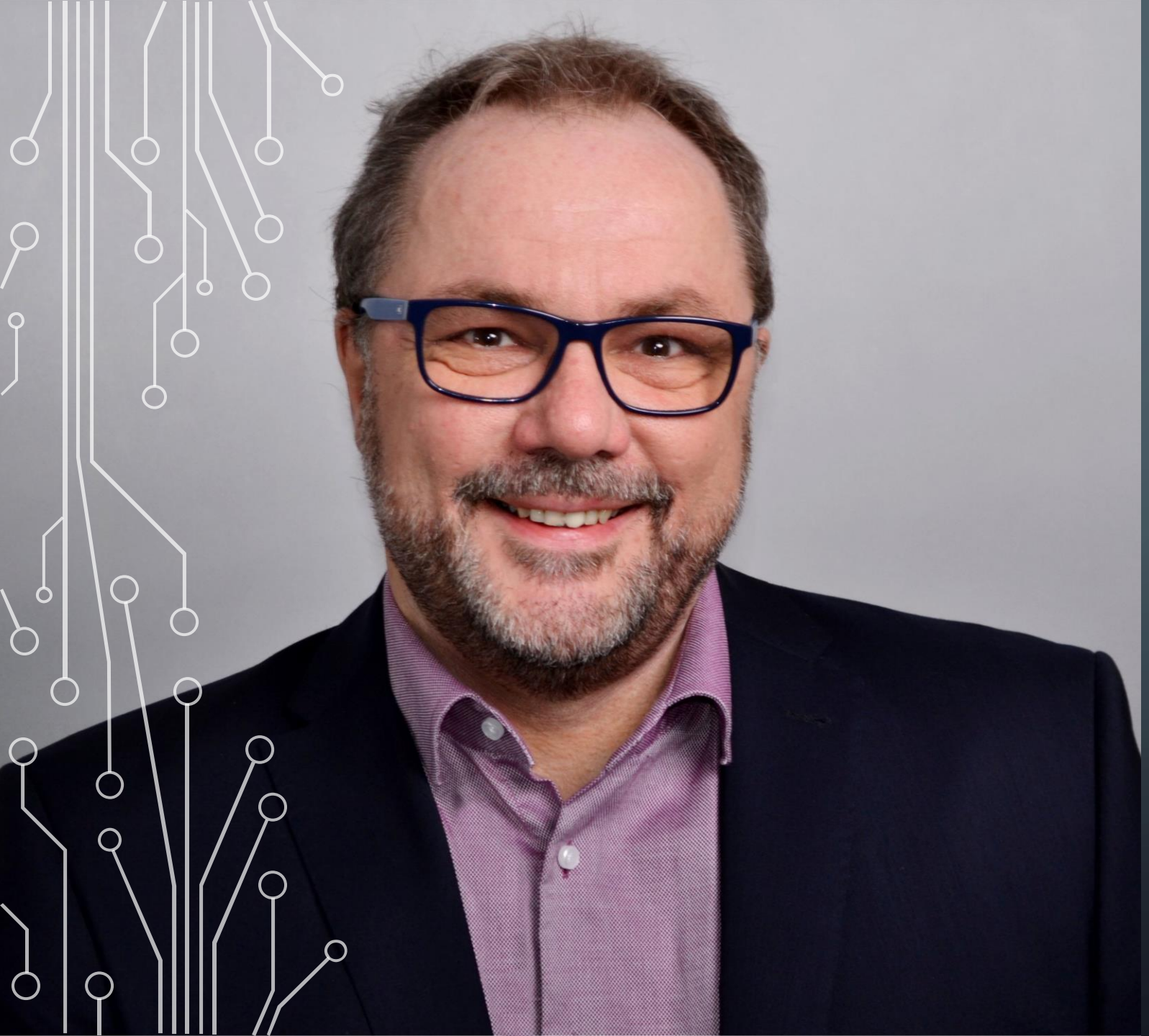
HOMEWORK

KRISTOF WERLING

KWERLINGIT GMBH

1 SEPT 2023



WHO AM I

- Kristof Werling
- 31 years of experience at HP: DevOp, Developer, Architect
- Started KwerlingIT GmbH
- Focus: IT / Cyber Security
- Security Audits, IT consulting, Trainings, Software creation

KwerlingIT GmbH
kristof.werling@kwerlingit.com
https://www.linkedin.com/in/kristof-werling/
https://www.kwerlingit.com

EXERCISE SUMMARY

- Create a Java program, which reads in a Markdown file (*.md) and translates it into HTML.

MARKDOWN & HTML

JUST WRITING SOME JAVA CODE AND GETTING TO KNOW SOME OF THE ODDITIES.

HEADINGS

Markdown	Html	Rendered Output
# Heading level 1	Heading level 1	Heading level 1
## Heading level 2	Heading level 1	Heading level 2
### Heading level 3	Heading level 1	Heading level 3
#### Heading level 4	Heading level 1	Heading level 4
##### Heading level 5	Heading level 1	Heading level 5
###### Heading level 6	Heading level 1	Heading level 6

PARAGRAPHS

Markdown	Html	Rendered Output
I really like using Markdown. I think I'll use it to format all of my documents from now on. Use a blank line to start / finish a paragraph	I really like using Markdown. I think I'll use it to format all of my documents from now on.	I really like using Markdown. I think I'll use it to format all of my documents from now on.

