Common Lisp UltraSpec – A Project For Modern Common Lisp Documentation

Michał „phoe” Herda
#lisp-pl @ Freenode
Jagiellonian University, Cracow, Poland
Previously on European Lisp Symposium 2016
Yet Another Rant About The State Of Common Lisp Documentation

Michał „phoe” Herda
Presenting the European Lisp Symposium 2017
Exclusive:
Yet Another Rant About The State Of Common Lisp Documentation, Part 2

Michał „phoe” Herda
Welcome to the Common Lisp HyperSpec.  
I hope it serves your need.

--Kent Pitman, X3J13 Project Editor

Here are some useful starting points:

A text-only version of this cover sheet is available.

Copyright 1996-2005, LispWorks Ltd. All Rights Reserved.
Aww... not going to set the last six to NIL?
The arrays will probably be numbers or characters.
Hmm... Is printing of arrays implementation-specific? I'm a bit disappointed that SBCL doesn't print it like
```
#2A("@@@@@@@@
```
when it's character.

A formatting is part of the spec, so it'd be a bit breaky
to extend that syntax.

ah, that's unfortunate.
of course, print-object is extensible.
I'm using 2D array rows like database table columns.
It's funny/sad when I get a type error, and it spams 1500 or so lines, whatever the limit is.
And I never find out what type it was expecting!

could someone help me improve/optimize http://ideone.com/0i68bt?
Currently this can be at least 20 times faster comparing to the
best solution (problem defined here:
http://www.spoj.com/problems/PALIN/)

clhs adjoin


phoe(ri) 2:freenode/#lisp(Ccnz) [Act: 4,5,6,8,9,10]

"origin" 08:25 03-kwi-17
CLHS: Function MAKE-ARRAY - LispWorks
www.lispworks.com/.../lw51/CLHS/.../f_mk_ar.htm ▼ Tłumaczenie strony
Syntax: make-array dimensions &key element-type initial-element initial-contents adjustable fill-pointer displaced-to displaced-index-offset. => new-array.

CLHS: Section The Arrays Dictionary - LispWorks
www.lispworks.com/documentation/.../c_arrays.htm ▼ Tłumaczenie strony
15.2 The Arrays Dictionary. System Class ARRAY · Type SIMPLE-ARRAY · System Class VECTOR · Type SIMPLE-VECTOR · System Class BIT-VECTOR.

CLHS: System Class ARRAY
c lhs.lisp.se/Body/t_array.htm ▼ Tłumaczenie strony
An array contains objects arranged according to a Cartesian coordinate system. An array provides mappings from a set of fixnums {i0,i1,...,ir-1} to corresponding ...
Ta strona była przez Ciebie odwiedzana.

CLHS: Function ADJUST-ARRAY
c lhs.lisp.se/Body/f_adjust.htm ▼ Tłumaczenie strony
Syntax: adjust-array array new-dimensions &key element-type initial-element initial-contents fill-pointer displaced-to displaced-index-offset. => adjusted-array.
About the Franz online ANS for Common Lisp

Purpose

The ANSI standard for Common Lisp is very large. Its introduction states it “is a language specification aimed at an audience of implementors and knowledgeable programmers. It is neither a tutorial nor an implementation guide.” While the document is considered an excellent language standard, its size (about 1400 pages) and organization place formidable difficulties when researching specific language questions. This search page is a tool that can reliably find all the scattered references to a particular element of the language.

Sources

Starting in 1986 the X3J13 technical subcommittee of X3 (since renamed NCITS) drafted a standard for Common Lisp. It was officially adopted by ANSI in 1994 as ANSI X3.226-1994. That document is the definitive standard for Common Lisp and can be ordered from ANSI in paper form ($350).

