

Doconce Quick Reference

Hans Petter Langtangen^{1,2}

¹Center for Biomedical Computing, Simula Research Laboratory

²Department of Informatics, University of Oslo

Apr 28, 2013

Contents

0.1	Supported Formats	1
0.2	Emacs syntax support	2
0.3	Title, Authors, and Date	2
0.4	Section Types	3
0.5	Inline Formatting	3
0.6	Lists	3
0.7	Comment lines	5
0.8	Inline comments	5
0.9	Verbatim/Computer Code	5
0.10	\LaTeX Mathematics	7
0.11	Figures and Movies	8
0.12	Tables	9
0.13	Labels and References	10
0.14	Citations and Bibliography	10
0.15	Generalized Citations	10
0.16	Index of Keywords	11
0.17	Capabilities of The Program doconce	11
0.18	Exercises	13
0.19	Environments	15
0.20	Preprocessing	15
0.21	Resources	16

WARNING: This quick reference is very incomplete!

Mission. Enable writing documentation with much mathematics and computer code *once, in one place* and include it in traditional \LaTeX books, thesis, and reports, and without extra efforts also make professionally looking web versions with Sphinx or HTML. Other outlets include Google's `blogspot.com`, Wikipedia/Wikibooks, IPython notebooks, plus a wide variety of formats for documents without mathematics and code.

34 **0.1 Supported Formats**

35 Doconce currently translates files to the following formats:

- 36 • \LaTeX (format `latex` and `pdflatex`)
- 37 • HTML (format `html`)
- 38 • reStructuredText (format `rst`)
- 39 • plain (untagged) ASCII (format `plain`)
- 40 • Sphinx (format `sphinx`)
- 41 • IPython notebook (format `ipynb`)
- 42 • MediaWiki (format `mwiki`)
- 43 • (Pandoc extended) Markdown (format `pandoc`)
- 44 • Googlecode wiki (format `gwiki`)
- 45 • Creoloe wiki (format `cwiki`)
- 46 • Epydoc (format `epydoc`)
- 47 • StructuredText (format `st`)

48 For documents with much code and mathematics, the best (and most sup-
49 ported) formats are `latex`, `pdflatex`, `sphinx`, and `html`; and to a slightly less
50 extent `mwiki` and `pandoc`. The HTML format supports blogging on Google and
51 Wordpress.

52 **0.2 Emacs syntax support**

53 The file `.doconce-mode.el` in the Doconce source distribution gives a "Doconce
54 Editing Mode" in Emacs. Store the file in the home directory and add `(load-file "~/doconce-mode.el")`
55 to the `.emacs` file.

56 Besides syntax highlighting of Doconce documents, this Emacs mode pro-
57 vides a lot of shortcuts for setting up many elements in a document:

	Emacs key	Action
	Ctrl+c f	figure
	Ctrl+c v	movie/video
	Ctrl+c h1	heading level 1 (section/h1)
	Ctrl+c h2	heading level 2 (subsection/h2)
	Ctrl+c h3	heading level 2 (subsection/h3)
	Ctrl+c hp	heading for paragraph
	Ctrl+c me	math environment: !bt equation !et
58	Ctrl+c ma	math environment: !bt align !et
	Ctrl+c ce	code environment: !bc !ec
	Ctrl+c cf	code from file: @@@CODE
	Ctrl+c table2	table with 2 columns
	Ctrl+c table3	table with 3 columns
	Ctrl+c table4	table with 4 columns
	Ctrl+c exer	exercise outline
	Ctrl+c slide	slide outline
	Ctrl+c help	print this table

59 0.3 Title, Authors, and Date

60 A typical example of giving a title, a set of authors, a date, and an optional table
61 of contents reads

```
TITLE: On an Ultimate Markup Language
AUTHOR: H. P. Langtangen at Center for Biomedical Computing, Simula Research Laboratory & Dept. of Inf
AUTHOR: Kaare Dump Email: dump@cyb.space.com at Segfault, Cyberspace Inc.
AUTHOR: A. Dummy Author
DATE: today
TOC: on
```

62 The entire title must appear on a single line. The author syntax is

```
name Email: somename@adr.net at institution1 & institution2
```

63 where the email is optional, the "at" keyword is required if one or more insti-
64 tutions are to be specified, and the & keyword separates the institutions (the
65 keyword and works too). Each author specification must appear on a single
66 line. When more than one author belong to the same institution, make sure
67 that the institution is specified in an identical way for each author.