TEXT FORMATTING

Markdown	Html	Rendered Output
I just love **bold text**.	I just love bold text.	I just love **bold text**.
Italicized text is the *cat's meow*.	Italicized text is the cat's meow.	Italicized text is the *cat's meow*.
This text is ***really important***.	This text is really important.	This text is ***really important***.

ORDERED LISTS

Markdown	Html	Rendered Output
1. First item 2. Second item 3. Third item 4. Fourth item	 - First item - Second item - Third item - Fourth item 	1.First item 2.Second item 3.Third item 4.Fourth item
1. First item 1. Second item 1. Third item 1. Fourth item	 - First item - Second item - Third item - Fourth item 	1.First item 2.Second item 3.Third item 4.Fourth item

ORDERED LISTS

Markdown	Html	Rendered Output
1. First item 8. Second item 3. Third item 5. Fourth item	 - First item - Second item - Third item - Fourth item 	1.First item 2.Second item 3.Third item 4.Fourth item
1. First item 2. Second item 3. Third item 1. Indented item 2. Indented item 4. Fourth item	 - First item - Second item - Third item - Indented item - Indented item - Fourth item 	1.First item 2.Second item 3.Third item 1. Indented item 2. Indented item 4.Fourth item

UNORDERED LISTS

Markdown	Html	Rendered Output
- First item - Second item - Third item - Fourth item	 - First item - Second item - Third item - Fourth item 	•First item •Second item •Third item •Fourth item
* First item * Second item * Third item * Fourth item	 - First item - Second item - Third item - Fourth item 	•First item •Second item •Third item •Fourth item

UNORDERED LISTS

Markdown	Html	Rendered Output
+ First item + Second item + Third item + Fourth item	 - First item - Second item - Third item - Fourth item 	•First item •Second item •Third item •Fourth item
- First item - Second item - Third item - Indented item - Indented item - Fourth item	 - First item - Second item - Third item - Indented item - Indented item - Fourth item 	•First item •Second item •Third item • Indented item • Indented item •Fourth item

EXERCISE #1

- Familiarize yourself with the Markdown syntax (use the Github flavored one)
- For this exercise focus on headings and bold & italics formatting
- As for the java program:
  - Assume syntactically correct Markdown text
  - Start translation headings from MD into HTML. Document how the code determines if a heading is a heading and how long the heading text is. Ask yourself, if that rule might get conflict with other MD syntax.
  - If time permits, do so for the bold and italics formatting as well.

CHALLENGES TO SOLVE

- There are tags, which have the same start and end tag (I.e.: *, **). That should be fairly straightforward in coding.
- There can be nested / interwoven tags (i.e.: *italics ** italics bold * bold **). This example is really difficult to process.
- There are tags, which end with a \n (i.e: #, * (list))
- There are tags, which might be longer than one character (i.e.: #, ###)

INFORMATION ON MARKDOWN

URL	Info
https://github.github.com/gfm/	Github Flavored Markdown
https://github.blog/2017-03-14-a-formal-spec-for-github-markdown/	A formal spec for GitHub Flavored Markdown
https://en.wikipedia.org/wiki/Markdown	Wiki page on Markdown
https://babelmark.github.io/	Many Render Engines in comparison
https://commonmark.org/	CommonMark: A strongly defined, highly compatible specification of Markdown
https://www.smashingmagazine.com/2020/12/commonmark-formal-specification-markdown/	CommonMark: A Formal Specification For Markdown
https://spec.commonmark.org/0.30/	CommonMark: Spec v. 0.30 (latest)
I am sure you will find more	

SUMMARY OF ISSUES

- Markdown is not fully defined. Different Rendering Engines interpret it differently at times. There is no universaly agreed on Grammer / Syntax.
- Because of the possibility to have nested tags as well as interwoven tags makes it complicated
- Opposite to (for example) HTML there are tags, which either do not require an end tag to finish
- There are tags, which mean different things in different context (e.g.: *)

TOOLS TO USE

- Markdown Viewer (For example: Windows Markdown Viewer)
- Web browser of your choosing
- Makrdown Editor: IntelliJ

ADDITIONAL INFORMATION AFTER DAY 1

WHAT KIND OF GRAMMAR IS MARKDOWN

- Markdown is a non-context free grammar syntax:

  Context free grammars consist of rules, which can be applied to a non-terminal symbol independently of its context (== location in the markdown text).

  Ex.: * can start *kursiv* or **bold** or an unordered list. * can appear at the beginning of the line or anywhere in the text of the line.

BNF TO DESCRIBE GRAMMARS

- Backus-Naur Form is a formal way to describe a grammar
- It is a set of rules, which define the syntax of the grammar
- Rules consist of non-terminal symbols (= can be replaced by other rules) and terminal symbols (cannot be resolved by rules. Ex.: '+', 'A', '2', '#')
- BNF is usually used for context free grammar
- Let's still see how good it can be used for (a subset) of Markdown

BNF IN GRAPHICAL FORM

- The following graphics are to understand in this context:
  - The Grammar only covers parts of Markdown
  - It is not context-free, and therefore not that feasable for EBNF
  - It still shows more clearly how to process the Markdown file

EBNF RULES

Backus-Naur-Grammar for a subset of Markdown

(* This is the EBNF(kind of) for a subset of Markdown. The character set is strictly ASCII, mostly char 0 to char 127.

It is not really a EBNF Grammar, as there are some context dependent rules, such as Bold or Italics compared to UnorderedList.

But the EBNF notation provides us with a much better grip on the interpretation of Markdown.