The ANS is a very large standard, about 1400 pages in printed form. As the technical deliberations of X3J13 were nearing completion, it became apparent to the committee that the editorial task to draft the standard would not be feasible using part-time volunteer effort. An informal industry consortium was formed through which several vendor organizations funded a full-time editor for about a year and a half. The consortium stipulated that drafts produced by that editor were to be made available to X3J13 to submit to ANSI, and also to members of the consortium and any other interested parties.

Three draft proposed American National Standards (dpANS) were subsequently produced. dpANS 1 was a working draft and received extensive review and revision by X3J13. dpANS 2 was then created to resolve all remaining technical issues; X3J13 made no intentional changes in technical content after dpANS 2. dpANS 3 improved wording in a few places and extensively reworked the Credits section, not a normative part of the standard. Further formatting changes and boilerplate additions were executed by ANSI in their process of.
adjoin site:franz.com

adjoin - Franz Inc.
franz.com/support/documentation/8.1/ansicl/dictentr/adjoin.htm ▼ Tłumaczenie strony
21 lip 2009 - If the item is not an existing element, adjoin adds it to list (as if by cons) and returns the resulting list; otherwise, nothing is added and the ...

pushnew - Franz Inc.
franz.com/support/documentation/8.2/ansicl/.../pushnew.htm ▼ Tłumaczenie strony
If :key is supplied, it is used to extract the part to be tested from both item and the list element, as for adjoin. The argument to the :key function is an element of the ...

Deprecated Argument Conventions - Franz Inc.
ftp.franz.com/support/documentation/8.1/ansicl/.../depreca1.htm ▼ Tłumaczenie strony
21 lip 2009 - adjoin, nset-difference, search. assoc, nset-exclusive-or, set-difference. count, nsblis, set-exclusive-or. delete, nsbst, subsli. delete-duplicates ...

Dictionary of Conses - Franz Inc.
franz.com/support/documentation/6.0/ansicl/.../dictio10.htm ▼ Tłumaczenie strony
14 2 44 adjoin 14 2 45 pushnew 14 2 46 set-difference nset-difference 14 2 47 set-exclusive-or nset-
L1sp.org

L1sp.org is a redirect service for Common Lisp documentation. For example, try these links:

- http://l1sp.org/ccl/@selector
- http://l1sp.org/cl/11.1.2.1.2
- http://l1sp.org/cl/defpackage
- http://l1sp.org/cl/glossary/binding
- http://l1sp.org/cl/~/
- http://l1sp.org/clim/present
- http://l1sp.org/clisp/ext:run-program
- http://l1sp.org/clx/open-display
- http://l1sp.org/mop/compute-slots
- http://l1sp.org/pcl/loop

You can also search for symbols:

adjoin Search

Symbol redirects are available for alexandria, asdf, ccl, cffi, cl, cl-fad, cl-gd, clim, clisp, cl-ppcre, cl-unicode, cl-webdav, cl-who, clx, documentation-template, drakma, flexi-streams, hunchentoot, lisp, mop, pcl, and sbcl.

Questions or comments? Email me.
http://www.lispworks.com/documentation/HyperSpec/...
http://www.sbcl.org/manual/...
http://ccl.clozure.com/manual/...
http://www.clisp.org/...
http://bauhh.dyndns.org:8000/clim-spec/...
http://bauhh.de/clxman/...
http://metamodular.com/CLOS-MOP/...

50+ (?) more websites with library-specific docs
Quicklisp beta

Quicklisp is a library manager for Common Lisp. It works with your existing Common Lisp implementation to download, install, and load any of over 1,400 libraries with a few simple commands.

Quicklisp is easy to install and works with ABCL, Allegro CL, Clasp, Clozure CL, CLISP, CMUCL, ECL, LispWorks, MKCL, SBCL, and Scieneer CL, on Linux, Mac OS X, and Windows. The libraries were last updated on February 27, 2017.

