Structured description of ConTEXt commands

Peter Münster

September 14, 2010

Topics

Introduction

The following slides present a possibility, to group all details about a ConTEXt command in a well structured and readable “command-description-file”.

The language of this file is Lua, because

- it’s a native language of LuaTEX
- and it’s convenient to read and to edit by human beings (better than XML)

Motivation

Main goal: getting one day the *Complete ConTEXt Command Reference*. With:

- detailed descriptions of the macros and their arguments
- classification in categories (chapters and sections)
- index
- cross-references
- perhaps various output formats (PDF, HTML, various styles, ...)

There are several hundreds of commands to describe, so in order to get there in the most efficient way, we must be able to concentrate on the content:

- ergonomic editing environment (using your favorite editor)
- not too verbose structuring syntax (Lua is nice!)
- avoidance of redundancy (description of inheritance)

Furthermore, the command descriptions can be used for a future ConTEXt syntax checker!

Examples

In the following examples, you will see, that there is no concept for optional arguments.

This is so, because in ConTEXt

- sometimes you need to add argument 1 to make use of argument 2 (e.g. `\placefigure`)
- sometimes the meaning of arguments depends on the number of supplied arguments (e.g. `\setupheadertexts`)
- sometimes the position of an argument varies (e.g. `\externalfigure[ref][file]` and `\externalfigure[file][parent]`)

Instead, we use *variants* to describe the different possibilities to call a macro.

blank

```
return {
    name = "blank", -- redundant (same as filename), but nice to see at top
    comment = "Adds space between 2 paragraphs.", -- short description
    categories = {"Spacing"},
    arguments = {                 -- list of all arguments, order doesn't matter
        keyword_list = {          -- just an arbitrary name for this argument
            type = "keywords",
            default = "big",
            keywords = {
                small = "insert small space",
                big = "insert big space",
                samepage = "no page break here",
            },
        },
    },
```

Examples

```
        variants = {  -- one variant for each possibility to call the macro
            {
                comment = "Just use the default setting.",
            },
            {
                comment = "Specify some keywords.",
                "keyword_list",
            },
        }
    }
```

placefloat

```
local t = {
    name = placefloat,
    comment = "Insert a floating element.",
    description = [[This command must be followed by a new paragraph
                    (empty line).]],
    fixpart = "place",
    varpart = "float",
    generator = "definefloat",
    categories = {"Structure", "Graphic"}, -- First category is main category,
    arguments = {          -- in the other categories, there are just references.
        keywords = {
            comment = "Where to place the float.",
            type = "keywords",
            default = "here",
        },
        label = {
            comment = "Label for referencing.",
```

Examples

```
            type = "label",
        },
        caption = {
            comment = "Text of the caption.",
            description = "If \quote{none}, then no caption is placed.",
            type = "text",
        },
        content = {
            comment = "Content of the float.",
            type = "text",
        },
    },
    variants = {
        {
            comment = "No placement option, no label.",
            "caption", "content",
        },
        {
            comment = "With placement option.",
```

Examples

```
            "keywords", "caption", "content",
        },
        {
            comment = "With placement and label.",
            description = [[If you need a label, but you do not want to
                            use the placement option, just leave the first
                            brackets empty.]],
            "keywords", "label", "caption", "content",
        },
    },
}

-- better readable if outside the arguments-table:
t.arguments.keywords.keywords = {
    left = "place float on left side",
    right = "place float on right side",
    here = "place float right here",
}
return t
```

startchapter

```
local t = {
    name = "startchapter",
    comment = "Defines a new chapter.",  -- short description (one-liner)
    description = "Perhaps longer description...",
    environment = true,         -- default is false
    categories = {"Structure"},
    arguments = {
        settings = {
            comment = "Settings for the chapter.",
            type = "settings",
        },
    },
    variants = {
        {
            comment = "The only variant.",
            "settings",
        },
```