*)

MDdocument = Blocks, eof;

Blocks = Block, { Blocks };

Block = Paragraph | Headline | Blockquote | List | Codeblock | HorizontalLine;

Headline = bol, ("#" | "##" | "###" | "####" | "#####" | "######"), Paragraph;

Paragraph = { String }, Newline;

Bold = (Asterisk, Asterisk), {String | Newline }, (Asterisk, Asterisk);

Italics = Asterisk, {String | Newline }, Asterisk;

EBNF RULES

List = UnorderedList | OrderedList;

UnorderedList = bol, { Space, Space, Space, Space }, Asterisk, InlineWhitespace, Paragraph;

OrderedList = bol, { Space, Space, Space, Space }, Number, InlineWhitespace, Paragraph;

Blockquote = "TBD";

Codeblock = "TBD";

HorizontalLine = "TBD";

Number = Digit, { Digit };

NonZeroDigit = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";

Digit = "0" | NonZeroDigit;

Newline = CR, LF | LF, CR | CR | LF;

InlineWhitespace = Tab | Space;

Tab = "\t";

Space = " ";

Whitespace = Space | Tab | LF | CR;

CR = "\r";

LF = "\n";

EBNF RULES

String = { Alphanum | Other | Specials | Backslash, Special | Slash | InlineWhiteSpace };

Alphanum = (Alphabet | Digit);

Alphabet = UCaseLetters1 | UCaseLetters2 | LCaseLetters1 | LCaseLetters2;

UCaseLetters1 = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" | "M";

UCaseLetters2 = "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z";

LCaseLetters1 = "a" | "b" | "c" | "d" | "e" | "f" | "g" | "h" | "i" | "j" | "k" | "l" | "m";

LCaseLetters2 = "n" | "o" | "p" | "q" | "r" | "s" | "t" | "u" | "v" | "w" | "x" | "y" | "z";

Asterisk = "*";

Backslash = "\";

Backtick = "`";

Slash = "/";

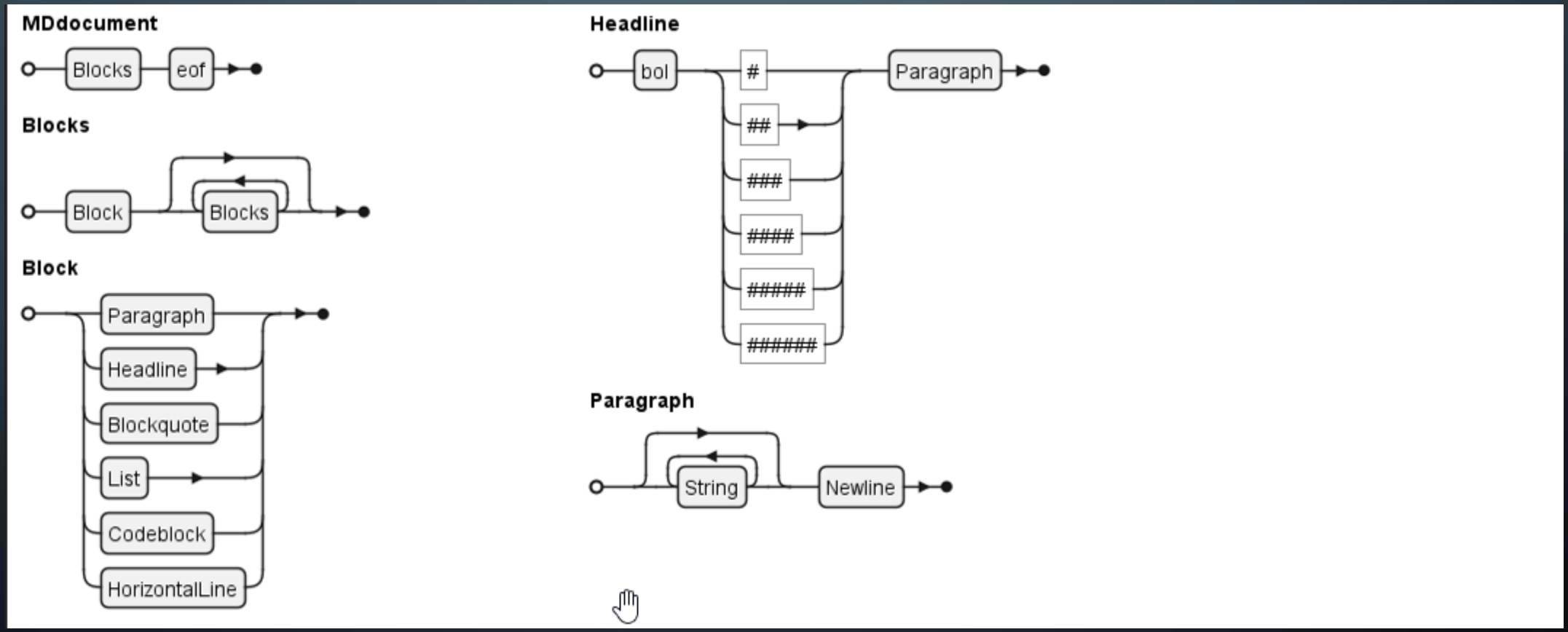
Others = "@" | "$" | "%" | "^" | "&" | "?" | "'" | "," | ";" | ":";

Special = "\" | "`" | "*" | "_" | "{" | "}" | "[" | "]" | "(" | ")" |
"#" | "+" | "-" | "." | "!";