To get started with the Quicklisp beta, download and load https://beta.quicklisp.org/quicklisp.lisp

PGP signature of quicklisp.lisp
sha256 of quicklisp.lisp =
4a7a5c2aebe0716417047854267397e24a44d0cce096127411e9ce9ccfeb2c17

Quicklisp is provided as-is without warranty of any kind. See the release notes for known problems.
Library Documentation Hosting for Common Lisp

Search by Category

- Web development
- Graphics
- GUI
- System & Low-level
- Testing framework
- Database
- Utility Collection
- Concurrency
Alexandria is a collection of portable public domain utilities that meet the following constraints:

* Utilities, not extensions: Alexandria will not contain conceptual extensions to Common Lisp, instead limiting itself to tools and utilities that fit well within the framework of standard ANSI Common Lisp. Test-frameworks, system definitions, logging facilities, serialization layers, etc. are all outside the scope of Alexandria as a library, though well within the scope of Alexandria as a project.

* Conservative: Alexandria limits itself to what project members consider conservative utilities. Alexandria does not and will not include anaphoric constructs, loop-like binding macros, etc.

* Portable: Alexandria limits itself to portable parts of Common Lisp. Even apparently conservative and useful functions remain outside the scope of Alexandria if they cannot be implemented portably. Portability is here defined as portable within a conforming implementation: implementation bugs are not considered portability issues.
Function  LASTCAR (list)
Returns the last element of LIST. Signals a type-error if LIST is not a proper list.

Function  (setf LASTCAR) (object list)
Sets the last element of LIST. Signals a type-error if LIST is not a proper list.

Function  MAKE-CIRCULAR-LIST
(length &key initial-element)
Creates a circular list of LENGTH with the given INITIAL-ELEMENT.

Type  CIRCULAR-LIST
Type designator for circular lists. Implemented as a SATISFIES type, so not recommended for performance intensive use. Main usefulness as the expected-type designator of a TYPE-ERROR.

Function  ENSURE-CAR (thing)
If THING is a CONS, its CAR is returned. Otherwise THING is returned.

Function  ENSURE-CONS (cons)
If CONS is a cons, it is returned. Otherwise returns a fresh cons with CONS in the car, and NIL in the cdr.

Function  ENSURE-LIST (list)
If LIST is a list, it is returned. Otherwise returns the list designated by LIST.

Function  REMOVE-FROM-PLIST (plist &rest keys)
Returns a property-list with same keys and values as PLIST, except that keys in the list designated by KEYS and values corresponding to them are removed. The returned property-list may share structure with the PLIST, but PLIST is not destructively modified. Keys are compared using EQ.
Authorship Information

The Common Lisp HyperSpec is *not* the ANSI Common Lisp standard, but is based on that standard (with permission from ANSI and X3).
Programming Language—Common Lisp

Content

Chapter 1 (Introduction)
1.1 Scope, Purpose, and History
1.2 History
1.3 Organization of the Document
1.4 Definitions
1.4.1 Notational Conventions
1.4.2 Font Key
1.4.3 Modified BNF Syntax
1.4.4 Additional Uses for Indirect Definitions in Modified BNF Syntax
1.4.5 Special Symbols
1.4.6 Objects with Multiple Notations
1.4.7 Error Handling
1.4.8 Numbers
1.4.9 Use of the Dot Character
1.4.10 NIL
1.4.11 Designators
1.4.12 Uncommon Words
1.4.13 Sections Not Formally Part Of This Standard
1.4.14 Interpreting Dictionary Entries
1.4.15 The "Affect 10 By" Section of a Dictionary Entry
1.4.16 The "Arguments" Section of a Dictionary Entry
1.4.17 The "Arguments and Values" Section of a Dictionary Entry
1.4.18 The "Binding Types Affected" Section of a Dictionary Entry
1.4.19 The "Class Precedence List" Section of a Dictionary Entry
1.4.20 Dictionary Entries for Type Specifiers
1.4.21 The "Compound Type Specifier Kind" Section of a Dictionary Entry
1.4.22 The "Compound Type Specifier Syntax" Section of a Dictionary Entry
1.4.23 The "Compound Type Specifier Arguments" Section of a Dictionary Entry
1.4.24 The "Compound Type Specifier Description" Section of a Dictionary Entry
1.4.25 The "Constant Value" Section of a Dictionary Entry
1.4.26 The "Description" Section of a Dictionary Entry
1.4.27 The "Examples" Section of a Dictionary Entry
1.4.28 The "Exceptional Situations" Section of a Dictionary Entry
1.4.29 The "Initial Value" Section of a Dictionary Entry
1.4.30 The "Argument Precedence Order" Section of a Dictionary Entry
1.4.31 The "Method Signature" Section of a Dictionary Entry
1.4.32 The "Name" Section of a Dictionary Entry
1.4.33 The "Notes" Section of a Dictionary Entry
1.4.34 The "Pronunciation" Section of a Dictionary Entry
1.4.35 The "Side Effects" Section of a Dictionary Entry
1.4.36 The "Super TYPES" Section of a Dictionary Entry