68 The date can be set as any text different from today if not the current date
69 is wanted, e.g., Feb 22, 2016.

70 The table of contents is removed by writing TOC: off.

71 0.4 Section Types

	Section type	Syntax
	chapter	===== Heading ===== (9 =)
	section	===== Heading ===== (7 =)
	subsection	===== Heading ===== (5 =)
	subsubsection	=== Heading === (3 =)
72	paragraph	__Heading.__ (2 _)
	abstract	__Abstract.__ Running text..
	appendix	===== Appendix: heading ===== (7 =)
	appendix	===== Appendix: heading ===== (5 =)
	exercise	===== Exercise: heading ===== (7 =)
	exercise	===== Exercise: heading ===== (5 =)

73 Note that abstracts are recognized by starting with `__Abstract.__` or `__Summary.__`
 74 at the beginning of a line and ending with three or more = signs of the next
 75 heading.

76 The `Exercise:` keyword can be substituted by `Problem:` or `Project:`. A
 77 recommended convention is that an exercise is tied to the text, a problem can
 78 stand on its own, and a project is a comprehensive problem.

79 0.5 Inline Formatting

80 Words surrounded by * are emphasized: `*emphasized words*` becomes *em-*
 81 *phasized words*. Similarly, an underscore surrounds words that appear in bold-
 82 face: `_boldface_` becomes **boldface**. Colored words are also possible: the
 83 text

```
'color{red}{two red words}'
```

84 becomes **two red words**.

85 0.6 Lists

86 There are three types of lists: *bullet lists*, where each item starts with *, *enu-*
 87 *meration lists*, where each item starts with o and gets consqutive numbers,
 88 and *description lists*, where each item starts with – followed by a keyword and
 89 a colon.

Here is a bullet list:

```
* item1
* item2
  * subitem1 of item2
  * subitem2 of item2
* item3
```

Note that sublists are consistently indented by one or more blanks..
 Here is an enumeration list:

```
o item1
o item2
  may appear on
  multiple lines
o subitem1 of item2
o subitem2 of item2
o item3
```

And finally a description list:

- keyword1: followed by
some text
over multiple
lines
- keyword2:
followed by text on the next line
- keyword3: and its description may fit on one line

90 The code above follows.

91 Here is a bullet list:

- 92 • item1
- 93 • item2
 - 94 – subitem1 of item2
 - 95 – subitem2 of item2
- 96 • item3

97 Note that sublists are indented. Here is an enumeration list:

- 98 1. item1
- 99 2. item2 may appear on multiple lines
 - 100 (a) subitem1 of item2
 - 101 (b) subitem2 of item2
- 102 3. item3

103 And finally a description list:

104 **keyword1:** followed by some text over multiple lines

105 **keyword2:** followed by text on the next line

106 **keyword3:** and its description may fit on one line

107 0.7 Comment lines

108 Lines starting with # are treated as comments in the document and translated
109 to the proper syntax for comments in the output document. Such comment
110 lines should not appear before \LaTeX math blocks, verbatim code blocks, or lists
111 if the formats `rst` and `sphinx` are desired.

112 Comment lines starting with ## are not propagated to the output document
113 and can be used for comments that are only interest in the Doconce file.

114 Large portions of text can be left out using Preprocess. Just place # `#ifdef EXTRA`
115 and # `#endif` around the text. The command line option `-DEXTRA` will bring the
116 text alive again.

117 When using the Mako preprocessor one can also place comments in the
118 Doconce source file that will be removed by Mako before Doconce starts pro-
119 cessing the file.

120 0.8 Inline comments

121 Inline comments meant as messages or notes, to authors during development
122 in particular, are enabled by the syntax

```
[name: running text]
```

123 where `name` is the name or ID of an author or reader making the comment, and
124 `running text` is the comment. Here goes an example. The inline comments
125 have simple typesetting in most formats, typically boldface name and every-
126 thing surrounded by parenthesis, but with \LaTeX output and the `-DTOTONOTES`
127 option to `ptex2tex` or `doconce ptex2tex`, colorful margin or inline boxes (us-
128 ing the `todonotes` package) make it very easy to spot the comments.

129 Running

```
doconce format html mydoc.do.txt --skip_inline_comments
```

130 removes all inline comments from the output. This feature makes it easy to turn
131 on and off notes to authors during the development of the document.