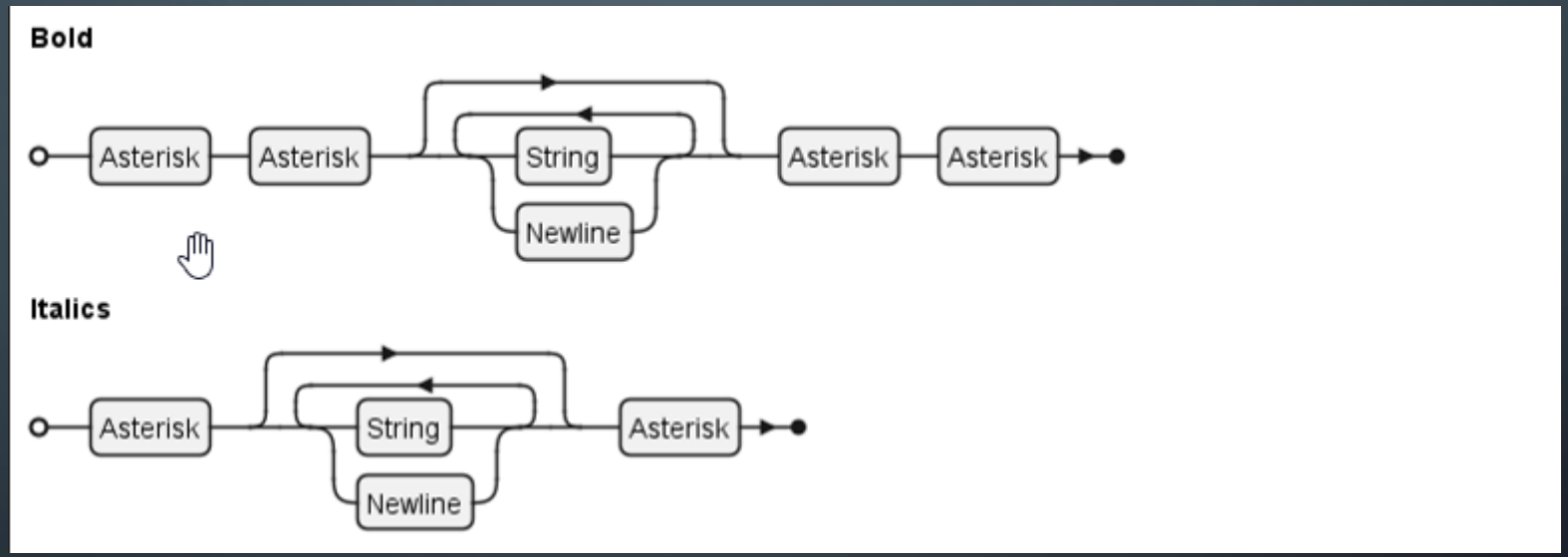
eof = "END OF FILE";

bol = "BEGINNING OF LINE";

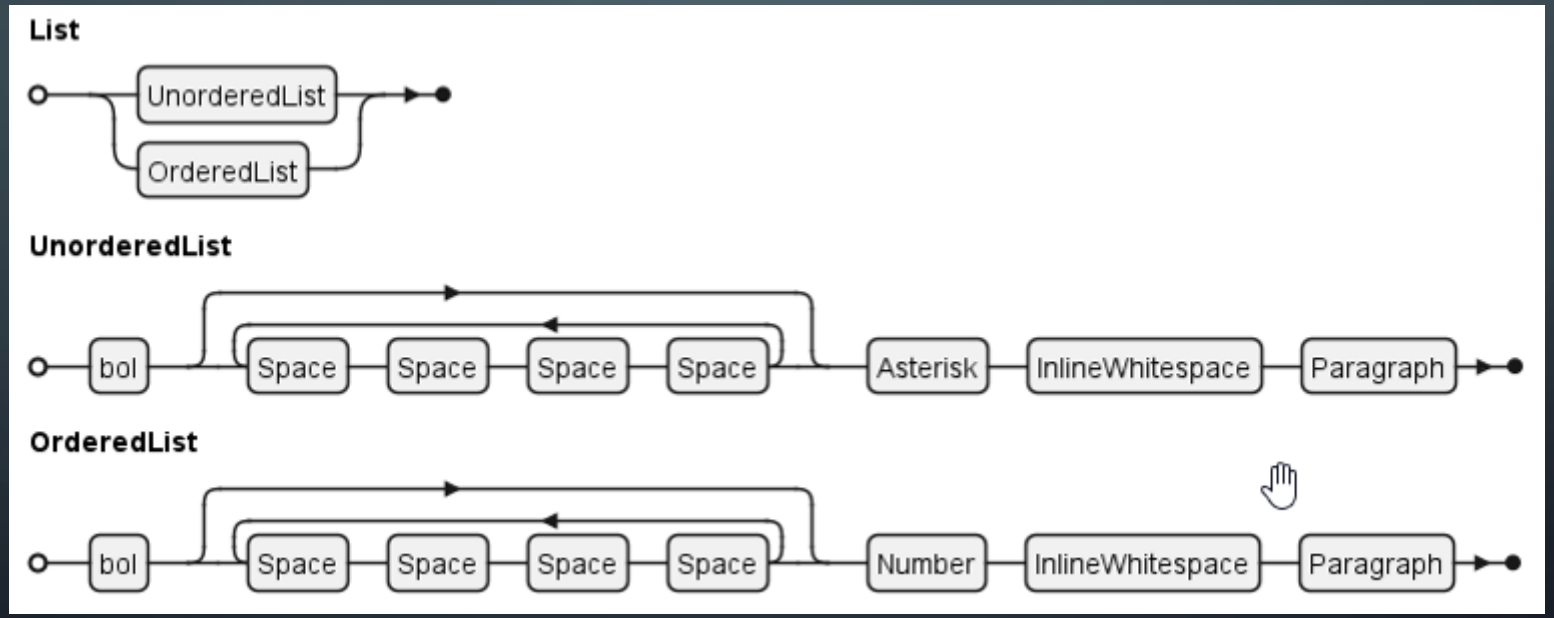
EBNF RULES


MDdocument
Blocks
eof
Blocks
Block
Blocks
Block
Paragraph
Headline
Blockquote
List
Codeblock
HorizontalLine
Headline
bol
#
##
###
####
#####
######
Paragraph
Paragraph
String
Newline