Index
1. this copyright notice and its date;
2. the main index page, ../Front/index.htm;
3. all HTML pages to which the main index page links using relative links;
4. all graphical (GIF) images to which it links using relative links, such as the LispWorks logo that appears on each page; and
5. all hypertext links, relative or absolute, such as the link to http://www.lispworks.com/ that appears on each page.

Permissions related to performance and to creation of derivative works are expressly NOT granted.

Permission to make partial copies is expressly NOT granted, EXCEPT that limited permission is granted to transmit and display a partial copy the Common Lisp HyperSpec for the ordinary purpose of direct viewing by a human being in the usual manner that hypertext browsers permit the viewing of such a complete document, provided that no recopying, redistribution, redisplay, or retransmission is made of any such partial copy.

Permission to make modified copies is expressly NOT granted.

Permission to add or replace any links or any graphical images to any of these pages is expressly NOT granted.

Permission to use any of the included graphical (GIF) images in any document other than the Common Lisp HyperSpec is expressly NOT granted.
Fin, Part 2
(...except it’s not)
About the Franz online ANS for Common Lisp

Purpose

The ANSI standard for Common Lisp is very large. Its introduction states it "is a language specification aimed at an audience of implementors and knowledgeable programmers. It is neither a tutorial nor an implementation guide." While the document is considered an excellent language standard, its size (about 1400 pages) and organization place formidable difficulties when researching specific language questions. This search page is a tool that can reliably find all the scattered references to a particular element of the language.

Sources

Starting in 1986 the X3J13 technical subcommittee of X3 (since renamed NCITS) drafted a standard for Common Lisp. It was officially adopted by ANSI in 1984 as ANSI X3.226-1984. That document is the definitive standard for Common Lisp and can be ordered from ANSI in paper form ($350).

The ANSI standard is a very large standard, about 1400 pages in printed form. As the technical deliberations of X3J13 were nearing completion, it became apparent to the committee that the editorial task to draft the standard would not be feasible using part-time volunteer effort. An informal industry consortium was formed through which several vendor organizations funded a full-time editor for about a year and a half. The consortium stipulated that drafts produced by that editor were to be made available to X3J13 to submit to ANSI, and also to members of the consortium and any other interested parties.

Three draft proposed American National Standards (dpANS) were subsequently produced. dpANS 1 was a working draft and received extensive review and revision by X3J13. dpANS 2 was then created to resolve all remaining technical issues. X3J13 made no intentional changes in technical content after dpANS 2. dpANS 3 improved wording in a few places and extensively reworked the Credits section, not a normative part of the standard. Further formatting changes and boilerplate additions were executed by ANSI in their process of producing the standard, but these have no technical implications.

The document in these pages is derived from dpANS2. It is a semi-mechanical translation of the original TeX into HTML. While ANSI X3.226-1984 is the definitive, official standard, this HTML version of dpANS2 is believed to be equivalent in all content of technical consequence.

