

(empty backward page)

Delete this whole page if you want to have a *book view* document.

Title

Subtitle
2024-07-10

Text-Intro, very short.

Table of Contents

<i>1 Approaches</i>	4
<i>2 Implementation</i>	6

1 Approaches

Table of Contents

1 Approaches.....	4
1.1 Heading2.....	4
1.2 Another chapter.....	4

1.1 Heading2

Long lines in text without columns

LibreOffice and a textual Markup language, for example AsciiDoc or LaTeX are two very different approaches to write (technical) documentation. Both have advantages and disadvantages.

The here used Markup language is ZmL It is specific defined proper to LibreOffice.

One intention to use ZmL and LibreOffice parallel for the same document is: LibreOffice has the disadvantage that “*what you see is **what you have***” is not true. It follows the known approach “*What you see is what you get*”, but some stuff is hidden which should be more obviously — The advantage of AsciiDoc is: You see what you have. For example specific formats (styles) with its names, exact written relative link, etc. AsciiDoc is a source format, it is a plain text without hidden stuff.

Text in columns possible

Writing in Columns has the advantage, that the eye of the reader can move only vertical, because of the limited line length.

Also small images can be assigned in a column.

This is the same text as above:

One intention to use ZmL and LibreOffice parallel for the same document is:

1.2 Another chapter

But the columns does not go via the whole page. They ends on a larger image, or a chapter title etc.

Direct styles are not supported. Italic and bold are translated to `Quotation` and `Emphasis`.

- A list with paragraph style `List1`. The bullet is a written char, also the bullet points are normal character in UTF fonts.

- And this list is `List1Left` without indent on left side to save space.

LibreOffice has the disadvantage that “*what you see is **what you have***” is not true. It follows the known approach “*What you see is what you get*”, but some stuff is hidden which should be more obviously — The advantage of AsciiDoc is: You see what you have. For example specific formats (styles) with its names, exact written relative link, etc. AsciiDoc is a source format, it is a plain text without hidden stuff.

- And also `List2Left` and `List2` are available.

- This bullet forms are encoded in ZmL. All UTF characters or any text are possible.

And here is normal text again.

An image is placed always to its paragraph of style `ImgCaptionText`. Free positioning of images must not be used. They are always bounded to text. This is similar as in HTML or also AsciiDoc. But it is sometimes a little bit sophisticated with images on the end of pages or columns.

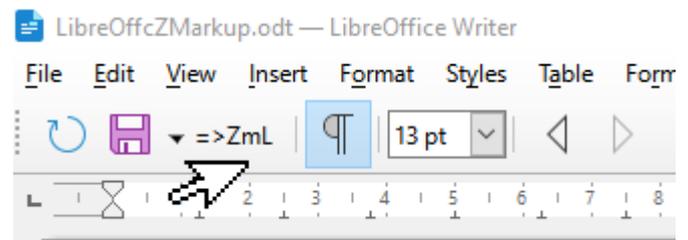


Figure 1: ZmL-icon.png

That's why you can use the `ImgCaptionTextCol` used here to break the column and place the image in top of the next column, or just `ImgCaptionTextPg`.

How to show code:

Simple with a code style, here

```
let one line free to insert outside of the columns.CodeCpp.
More as one line are the next paragraph.
```

But you can also automatically include code from sources, write it in the ZmL plain text and generate this docu newly.

```
include:../../ExampleCode/cpp/
example.c::labelSnippet::45
void operation(float X) //
    return 3.5 * x;      ...marker
}
```

A horizontal line

What about tables? Very important, runs but not yet elaborately testes.

Links to generated HTML pages are supported, tested with Javadoc, with automatic supplementation of there sophisticated target level inside the HTML.

Note: If you want to have a really empty paragraph, you should give him a style which is not Text or Text body. The next empty paragraph has the style "`AddInfo`".

This is `AddInfo`.

What else= See https://vishia.org/LibreOffc/html/Videos_LOffcZmL.html

This works also for columns because here the line width is limited to 45, the code lines are truncated. Note, that the code is usual only an illustration how does it work and not a source for copy the code.

You can use the `marker` with character style `cm` for explaining.

To work in both, LibreOffice and ZmL with a plain text or AsciiDoc editor, Save always the ZmL on end of work, regenerate LibreOffice, save again till the result is equal. The Odt to ZmL converter makes some corrections for line breaking etc. which does not affect the text and formatting. It is only formally. Changing ZmL, converting to odt and back converting to ZmL cleans the situation. It is similar source code development. Higher effort, but if all runs, all is proper.

2 Implementation

Table of Contents

2 Implementation.....	6
2.1 Sub chapter.....	6

2.1 Sub chapter

Usual used styles are:

Code block appearance

Simple code block
with some lines.

Cmd line
or file tree presentation

REM A windows batch file
or a shell script

##Some configuration data
a = "test"

```
void javaOperation(float arg) {
    return;
}
```

```
void cppOperation(float arg) {
    return;
}
```

```
##This is a otx script:
<:otx: VarV_UFB: evSrc, fb, evin, din>
```

A Zml code snippet <:c:code characters.>

Copy this part in your document to copy the styles, and to see how the styles appear.

P-style, C-style and appearance:

- Code, ccode: And here is simple code
- CodeCmd, cCmd: this is a cmd call arguments example
- CodeScript, cS: A part of a script
- CodeCfg: cCfg: config data some configuration data
- CodeJava, cJ: javaOperation with arguments
- CodeCpp, cc: also C or C++ language cppOperation() given
- CodeOtx, cOtx: A specific code style in line written as <:otx: VarV_UFB:
- CodeZmL, cZmL: Specific code style for this topic <:c:code characters.>
- cM: A Marker should be used also inside code blocks and in the explanation. Should look demonstrative

last full line after columns