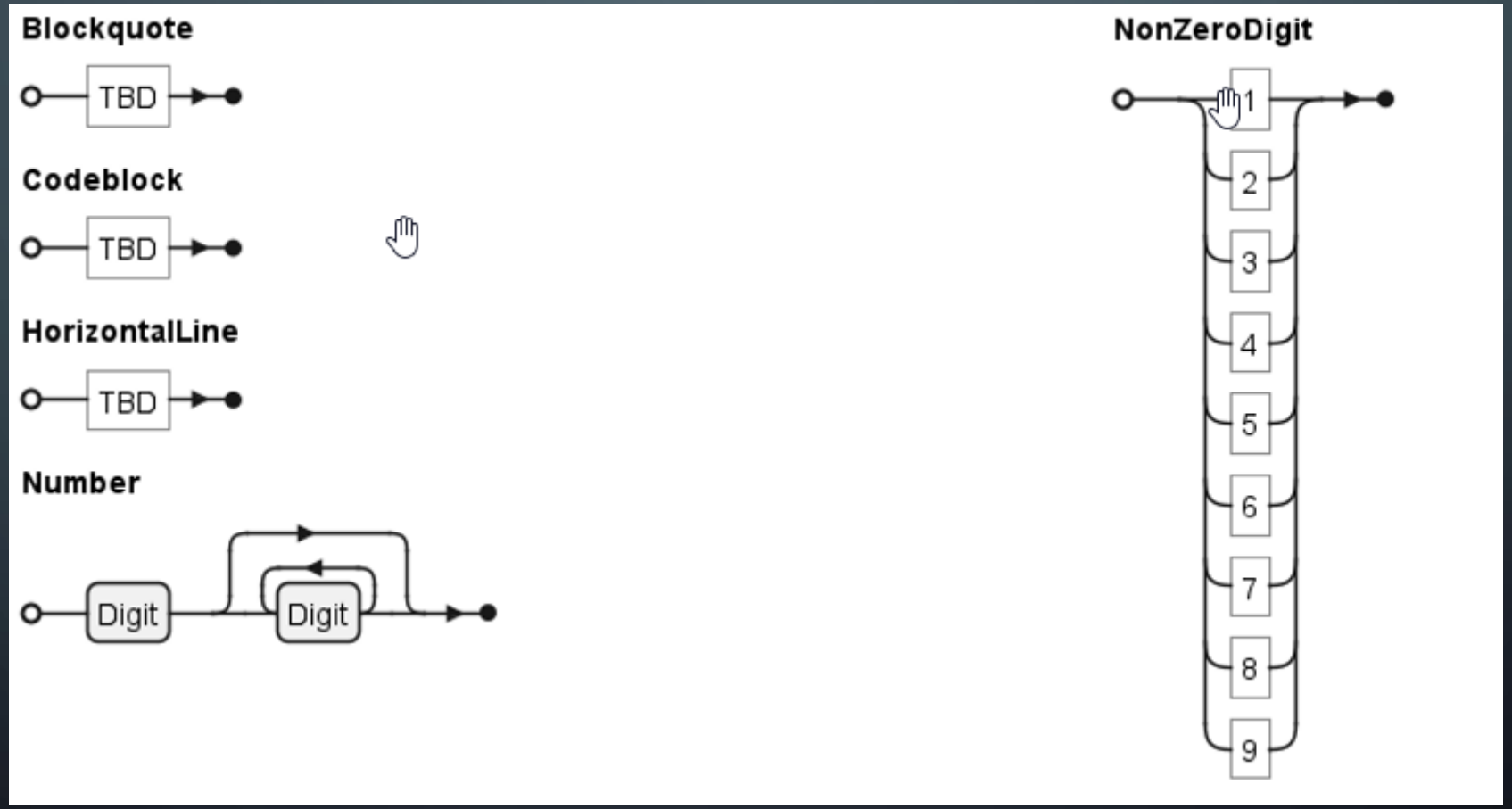
EBNF RULES


Bold
Asterisk
Asterisk
String
Newline
Asterisk
Asterisk
Italics
Asterisk
String
Newline
Asterisk