The presentation of the language standard on these pages copyright © Franz Inc.

Comments are welcome: webmaster@franz.com
The document in these pages is derived from dpANS2. It is a semi-mechanical translation of the original TeX into HTML. While ANSI X3.226-1994 is the definitive, official standard, this HTML version of dpANS2 is believed to be equivalent in all content of technical consequence.
A Possible Solution To The State Of Common Lisp Documentation, Part 2

Common Lisp UltraSpec

Michał „phoe” Herda
\begin{itemize}
  \item MAPCAR
  \item MAPLIST
  \item MAPC
  \item MAPL
  \item MAPCAN
  \item MAPCON
\end{itemize}

\texttt{begincom\{mapc, mapcar, mapcan, mapl, maplist, mapcon\}\texttt{ftype\{Function\}}}

\texttt{label Syntax::}

\texttt{DefunWithValues mapc \{function \{\texttt{rest} \plus\{lists\}\}\} \{list-1\}}
\texttt{DefunWithValues mapcar \{function \{\texttt{rest} \plus\{lists\}\}\} \{result-list\}}
\texttt{DefunWithValues mapcan \{function \{\texttt{rest} \plus\{lists\}\}\} \{concatenated-results\}}
\texttt{DefunWithValues mapl \{function \{\texttt{rest} \plus\{lists\}\}\} \{list-1\}}
\texttt{DefunWithValues maplist \{function \{\texttt{rest} \plus\{lists\}\}\} \{result-list\}}
\texttt{DefunWithValues mapcon \{function \{\texttt{rest} \plus\{lists\}\}\} \{concatenated-results\}}

\texttt{label Arguments and Values::}

\texttt{param\{function\}---a \texttt{term\{designator\}} for a \texttt{term\{function\}} that must take as many \texttt{term\{arguments\}} as there are \texttt{param\{lists\}}.}

\texttt{issue\{DOTTED-LIST-ARGUMENTS:CLARIFY\}}
\texttt{param\{list\}---a \texttt{term\{proper list\}}.}
\texttt{param\{list-1\}---the first \texttt{param\{list\}} (which must be a \texttt{term\{proper list\}}).}
\texttt{endissue\{DOTTED-LIST-ARGUMENTS:CLARIFY\}}
Function MAPC, MAPCAR, MAPCAN, MAPL, MAPLIST, MAPCON

The mapping operation involves applying //function// to successive sets of arguments in which one argument is obtained from each //[[CL:Glossary:sequence]]//. Except for **mapc** and **mapl**, the result contains the results returned by //function//. In the cases of **mapc** and **mapl**, the resulting //[[CL:Glossary:sequence]]// is //list//.

//function// is called first on all the elements with index '0', then on all those with index '1', and so on. //result-type// specifies the //[[CL:Glossary:type]]// of the resulting //[[CL:Glossary:sequence]]//. If //function// is a //[[CL:Glossary:symbol]]//, it is **[[CL:Functions:coerce]]**d to a //[[CL:Glossary:function]]// as if by **[[CL:Functions:symbol-function]]**.

**mapcar** operates on successive //[[CL:Glossary:element|elements]]// of the //lists//. //function// is applied to the first //[[CL:Glossary:element]]// of each //list//, then to the second //[[CL:Glossary:element]]// of each //list//, and so on. The iteration terminates when the shortest //list// runs out, and excess elements in other lists are ignored. The value returned by **mapcar** is a //[[CL:Glossary:list]]// of the results of successive calls to //function//.

**mapc** is like **mapcar** except that the results of applying //function// are not accumulated. The //list// argument is returned.

**maplist** is like **mapcar** except that //function// is applied to successive sublists of the //lists//. //function// is first applied to the //lists// themselves, and then to the //[[CL:Glossary:cdr]]// of each //list//, and then to the //[[CL:Glossary:cdr]]// of the //[[CL:Glossary:cdr]]// of each //list//, and so on.

