



Introduction



Overview



Demo



Challenges



Conclusion

Quickref: a Stress Test for Texinfo

~ TUG 2019 ~

Didier Verna
EPITA / LRDE

didier@lrde.epita.fr



[lrde/~didier](#)



[@didierverna](#)



[didier.verna](#)



[in/didierverna](#)

Introduction

- ▶ Common Lisp: Social / Community Aspects
 - ▶ The most expressive and extensible language (homoiconicity)
 - ▶ Drawbacks: ~~technical~~ social challenges
 - ▶ Individualism
 - ▶ (Too) Many different solutions for every problem
 - ▶ Quality difficult to assert
 - ▶ Many of them ad-hoc or 80% finished
 - ▶ Lack of documentation
- ▶ Consolidation Efforts
 - ▶ Websites, Resources (guides, tutorials, wikis etc.)
 - ▶ ASDF, Quicklisp
- ▶ Introducing Quickref
 - ▶ Global automatic documentation project for Quicklisp libraries
 - ▶ `<don>` Reference manuals \neq user manuals `</don>`

 **Outline**

System Overview

Demonstration

Challenges

Conclusion & Perspectives

   Outline 

System Overview

Demonstration

Challenges

Conclusion & Perspectives

 **Features** 

- ▶ 2000 or so libraries
- ▶ Public website: `quickref.common-lisp.net`
- ▶ Personal copy: Docker image / Lisp REPL
- ▶ Private website: local installation only

Documentation Extraction

- ▶ Distribution (README files *etc.*)
- ▶ ASDF metadata (author, description, repository, *etc.*)
- ▶ Language-level documentation (docstrings)
- ▶ The rest (software components)
 - ▶ Code Walking (~~lightweight but very difficult~~)
 - ▶ Introspection (easier but requires loading)
system components, packages, constants, variables, macros, functions, methods, structures, classes, types, etc.

Documentation Extraction

- ▶ Distribution (README files *etc.*)
- ▶ ASDF metadata (author, description, repository, *etc.*)
- ▶ Language-level documentation (docstrings)

▶ ASDF metadata

```
(asdf:defsystem :net.didierverna.declt
  :long-name "Documentation Extractor from Common Lisp to Texinfo"
  :description "A reference manual generator for Common Lisp libraries"
  :author "Didier Verna"
  :mailto "didier@didierverna.net"
  :homepage "http://www.lrde.epita.fr/~didier/software/lisp/"
  :source-control "https://github.com/didierverna/declt"
  :license "BSD"
  ...)
```

Documentation Extraction

- ▶ Distribution (README files *etc.*)
- ▶ ASDF metadata (author, description, repository, *etc.*)
- ▶ Language-level documentation (docstrings)
- ▶ The rest (software components)

Documentation strings

```
(defmacro @defconstant (name &body body)
  "Execute BODY within a @defvr {Constant} NAME environment.
  NAME is escaped for Texinfo prior to rendering.
  BODY should render on *standard-output*."
  `(@defvr "Constant" ,name ,@body))
```


Toolchain



▶ Why Texinfo?

- ▶ Well suited to technical documentation (reference manual)
- ▶ Easily converted (PDF, HTML, Info, etc.)
- ▶ Built-in index / anchoring / cross-reference facility

▶ Declt: Introspection

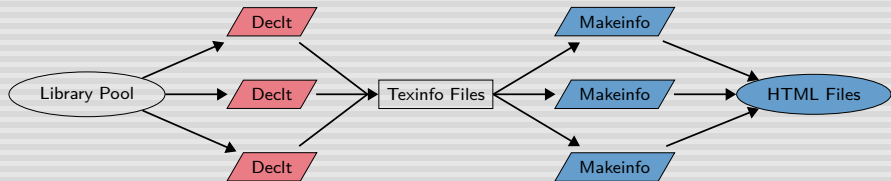
- ▶ Compilation / loading (of dependencies) may be required
- ▶ Avoid loading 2000 libraries in the same Lisp image!
- ▶ Run Declt in external processes

▶ Makeinfo: Perl/C script

- ▶ *Ditto*

▶ Quickref: Additional glue + loop over all Quicklisp libraries

Performance



- ▶ Sequential Processing
 - ▶ Absolute worst-case sequential scenario: 7h
 - ▶ Typical scenario: 2h
- ▶ Parallel Processing + scheduling algorithm
 - ▶ 4x performance improvement



Outline

System Overview

Demonstration

Challenges

Conclusion & Perspectives

English Conjugation Point

demo break
≠
demo breaks



Outline

System Overview

Demonstration

Challenges

Conclusion & Perspectives

A Stress Test for Texinfo

- ▶ Scalability
 - ▶ 2000 or so libraries
 - ▶ Dependency management
 - ▶ Foreign dependencies
 - ▶ Library / documentation size
- ▶ Texinfo Figures
 - ▶ File sizes: 7Ko – 15Mo (x2 HTML)
 - ▶ Lines of code: 364 – 285,020
 - ▶ Index entries: 14 – 44,500
 - ▶ Processing time: 0.3s – 1, 38s