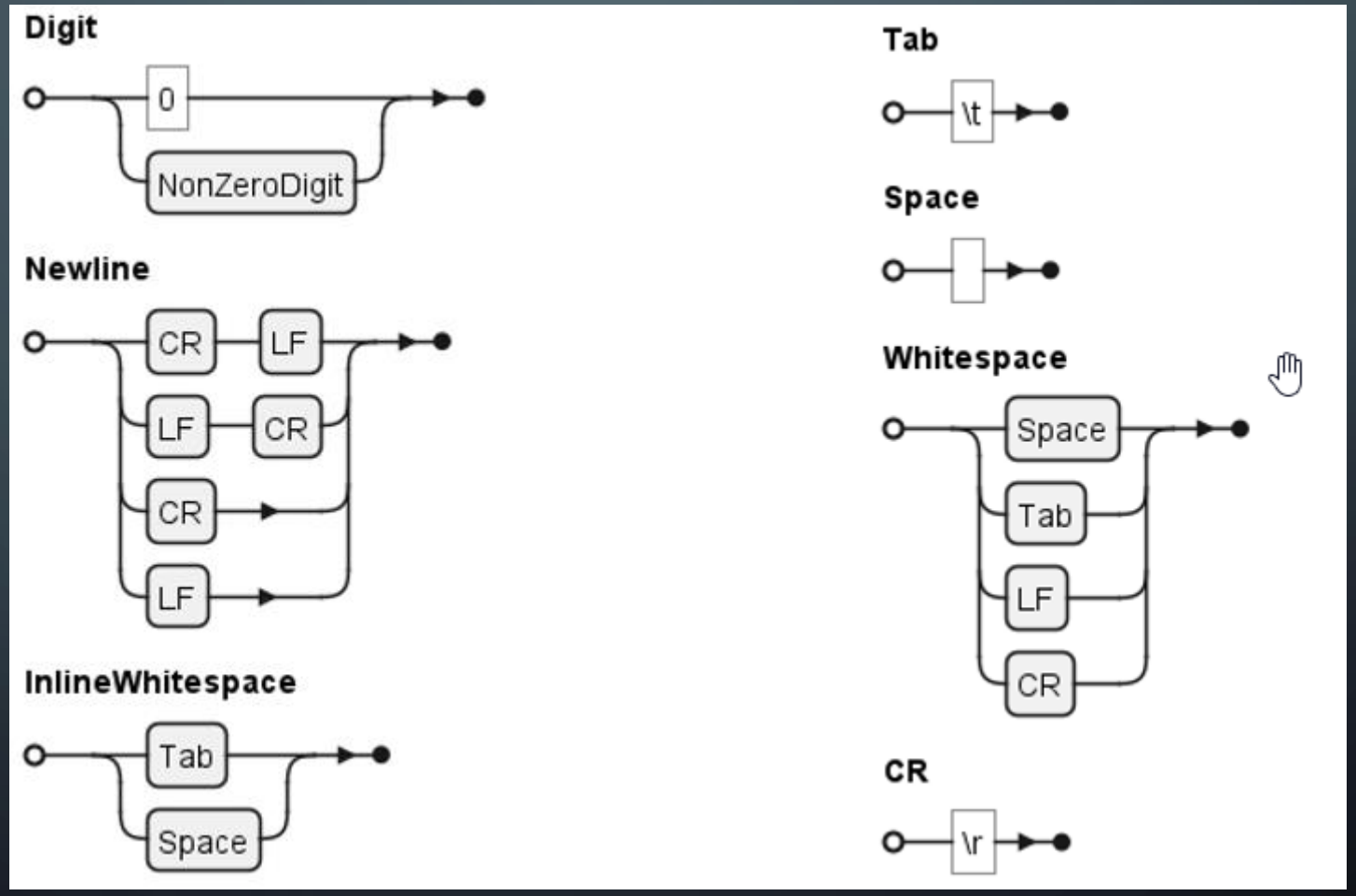
EBNF RULES


List
UnorderedList
OrderedList
UnorderedList
bol
Space
Space
Space
Space
Asterisk
InlineWhitespace
Paragraph
OrderedList
bol
Space
Space
Space
Space
Number
InlineWhitespace
Paragraph