**mapl** is like **maplist** except that the results of applying //function// are not accumulated: //list-1// is returned.
Function MAPC, MAPCAR, MAPCAN, MAPL, MAPLIST, MAPCON

Syntax

- `mapc function &rest lists+ → list-1`
- `mapcar function &rest lists+ → result-list`
- `mapcan function &rest lists+ → concatenated-results`
- `mapl function &rest lists+ → list-1`
- `maplist function &rest lists+ → result-list`
- `mapcon function &rest lists+ → concatenated-results`

Arguments and Values

- `function` - a designator for a function that must take as many arguments as there are lists.
- `list` - a proper list.
- `list-1` - the first list (which must be a proper list).
- `result-list` - a list.
- `concatenated-results` - a list.

Description

The mapping operation involves applying `function` to successive sets of arguments in which one argument is obtained from each sequence. Except for `mapc` and `mapl`, the result contains the results returned by `function`. In the cases of `mapc` and `mapl`, the resulting sequence is `list`.

`function` is called first on all the elements with index 0, then on all those with index 1, and so on. `result-type` specifies the type of the resulting sequence. If `function` is a symbol, it is coerced to a function as if by `symbol-function`.

`mapcar` operates on successive elements of the lists. `function` is applied to the first element of each list, then to the second element of each list, and so on. The iteration terminates when the shortest list runs out, and excess elements in other lists are ignored. The value returned by `mapcar` is a list of the results of successive calls to `function`.

`mapc` is like `mapcar` except that the results of applying `function` are not accumulated. The list argument is returned.
there's some issue in chapters Reader or Printer: I forgot to type ", respectively" at the end of some \Defaults
* replace all star-foo-star variables with just foo
* check all dictionaries if not a single symbol was omitted
* add dictionary entries for lambda list keywords (if they're not there already)

\input setup
\begin{chapter}{Introduction}{ChapOne}{Introduction}

\chapter{Scope, Purpose, and History}
\begin{subsection}{Scope and Purpose}
The specification set forth in this document is designed to promote the portability of Common Lisp programs among a variety of data processing systems. It is a language specification aimed at an audience of implementors and knowledgeable programmers. It is neither a tutorial nor an implementation guide.

\end{subsection}

\begin{subsection}{History}
Lisp is a family of languages with a long history. John McCarthy during the 1956 Dartmouth Summer provided McCarthy's motivation was to develop an algebraic system of artificial intelligence work. Implementation efforts for Lisp 704, the IBM 7090, the Digital Equipment Corporation VSP-10. The primary dialect of Lisp between 1960 and 1970 were two predominant dialects of Lisp, both are now known as Interlisp. For further information about very early Lisp, see [2].