132 All inline comments to readers can also be physically removed from the
133 Doconce source by

```
doconce remove_inline_comments mydoc.do.txt
```

134 0.9 Verbatim/Computer Code

135 Inline verbatim code is typeset within back-ticks, as in

```
Some sentence with 'words in verbatim style'.
```

136 resulting in Some sentence with words in verbatim style.

137 Multi-line blocks of verbatim text, typically computer code, is typeset in be-
138 tween `bc xxx!` and `ec!` directives, which must appear on the beginning of the
139 line. A specification `xxx` indicates what verbatim formatting style that is to be
140 used. Typical values for `xxx` are `nothing`, `cod` for a code snippet, `pro` for a com-
141 plete program, `sys` for a terminal session, `dat` for a data file (or output from
142 a program), `Xpro` or `Xcod` for a program or code snippet, respectively, in pro-
143 gramming `X`, where `X` may be `py` for Python, `cy` for Cython, `sh` for Bash or other
144 Unix shells, `f` for Fortran, `c` for C, `cpp` for C++, `m` for MATLAB, `p1` for Perl. For
145 output in `latex` one can let `xxx` reflect any defined verbatim environment in the
146 `ptex2tex` configuration file (`.ptex2tex.cfg`). For `sphinx` output one can insert
147 a comment

```
# sphinx code-blocks: pycod=python cod=fortran cppcod=c++ sys=console
```

148 that maps environments (`xxx`) onto valid language types for Pygments (which
149 is what `sphinx` applies to typeset computer code).

150 The `xxx` specifier has only effect for `latex` and `sphinx` output. All other
151 formats use a fixed monospace font for all kinds of verbatim output.

152 Here is an example of computer code (see the source of this document for
153 exact syntax):

```

from numpy import sin, cos, exp, pi
def f(x, y, z, t):
    return exp(-t)*sin(pi*x)*sin(pi*y)*cos(2*pi*z)

```

154 Computer code can also be copied from a file:

```

@@@CODE doconce_program.sh
@@@CODE doconce_program.sh fromto: doconce clean@^doconce split_rst
@@@CODE doconce_program.sh from-to: doconce clean@^doconce split_rst

```

155 The @@@CODE identifier must appear at the very beginning of the line. The first
156 specification copies the complete file doconce_program.sh. The second speci-
157 fication copies from the first line matching the *regular expression* doconce clean
158 up to, but not including the line matching the *regular expression* ^doconce split_rst.
159 The third specification behaves as the second, but the line matching the first
160 regular expression is not copied (aimed at copying text between begin-end
161 comment pair in the file).

162 The copied line from file are in this example put inside bc shpro! and ec!
163 directives, if a complete file is copied, while the directives become bc shcod!
164 and ec! when a code snippet is copied from file. In general, for a filename
165 extension .X, the environment becomes bc Xpro! or bc Xcod! for a complete
166 program or snippet, respectively. The environments (Xcod and Xpro) are only
167 active for latex and sphinx output.

168 Important warnings:

- 169 • A code block must come after some plain sentence (at least for successful
170 output in reStructuredText), not directly after a section/paragraph heading,
171 table, comment, figure, or movie.
- 172 • Verbatim code blocks inside lists can be ugly typeset in some output for-
173 mats. A more robust approach is to replace the list by paragraphs with
174 headings.

175 0.10 L^AT_EX Mathematics

176 Doconce supports inline mathematics and blocks of mathematics, using stan-
177 dard L^AT_EX syntax. The output formats html, sphinx, latex, pdflatex, pandoc,
178 and mwiki work with this syntax while all other formats will just display the raw
179 L^AT_EX code.

180 Inline expressions are written in the standard L^AT_EX way with the mathemat-
181 ics surrounded by dollar signs, as in $Ax = b$. To help increase readability in
182 other formats than sphinx, latex, and pdflatex, inline mathematics may have
183 a more human readable companion expression. The syntax is like

```

 $\sin(\|\mathbf{u}\|)$ 

```

184 That is, the L^AT_EX expression appears to the left of a vertical bar (pipe symbol)
185 and the more readable expression appears to the right. Both expressions are
186 surrounded by dollar signs.