Metadata Format Underspecification

```
:author "Didier Verna"  
:author "Didier Verna <didier@lrde.epita.fr>"  
:author "Didier Verna didier@lrde.epita.fr"  
:author "didier@lrde.epita.fr"  
:author "<didier@lrde.epita.fr>"  
:author "Didier Verna and Antoine Martin"  
:author "Didier Verna, Antoine Martin"  
:author "Didier Verna Antoine Martin"  
:author "D. Verna Antoine E Martin"  
:author "D. Verna Antoine \"Joe Cool\" Martin"  
:author ("Didier Verna" "Antoine Martin")
```

Metadata Format Underspecification

```
:author"
```

```
Original Authors:
```

```
Salvi Péter,  
Naganuma Shigeta,  
Tada Masashi,  
Abe Yusuke,  
Jianshi Huang,  
Fujii Ryo,  
Abe Seika,  
Kuroda Hisao
```

```
Author Post MSI CLML Contribution:
```

```
Mike Maul <maul.mike@gmail.com>"
```


Metadata Format Underspecification

```
:author "(let ((n \"Christoph-Simon Senjak\")) ~  
          (format nil \"~A <~C~C~C~C~A>\" ~  
                  n (elt n 0) (elt n 10) (elt n 16) ~  
                    #\\@ \"uxul.de\")))"
```

- ▶ Social incentive: people don't like *their work* to look bad on *my* public website...

Definitions Grouping

Example 1: accessors

`context-hyperlinksp` *CONTEXT* [Function]

`(setf context-hyperlinksp)` *BOOL CONTEXT* [Function]

Access *CONTEXT*'s `hyperlinksp` flag.

Package [net.didierverna.declt], page 29,

Source [doc.lisp], page 24, (file)

Definitions Grouping

Example 2: generic functions

document *ITEM* *CONTEXT*

[Generic Function]

Render *ITEM*'s documentation in *CONTEXT*.

Package [net.didierverna.declt], page 29,

Source [doc.lisp], page 24, (file)

Methods

document (*SYSTEM* system) *CONTEXT*

Render *SYSTEM*'s documentation in *CONTEXT*.

Source [asdf.lisp], page 26, (file)

document (*MODULE* module) *CONTEXT*

Render *MODULE*'s documentation in *CONTEXT*.

Source [asdf.lisp], page 26, (file)

Definitions Grouping

- ▶ Only use the low level interface: `@defn`, `@defvr`, *etc.*

- ▶ Environment nesting → unwanted indentation
- ▶ Heterogeneous `@def...` / `@def...x` prohibited
- ▶ Heterogeneous categories authorized

```
@defn {Function} ...
```

```
@defnx {Compiler Macro} ...
```

- ▶ Remaining Limitations

- ▶ Only 9 fixed canonical categories
- ▶ Some distinctions arguable (*e.g.* typed vs. untyped)
- ▶ Heterogeneous mixing still prohibited

```
@defn {Function} foo ...
```

```
@defvr {Symbol Macro} foo ...
```

Pretty Printing

- ▶ Names can be anything → escaping vs. “revealing”
 - ▶ `|my stuff|` vs. `my_stuff`
 - ▶ `(setf foo)` vs. `(setf foo)`
 - ▶ `|argument(s)|` vs. `argument(s)`
- ▶ Symbol qualification: `my.long.package.name: symbol`
 - ▶ In general: avoid
 - ▶ Sometimes leads to ambiguous output (e.g. method specializers)
- ▶ Docstrings: what to do? Verbatim, simple heuristic(s), markup *etc.*
- ▶ References: `@ref{anchor, , label}` gives varying output

HTML label


PDF [label], page 12, → *trailing comma (or not)*

Info *note label: anchor.

Emacs See label. → *Casing seems to vary*

Anchoring

- ▶ Anchor names limitations (dots, commas, colons, parens)
 - ▶ Use <dot> etc. (ugly; use UTF8 characters instead?)
 - ▶ Anchor text less constrained, not well documented
- ▶ Avoid nodes as much as possible...
 - ▶ Problems above
 - ▶ Uniqueness of names
 - ▶ No control over the display
- ▶ ...in particular, nodes associated with Lisp symbols



Outline



System Overview

Demonstration

Challenges

Conclusion & Perspectives

Conclusion

- ▶ A successful project
 - ▶ Almost 2000 libraries nicely documented
 - ▶ Less than 2% still cause problems
 - ▶ The community is grateful
- ▶ A successful stress test for Texinfo
 - ▶ Reliable and scalable
 - ▶ Sometimes gets in the way, but still a good choice

Perspectives

- ▶ Casing
- ▶ More / different indexes
- ▶ More links / cross-references (external notably)
- ▶ More / improved pretty-printing
- ▶ Provide PDF & Info on the website as well
- ▶ Emacs / Slime integration
- ▶ More index (web) pages
- ▶ ...