

Neio: TeX for the 21st century

Titouan Vervack

Promotor: Professor Marko van Dooren

What is it?

- Markup language
- Inspired by TeX and Markdown

The good and bad: Markdown

- Not extensible
- Too simple (no customisation)
- No programming model
- + Easy to read/write
- + Very easy to get started

The good and bad: TeX

- Steep initial learning curve
- Allows redefinition of commands
- Overly complex
- Not statically typed
- + Well suited for complex documents
- + Has a programming model
- + Lots of packages

The good and bad: Word

- Corruptable fileformat
 - Not platform independent
 - Overkill for very simple documents
 - Not well suited for very complex documents
- + Very well known
 - + Easy to use for anyone

Goals

- Easy to get started with
- As powerful as TeX
- As simple as Markdown
- Static typesystem

Neio script files

- Markdownlike syntax
 - Adoptability
 - Easy to read/write
 - Non-corruptable

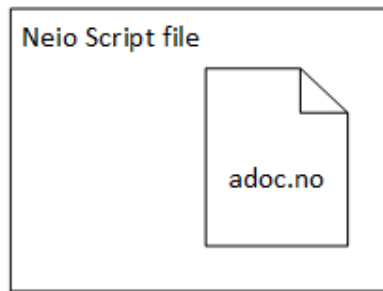
```
1 [Document]
2
3 # Chapter 1
4 This is the first paragraph of our simple document.
5
6 ## Chapter 1.1
7 This is a chapter of the next indentation level.
8
9 # Chapter 2
10 And here we have our last chapter.
```

Neio class files

- Javalike syntax
 - Adoptability
 - Easy translation

```
1 namespace neio.stdlib;
2
3 import neio.lang.Content;
4
5 class Chapter extends Content;
6
7 String title;
8 Integer level;
9
10 Chapter(String title, Integer level) {
11     this.title = title;
12     this.level = level;
13 }
14
15 Paragraph newline(String parText) {
16     Paragraph p = new Paragraph(parText);
17     addContent(p);
18
19     return p;
20 }
21
22 Image image(String caption, String imageName) {
23     Image neioImage = new Image(caption, imageName);
24     addContent(neioImage);
25
26     return neioImage;
27 }
28
29 nested Chapter #(String title, Integer level) {
30     if (level <= this.level) {
31         return nearestAncestor(Chapter.class).#(title, level);
32     }
33     Chapter chapter = new Chapter(title, level);
34     addContent(chapter);
35
36     return chapter;
37 }
```


Compile flow



Why output in java?

- Chameleon can output Java
- Java is statically typed
- Java is platform independent
- Java semantic \approx Neio semantic

Building a document

- Start out with documentclass
- Everything is a methodcall

```
1 [Document]
2
3 # Chapter 1
4 This is the first paragraph of our simple document.
5
6 ## Chapter 1.1
7 This is a chapter of the next indentation level.
8
9 # Chapter 2
10 And here we have our last chapter.
```

```
1 new Document()
2   .#("Chapter 1")
3     .newline("This is the first paragraph of our simple document.")
4       .##("Chapter 1.1")
5         .newline("This is a chapter of the next indentation level.")
6       .#("Chapter 2")
7         .newline("And here we have our last chapter.");
```

Building a document

```
1 new Document()  
2     .#("Chapter 1")  
3     .newline("This is the first paragraph of our simple document.")  
4     .##("Chapter 1.1")  
5     .newline("This is a chapter of the next indentation level.")  
6     .#("Chapter 2")  
7     .newline("And here we have our last chapter.");
```

```
7 public class simpleDocument {  
8     public static void main(String[] args) {  
9         neio.stdlib.Document $var0 = new Document();  
10  
11         neio.stdlib.Chapter $var1 = $var0.hash("Chapter 1");  
12         neio.stdlib.Paragraph $var2 = $var1.newline("This is the first paragraph of our simple document.");  
13  
14         neio.stdlib.Chapter $var3 = $var1.hash("Chapter 1.1", 2);  
15         neio.stdlib.Paragraph $var4 = $var3.newline("This is a chapter of the next indentation level.");  
16  
17         neio.stdlib.Chapter $var5 = $var0.hash("Chapter 2");  
18         neio.stdlib.Paragraph $var6 = $var5.newline("And here we have our last chapter.");  
19  
20         java.lang.String $var7 = new TexFileWriter($var0).write("simpleDocument");  
21         new TexToPDFBuilder().build($var7);  
22     }  
23 }
```

Building a document: result

Chapter 1

This is the first paragraph of our simple document.

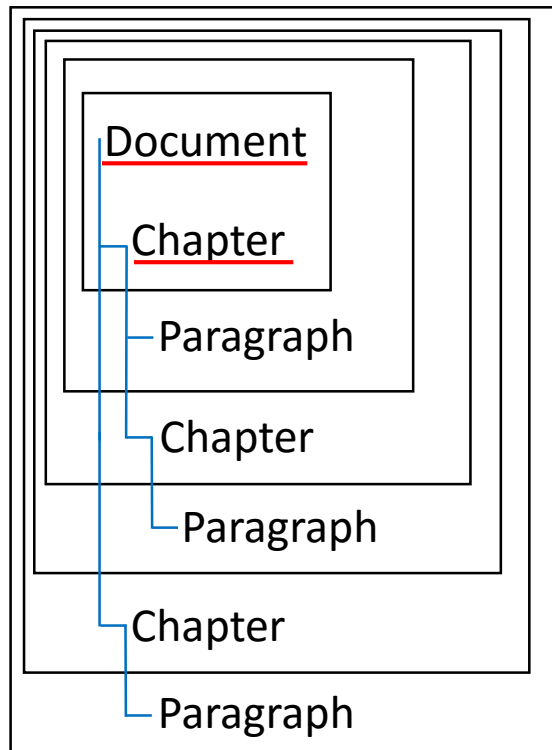
Chapter 1.1

This is a chapter of the next indentation level.