187 Blocks of L^AT_EX mathematics are written within bt! and et! (begin/end TeX)
188 directives starting on the beginning of a line:

```

!bt
\begin{align*}
\nabla\cdot\pmb{u} &= 0,\\
\nabla\times\pmb{u} &= 0.
\end{align*}
!et

```

189 This \LaTeX code gets rendered as

$$\begin{aligned}\nabla \cdot \mathbf{u} &= 0, \\ \nabla \times \mathbf{u} &= 0.\end{aligned}$$

190 Here is a single equation:

```

!bt
\[ \frac{\partial\pmb{u}}{\partial t} + \pmb{u}\cdot\nabla\pmb{u} = 0.\]
!et

```

191 which results in

192
$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} = 0.$$

193 Any \LaTeX syntax is accepted, but if output in the `sphinx`, `pandoc`, `mwiki`,
194 `html`, or `ipynb` formats is also important, one should follow these rules:

- 195 • Use only the equation environments `\[`, `\]`, `equation`, `equation*`, `align`,
196 and `align*`.
- 197 • MediaWiki (`mwiki`) does not support references to equations.

198 (Doconce performs extensions to `sphinx` and other formats such that labels in
199 `align` environments work well.)



NOTICE

\LaTeX supports lots of fancy formatting, for example, multiple plots in the same figure, footnotes, margin notes, etc. Allowing other output formats, such as `sphinx`, makes it necessary to only utilize very standard \LaTeX and avoid, for instance, more than one plot per figure. However, one can use preprocessor if-tests on the format (typically `if FORMAT in ("latex", "pdflatex")`) to include special code for `latex` and `pdflatex` output and more straightforward typesetting for other formats. In this way, one can also allow advanced \LaTeX features and fine tuning of resulting PDF document.

200 **LaTeX Newcommands.** The author can define `newcommand` statements in
 201 files with names `newcommands*.tex`. Such commands should only be used
 202 for mathematics (other `LaTeX` constructions are only understood by `LaTeX` itself).
 203 The convention is that `newcommands_keep.tex` contains the newcommands
 204 that are kept in the document, while those in `newcommands_replace.tex` will
 205 be replaced by their full `LaTeX` code. This conventions helps make readable doc-
 206 uments in formats without `LaTeX` support. For `html`, `sphinx`, `latex`, `pdflatex`,
 207 `mwiki`, `ipynb`, and `pandoc`, the mathematics in newcommands is rendered
 208 nicely anyway.

209 0.11 Figures and Movies

210 Figures and movies have almost equal syntax:

FIGURE: `[relative/path/to/figurefile, width=500]` Here goes the caption which must be on a single line.

MOVIE: `[relative/path/to/moviefile, width=500]` Here goes the caption which must be on a single line. 1

211 Note three important syntax details:

- 212 1. A mandatory comma after the figure/movie filename,
- 213 2. all of the command must appear on a single line,
- 214 3. there must be a blank line after the command.

215 The figure file can be listed without extension. Doconce will then find the ver-
 216 sion of the file with the most appropriate extension for the chosen output for-
 217 mat. If not suitable version is found, Doconce will convert another format to the
 218 needed one.

219 Movie files can either be a video or a wildcard expression for a series of
 220 frames. In the latter case, a simple device in an HTML page will display the
 221 individual frame files as a movie.

222 Combining several image files into one can be done by the

```
doconce combine_images image1 image2 ... output_image
```

223 This command applies `montage` or PDF-based tools to combine the images to
 224 get the highest quality.

225 YouTube and Vimeo movies will be embedded in `html` and `sphinx` docu-
 226 ments and otherwise be represented by a link. The syntax is

MOVIE: `[http://www.youtube.com/watch?v=_07iUiftbKU, width=420 height=315]` YouTube movie.

MOVIE: `[http://vimeo.com/55562330, width=500 height=278]` Vimeo movie.