\end{subsection}

MacLisp improved on the Lisp 1.5 notion of special variables, introduced the concept of functions that could take a variable number of arguments, macros,
Parsing dpANS with RegEx
You can't parse [X]HTML with regex. Because HTML can't be parsed by regex. Regex is not a tool that can be used to correctly parse HTML. As I have answered in HTML-and-regex questions here so many times before, the use of regex will not allow you to consume HTML. Regular expressions are a tool that is insufficiently sophisticated to understand the constructs employed by HTML. HTML is not a regular language and hence cannot be parsed by regular expressions. Regex queries are not equipped to break down HTML into its meaningful parts. so many times but it is not getting to me. Even enhanced irregular regular expressions as used by Perl are not up to the task of parsing HTML. You will never make me crack. HTML is a language of sufficient complexity that it cannot be parsed by regular expressions. Even Jon Skeet cannot parse HTML using regular expressions. Every time you attempt to parse HTML with regular expressions, the unholy child weeps the blood of virgins, and Russian hackers pwn your webapp. Parsing HTML with regex summons tainted souls into the realm of the living. HTML and regex go together like love, marriage, and ritual infanticide. The <center> cannot hold it is too late. The force of regex and HTML together in the same conceptual space will destroy your mind like so much watery putty. If you parse HTML with regex you are giving in to Them and their blasphemous ways which doom us all to inhuman toil for the One whose Name cannot be expressed in the Basic Multilingual Plane, he comes. HTML-plus-regexp will liquify the nerves of the sentient whilst you observe, your psyche withering in the onslaught of horror. Regex-based HTML parsers are the cancer that is killing StackOverflow it is too late it is too late we cannot be saved the trangression of a child ensures regex will consume all living tissue (except for HTML which it cannot, as previously prophesied) dear lord help us how can anyone survive this scourage using regex to parse HTML has doomed humanity to an eternity of dread torture and security holes using regex as a tool to process HTML establishes a breach between this world and the dread realm of corrupt entities (like SGML entities, but more corrupt) a mere glimpse of the world of regex parsers for HTML will instantly transport a programmer's consciousness into a world of ceaseless screaming, he comes, the pestilent slithy regex-infection will devour your HTML parser, application and existence for all time like Visual Basic only worse he comes he comes do not fight he comes, his unholy radiance destroying all enlightenment, HTML tags leaking from your eyes like liquid pain, the song of regular expression parsing will extinguish the voices of mortal man from the sphere I can see it can you see it is beautiful the final snuffing of the lies of Man ALL IS LOST ALL IS LOST the pony he comes he comes he comes the ichor permeates all MY FACE MY FACE oh god no NO NOOOO NO stop the angels are not real IALGO IS TONY THE PONY HE COMES

Have you tried using an XML parser instead?
Parsing TeX (context-sensitive grammar) with RegEx (regular grammar):

You're gonna have a bad time
Parsing a known subset of TeX used in dpANS with RegEx (regular grammar):

EVERYTHING WENT BETTER THAN EXPECTED

(with proper care)
What’s done so far?

- Done
  - All Dictionaries (e.g. MAPCAR, ADJOIN, …)
  - Glossary
- Not done / TODO
  - Concepts (e.g. 2.2 Reader Algorithm)
  - Linking (esp. glossary entries)
What’s done so far?

- Done
  - All Dictionaries (e.g. MAPCAR, ADJOIN, …)
  - Glossary
- Not done / TODO
  - Concepts (e.g. 2.2 Reader Algorithm)
  - Linking (esp. glossary entries)
  - Review
  - Diffs to the original dpANS
- Once it’s finished, SHIP IT!

> Why hasn’t anyone made a more eye-frendly version of the Common Lisp Hyper Spec? Having good, easily-browsable documentation is a core-problem.

A friend of mine is working on that as we speak. The CLUS, or Common Lisp UltraSpec:

https://phoe.tymoon.eu/clus/doku.php
What am I aiming for?

- Editable
- Complete
- Downloadable
- Mirrorable/Clonable
- Versioned
- Modular
- Updatable
- Portable
- Unified
- Community-based
„Burning out”

May 15, 2016 – Apr 3, 2017
Contributions to master, excluding merge commits

ELS ‘16
Laziness ↑
...or is it?

WORK

ELS ‘17
Burnout ↑
...or is it?

...I think ELS is coming
Thoughts

- A translator, or an editor?
- A scribe, or a programmer?
- dpANS as humanistic material
- If I am a „new” editor of the specification, then who is the „new” X3J13?
Summing it up:

- Parse the dpANS specification
- Edit it where necessary
- Make it further parsable
- Give it to the community
- Let the community turn it into X
  - Emacs-readable version
  - Web version
  - Printable version
  - S-expression version
  - Community wiki
  - Repository for documents from the whole CL universe
  - Common Lisp the Language 3
Live demo/manifesto:

http://phoe.tymoon.eu/clus/

Thanks to Shinmera for the hosting!
The Actual End, Part 2
(...except it’s not)

- because CLUS is not yet finished
- because I will be looking for contributors
- because it’s questions time

Thanks for listening!