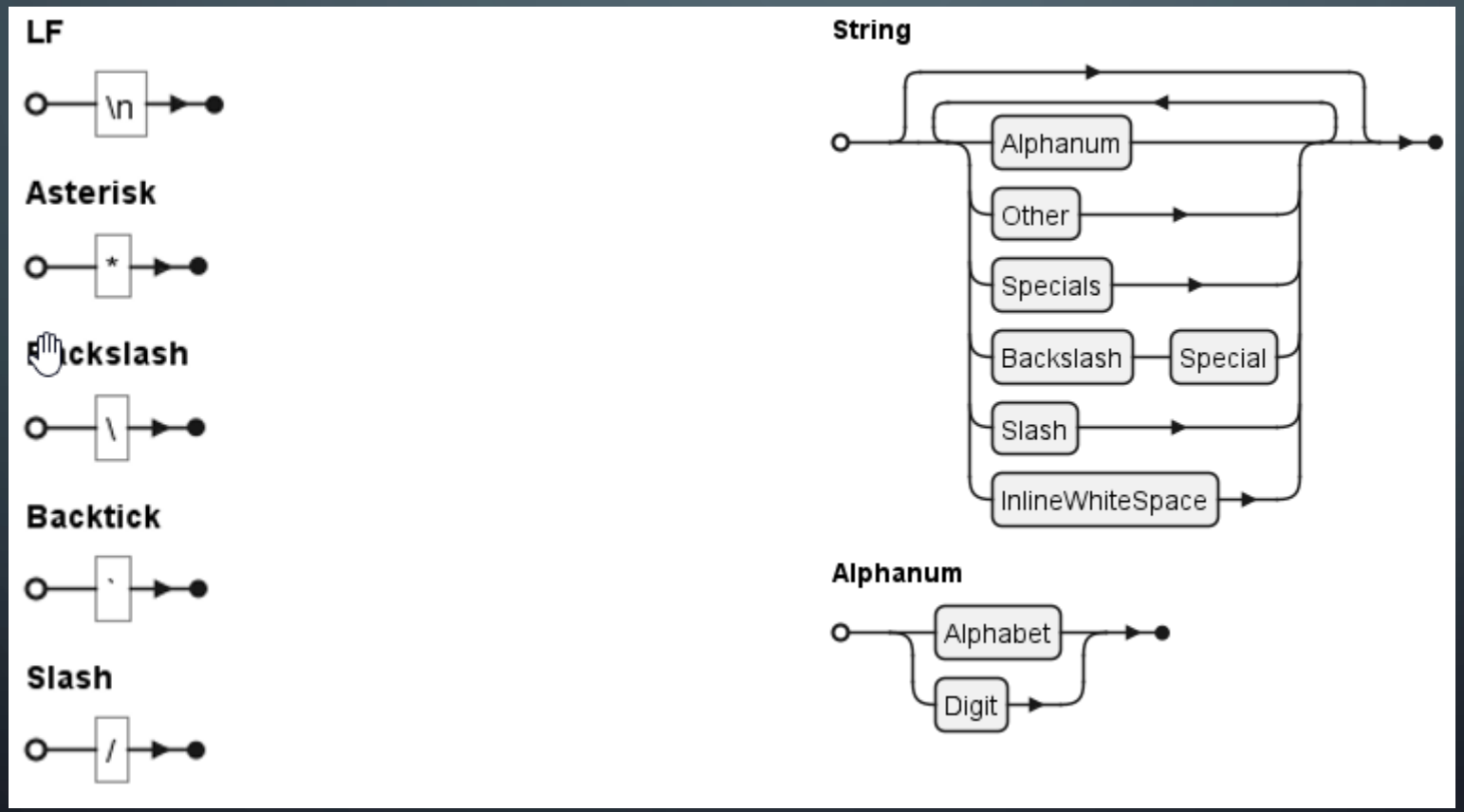
EBNF RULES


Blockquote
TBD
Codeblock
TBD
HorizontalLine
TBD
Number
Digit
Digit
NonZeroDigit
1
2
3
4
5
6
7
8
9

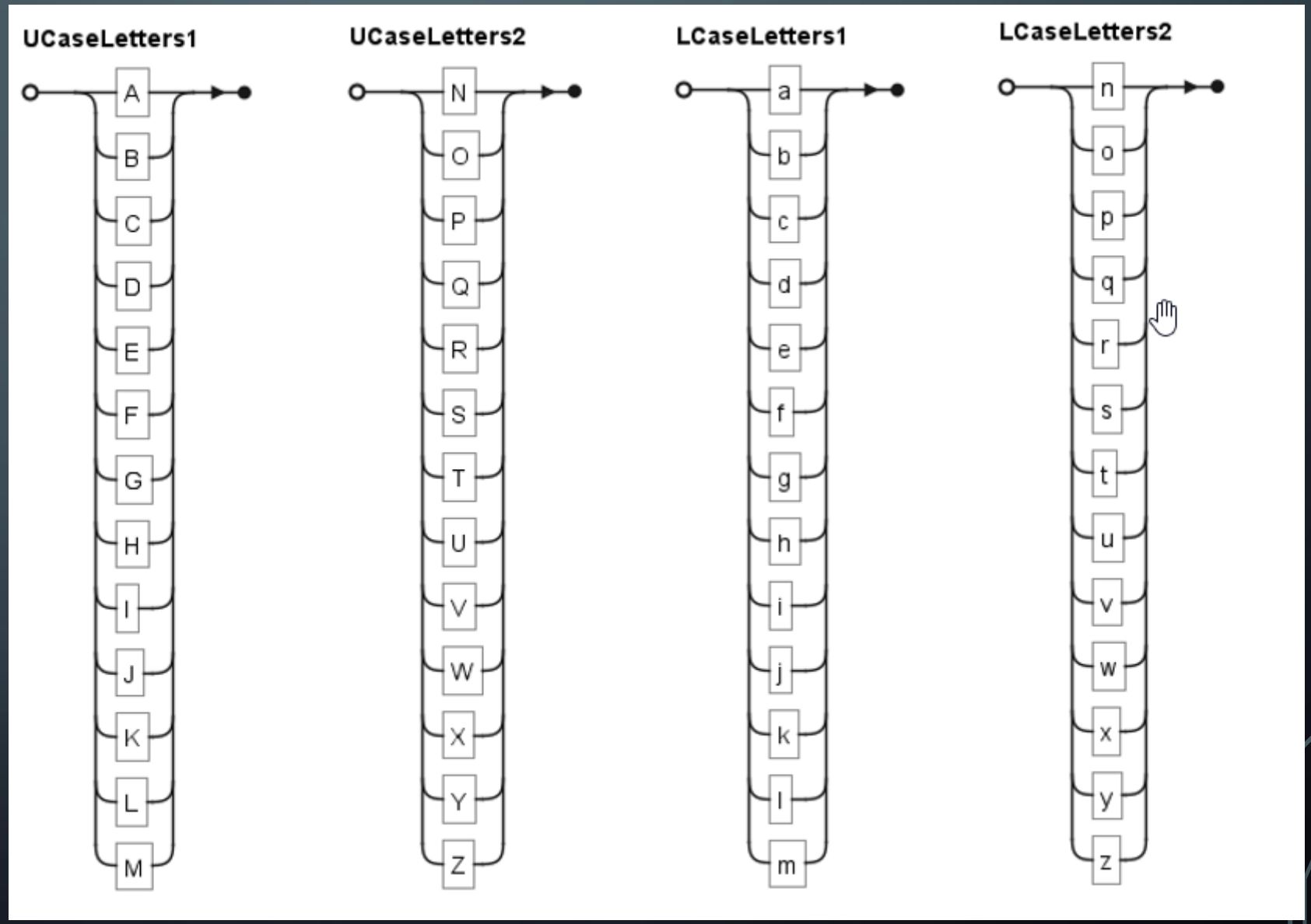
EBNF RULES


Digit
0
NonZeroDigit
Newline
CR
LF
LF
CR
CR
LF
InlineWhitespace
Tab
Space
Tab
\t
Space
Whitespace
Space
Tab
LF
CR
CR
\r