Examples

```
    },
}

t.arguments.settings.settings = { -- don't clutter the arguments-table...
    title = {
        comment = "The title of the chapter.",
        values = {"text"},
    },
    incrementnumber = {
        values = {"yes", "no", "list"},
        default = "yes",
    },
}

return t
```

externalfigure

```
return {
    name = "externalfigure",
    comment = "Insert a figure from an external file.",
    categories = {"Graphic"},
    arguments = {
        reference = {
            comment = "For referencing a predefined setting.",
            type = "label",
        },
        file = {
            comment = "Filename with the figure to insert.",
            description = "The name does not need a suffix."
            type = "file",
        },
        settings = {
            comment = "short comment for this option",
            description = "long description for this option",
```

Examples

```
            type = "settings",
            inherit = "useexternalfigure",
        },
        parent = {
            comment = "I don't know...",
            type = "parent",
        },
    },
    variants = {
        {"file"},
        {"reference", "file"},
        {"reference", "file", "settings"},
        {"file", "parent"},
    },
    examples = { -- to be put perhaps into externalfigure-examples.tex
        [=[\externalfigure[hacker][width=3cm]]=],
        [=[\externalfigure[my label][hacker][frame=on]]=],
    }
}
```

setupframed

```
local t = {
    name = "setupframed",
    comment = "Setting up \cmd{framed}.",
    categories = {"Packaging"},
    arguments = {
        name = {
            comment = "Name of command defined with \cmd{defineframed}",
            type = "label",
        },
        settings = {
            comment = "Some settings.",
            type = "settings",
        },
    },
    variants = {
        {
            comment = "The simple variant.",
```

Examples

```
                "settings",
            },
            {
                comment = "Setting up a command defined with \cmd{defineframed}.",
                "name", "settings",
            },
        },
    }

    -- better readable if outside the arguments-table:
    t.arguments.settings.settings = {
        height = {
            values = {"fit", "broad", "dimension",},
        },
        width = {
            values = {"fit", "broad", "fixed", "local", "dimension",},
        },
        autowidth = {
            values = {"yes", "no", "force",},
```

Examples

```
},
offset = {
    values = {"none", "overlay", "default", "dimension",},
},
location = general_settings"location",
option = {
    values = {"none", "empty",},
},
strut = general_settings"strut",
align = general_settings"align",
bottom = {
    values = {"command",},
},
top = {
    values = {"command",},
},
frame = {
    values = {"on", "off", "none", "overlay",},
    default = "on",
```

Examples

```
        },
        topframe = {
            values = {"on", "off",},
        },
        background = {
            values = {"screen", "color", "none", "foreground", "name",},
        },
        backgroundscreen = {
            values = {"number",},
        },
        backgroundcolor = {
            values = {"name",},
        },
        depth = {
            values = {"dimension",},
        },
    }

    return t
```

framed

```
return {
    name = "framed",
    comment = "Make box with a frame.",
    categories = {"Packaging"},
    validated = false, -- only qualified people (Hans, Wolfgang, etc.)
                       -- should set this to true
    arguments = {
        settings = { -- it must be just the same as the name in setupframed
            comment = "short comment for this option",
            description = "long description for this option",
            type = "settings",
            inherit = "setupframed",
        },
        content = {
            comment = "Stuff to put into the box.",
            type = "text",
        },
```

Examples

```
        },
        variants = {
            {
                comment = "No special settings.",
                "content"
            },
            {
                comment = "With some local settings.",
                "settings", "content"
            },
        },
    }
```

placebookmarks

```
return {
    name = "placebookmarks",
    comment = "Specify heads for which bookmarks should be created.",
    categories = {"Interaction"},
    validated = false,
    arguments = {
        head_list = {
            comment = "List with heads to create bookmarks for.",
            type = "label",
        },
        open_list = {
            comment = "List with heads whose bookmarks are open.",
            type = "label",
        },
        settings = {
            comment = "Some settings.",
            type = "settings",
```

Examples

```
                settings = {
                    force = {
                        comment = "Why is this needed?",
                        values = {"no", "yes",},
                        default = "no",
                    },
                    number = {
                        comment = "Whether to place the section numbers.",
                        values = {"no", "yes",},
                        default = "no",
                    },
                },
            },
        },
        variants = {
            {"head_list"},
            {"head_list", "open_list"},
            {"head_list", "open_list", "settings"},
            {"head_list", "settings"},
```