Chapter 2

And here we have our last chapter.

Context types



```
1 [Document]
2
3 # Chapter 1
4 This is the first paragraph of our simple document.
5
6 ## Chapter 1.1
7 This is a chapter of the next indentation level.
8
9 # Chapter 2
10 And here we have our last chapter.
```

```
1 new Document()
2     .#("Chapter 1")
3     .newline("This is the first paragraph of our simple document.")
4     .##("Chapter 1.1")
5     .newline("This is a chapter of the next indentation level.")
6     .#("Chapter 2")
7     .newline("And here we have our last chapter.");
18 neio.stdlib.Paragraph $var6 = $var5.newline("And here we have our last chapter.");
19
20 java.lang.String $var7 = new TexFileWriter($var0).write("simpleDocument");
21 new TexToPDFBuilder().build($var7);
22
23 }
```

Nested methods

- Allow for nesting of method
- nested type f(arg, Integer)

```
29 nested Chapter #(String title, Integer level) {  
30     if (level <= this.level) {  
31         return nearestAncestor(Chapter.class).#(title, level);  
32     }  
33     Chapter chapter = new Chapter(title, level);  
34     addContent(chapter);  
35  
36     return chapter;  
37 }
```

```
## Chapter 1.1  
This is a chapter of the next indentation level.  
  
# Chapter 2  
And here we have our last chapter.
```

```
neio.stdlib.Chapter $var3 = $var1.hash("Chapter 1.1", 2);  
neio.stdlib.Paragraph $var4 = $var3.newline("This is a chapter of the next indentation level.");  
  
neio.stdlib.Chapter $var5 = $var0.hash("Chapter 2");  
neio.stdlib.Paragraph $var6 = $var5.newline("And here we have our last chapter.");
```

Customisability: inheritance

```
1 namespace neio.mypackage;
2
3 import neio.stdlib.Document;
4
5 class MyDocument extends Document;
6
7 String header() {
8     String s = super.header();
9     return s + "This is my document!";
10 }
```

```
1 [MyDocument]
2
3 # Chapter 1
4 This is the first paragraph of our simple document.
5
6 ## Chapter 1.1
7 This is a chapter of the next indentation level.
8
9 # Chapter 2
10 And here we have our last chapter.
```

This is my document!

Chapter 1

This is the first paragraph of our simple document.

Chapter 1.1

This is a chapter of the next indentation level.

Chapter 2

And here we have our last chapter.

Customisability: executing code

```
1 [Document]
2
3 # Chapter 1
4 This is the first paragraph of our simple document.
5
6 `#("Chapter 1.1", 2)`
7
```

```
neio.stdlib.Document $var0 = new Document();
neio.stdlib.Chapter $var1 = $var0.hash("Chapter 1");
neio.stdlib.Paragraph $var2 = $var1.newline("This is the first paragraph of our simple document.");

{
    $var1.hash("Chapter 1.1", 2);
}
```

Customisability: executing code

```
8 `
9 Chapter chap = new Chapter("Chapter 2", 1);
10 chap.newline("This is a coded paragraph");
11 addChapter(chap);
12 `
```

```
{
  Chapter chap = new Chapter("Chapter 2", 1);
  chap.newline("This is a coded paragraph");
  $var0.addChapter(chap);
}
```



```
8 ``
9 Chapter chap = new Chapter("Chapter 2", 1);
10 chap.newline("This is a coded paragraph");
11 addChapter(chap);
12 ``
```

```
Chapter chap = new Chapter("Chapter 2", 1);
chap.newline("This is a coded paragraph");
$var0.addChapter(chap);
```

Add content through code

Chapter 1

```
1 [Document]
2
3 # Chapter 1
4
5 ``return new Itemize()``
6 * First
7 * Second
8 * Third
```

- First
- Second
- Third

```
neio.stdlib.Document $var0 = new Document();
neio.stdlib.Chapter $var1 = $var0.hash("Chapter 1");


neio.stdlib.Itemize $var2 = $var1.addContent(new Itemize());
neio.stdlib.ItemizeItem $var3 = $var2.star("First");
neio.stdlib.ItemizeItem $var4 = $var2.star("Second");
neio.stdlib.ItemizeItem $var5 = $var2.star("Third");
```

Customisability: custom commands

```
1 namespace neio.mypackage;
2
3 import neio.stdlib.Document;
4
5 class MyDocument extends Document;
6
7 private Integer defaultFontSize = 12;
8
9 void increaseFontSize() {
10     defaultFontSize = defaultFontSize + 2;
11 }
12
13 void setFontSize(Integer defaultFontSize) {
14     this.defaultFontSize = defaultFontSize;
15 }
16
17 String header() {
18     String s = super.header();
19     return s + "This is my document!";
20 }
```

```
1 [MyDocument]
2
3 \increaseFontSize
4 # Chapter 1
5 This is the first paragraph of our simple Document.
6
7 \setFontSize(12)
8 ## Chapter 1.1
9 This is a chapter of the next indentation level.
```

Future work before hand in

- Slideshows
- Surround methodes **bold text**  **bold text**
- Further controles on static typing
- Create more documentclasses (article, letter, book,...)

Future work

- Compiler optimizations
- Aliases for methods (e.g. alias star for *)
- Automatic double latex compile detection
- Native output to languages other than latex
- Further implementation of commonly used latex packages