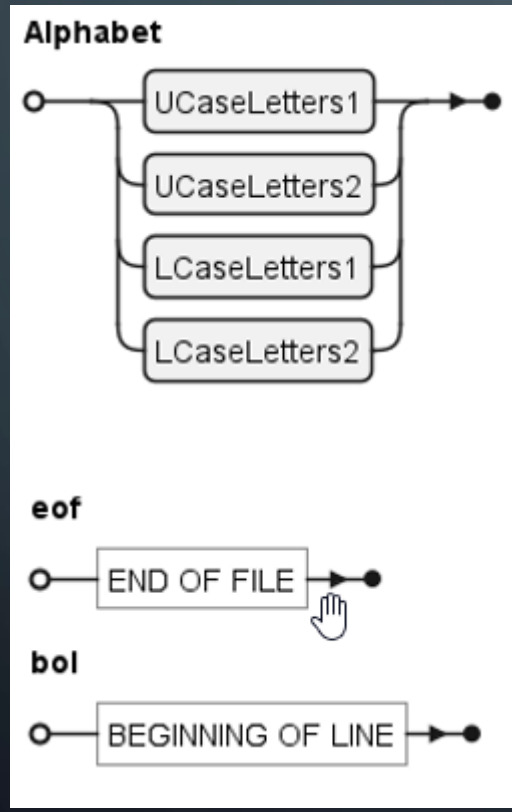
EBNF RULES

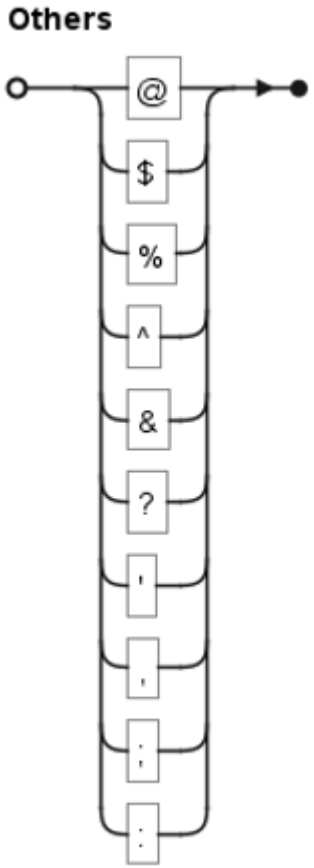

LF
\n
Asterisk
*
Backslash
\
Backtick
`
Slash
/
String
Alphanum
Other
Specials
Backslash
Special
Slash
InlineWhiteSpace
Alphanum
Alphabet
Digit

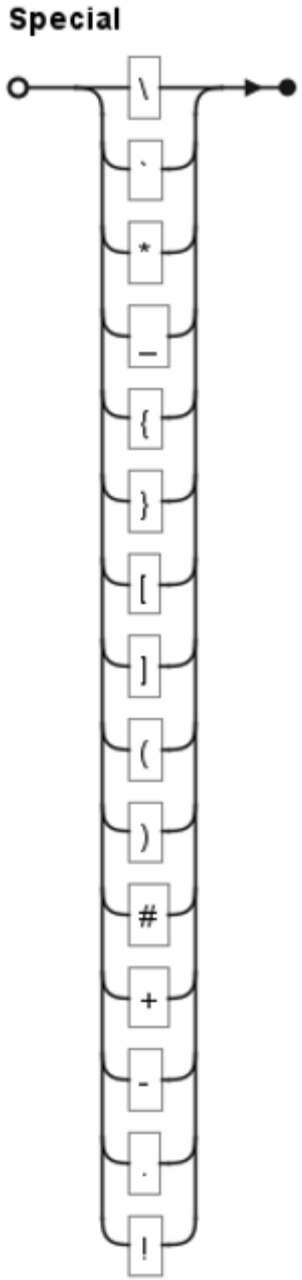
EBNF RULES


UCaseLetters1
A
B
C
D
E
F
G
H
I
J
K
L
M
UCaseLetters2
N
O
P
Q
R
S
T
U
V
W
X
Y
Z
LCaseLetters1
a
b
c
d
e
f
g
h
i
j
k
l
m
LCaseLetters2
n
o
p
q
r
s
t
u
v
w
x
y
z

EBNF RULES


Alphabet
UCaseLetters1
UCaseLetters2
LCaseLetters1
LCaseLetters2
eof
END OF FILE
bol
BEGINNING OF LINE


Others
@
$
%
^
&
?
'
,
;
:


Special
\
`
*
_
{
}
[
]
(
)
#
+
-
.
!

EBNF RULES

SVG Version of
BNF

ARE THERE OTHER WAYS TO APPROACH THE PROBLEM?

- (Finite) State Machines come to mind

- How would the solution look like expressed for a FSM?