Examples

```
        }
    }
```

setupheadertexts

```
return {
    name = "setupheadertexts",
    comment = "Insert text in the header line."
    description = "A lot of text..."
    categories = {"Layout"},
    arguments = {
        text_middle = {
            comment = "Text placed in the centre of the header.",
            type = "text",
            square_brackets = true, -- needed because
                                    -- type = "text" implies braces
        },
        text_left = {
            comment = "Text placed at the left side.",
            type = "text",
            square_brackets = true,
        },
```

Examples

```
text_right = {
    comment = "Text placed at the right side.",
    type = "text",
    square_brackets = true,
},
text_left_odd = {
    comment = "Text placed at the left side of odd pages.",
    type = "text",
    square_brackets = true,
},
text_right_odd = {
    comment = "Text placed at the right side of odd pages.",
    type = "text",
    square_brackets = true,
},
```

Examples

```
        text_left_even = {
            comment = "Text placed at the left side of even pages.",
            type = "text",
            square_brackets = true,
        },
        text_right_even = {
            comment = "Text placed at the right side of even pages.",
            type = "text",
            square_brackets = true,
        },
    },
```

Examples

```
        variants = {
            {
                comment = "Very simple usage.",
                "text_middle",
            },
            {
                comment = "Two elements are placed in the header.",
                "text_left", "text_right",
            },
            {
                comment = "Usage for double-sided layout.",
                "text_left_odd", "text_right_odd",
                "text_left_even", "text_right_even",
            },
        },
    }
```

startitemgroup

```
local t = {
    name = "startitemgroup",
    comment = "Creates an item list.",
    environment = true,
    varpart = "itemgroup",
    generator = "defineitemgroup",
    categories = {"Structure"},
    arguments = {
        keywords = {
            comment = "Control the layout.",
            type = "keywords",
            default = "standard",
        },
        settings = {
            comment = "Many settings.",
            inherit = "setupitemgroup", -- inherit "settings"
                                        -- from setupitemgroup
```

Examples

```
                type = "settings",
                settings = {    -- just to show, inherit + further settings
                    dummy = {
                        values = {"foo", "bar"},
                        comment =
                            "Setting that is not inherited from setupitemgroup.",
                    },
                },
            },
        },
        variants = {
            {
                comment = "Use default setup.",
            },
            {
                comment = "Use some keywords.",
                "keywords",
            },
            {
```

Examples

```
            comment = "Use some settings.",
            "settings",
        },
        {
            comment = "Use keywords and settings.",
            "keywords", "settings",
        },
    },
}

t.arguments.keywords.keywords = {
    a = "Alphabetic numerating.",
    A = "Uppercase alphabetic numerating.",
    KA = "todo",
    n = "todo",
}

return t
```

general-keywords

```
return {
    nowhite = [[Don't apply whitespace that has been setup with
            \cmd{setupwhitespace}]],
}
```

general-settings

```
return {
    align = {
        comment = "How to align something.",
        values = {"no", "flushleft", "flushright", "middle", "normal",
                  "high", "low", "lohi"},
    },
    location = {
        comment = "The location of an element.",
        description = "More text to explain the details.",
        values = {"depth", "hanging", "high", "lohi", "low", "top",
                  "middle", "bottom"},
    },
    strut = {
        comment = "Management of a strut.",
        description = "More text to explain the details.",
        values = {"yes", "no", "global", "local"},
```

Examples

```
        },
    }
```

Perspectives for the future

When this syntax is generally accepted, we can:

- put these files on a public svn-server
- integrate the content of the `cont-en.xml` file and the wiki
- add a LuaTEX script that generates a nice looking PDF (once per night for example)
- and then add more and more content

We hope, that this command description syntax will make it a pleasure to add explanations of macros and their arguments.

Thanks for your attention!
Taco and Peter