227 The latter results in

228 Vimeo movie. `http://vimeo.com/55562330`

229 **0.12 Tables**

230 The table in Section 0.4 was written with this syntax:

Section type	Syntax
chapter	'===== Heading =====' (9 '=')
section	'===== Heading =====' (7 '=')
subsection	'===== Heading =====' (5 '=')
subsubsection	'===== Heading =====' (3 '=')
paragraph	'__Heading__' (2 '_')

231 Note that

- 232 • Each line begins and ends with a vertical bar (pipe symbol).
- 233 • Column data are separated by a vertical bar (pipe symbol).
- 234 • There may be horizontal rules, i.e., lines with dashes for indicating the
235 heading and the end of the table, and these may contain characters 'c', 'l',
236 or 'r' for how to align headings or columns. The first horizontal rule may
237 indicate how to align headings (center, left, right), and the horizontal rule
238 after the heading line may indicate how to align the data in the columns
239 (center, left, right).
- 240 • If the horizontal rules are without alignment information there should be
241 no vertical bar (pipe symbol) between the columns. Otherwise, such a
242 bar indicates a vertical bar between columns in \LaTeX .
- 243 • Many output formats are so primitive that heading and column alignment
244 have no effect.

245 **0.13 Labels and References**

246 The notion of labels and references (as well as bibliography and index) is
247 adopted from \LaTeX with a very similar syntax. As in \LaTeX , a label can be in-
248 serted anywhere, using the syntax

`label{name}`

249 with no backslash preceding the label keyword. It is common practice to choose
250 name as some hierarchical name, say a:b:c, where a and b indicate some
251 abbreviations for a section and/or subsection for the topic and c is some name
252 for the particular unit that has a label.

253 A reference to the label name is written as

`ref{name}`

254 again with no backslash before ref.

255 Use labels for sections and equations only, and precede the reference by
256 "Section" or "Chapter", or in case of an equation, surround the reference by
257 parenthesis.

258 0.14 Citations and Bibliography

259 Single citations are written as

```
cite{name}
```

260 where `name` is a logical name of the reference (again, \LaTeX writers must not in-
261 sert a backslash). Bibliography citations often have `name` on the form `Author1_Author2_YYYY`,
262 `Author_YYYY`, or `Author1_etal_YYYY`, where `YYYY` is the year of the publica-
263 tion. Multiple citations at once is possible by separating the logical names by
264 comma:

```
cite{name1,name2,name3}
```

265 The bibliography is specified by a line `BIBFILE: papers.pub`, where `papers.pub`
266 is a publication database in the Publish format. \BibTeX .bib files can easily be
267 combined to a Publish database (which Doconce needs to create bibliogra-
268 phies in other formats than \LaTeX).

269 0.15 Generalized Citations

270 There is a *generalized referencing* feature in Doconce that allows a reference
271 with `ref` to have one formulation if the label is in the same document and an-
272 other formulation if the reference is to an item in an external document. This
273 construction makes it easy to work with many small, independent documents
274 in parallel with a book assembly of some of the small elements. The syntax of
275 a generalized reference is

```
ref[internal][cite][external]
```

```
# Example:  
As explained in  
ref[Section ref{subsec:ex}][in cite{testdoc:12}][a "section":  
"testdoc.html#_sec2" in the document  
"A Document for Testing Doconce": "testdoc.html" cite{testdoc:12}],  
Doconce documents may include movies.
```

276 The output from a generalized reference is the text `internal` if all `ref{label}`
277 references in `internal` are references to labels in the present document. Oth-
278 erwise, if `cite` is non-empty and the format is `latex` or `pdflatex` one assumes
279 that the references in `internal` are to external documents declared by a com-
280 ment line `# Externaldocuments: testdoc, mydoc` (usually after the title, au-
281 thors, and date). In this case the output text is `internal cite` and the \LaTeX
282 package `xr` is used to handle the labels in the external documents. If none of
283 the two situations above applies, the `external` text will be the output.

284 0.16 Index of Keywords

285 Doconce supports creating an index of keywords. A certain keyword is regis-
286 tered for the index by a syntax like (no backslash!)

```
index{name}
```

287 It is recommended to place any index of this type outside running text, i.e., after
288 (sub)section titles and in the space between paragraphs. Index specifications

289 placed right before paragraphs also gives the doconce source code an indica-
 290 tion of the content in the forthcoming text. The index is only produced for the
 291 latex, pdflatex, rst, and sphinx formats.

292 **0.17 Capabilities of The Program** doconce

293 The doconce program can be used for a number of purposes besides trans-
 294 forming a .do.txt file to some format. Here is the list of capabilities:

```
Usage: doconce command [optional arguments]
commands: format help sphinx_dir subst replace replace_from_file clean spellcheck ptex2tex expand_commm

# transform doconce file to another format
doconce format html|latex|pdflatex|rst|sphinx|plain|gwiki|mwiki|cwiki|pandoc|st|epytext file.do.txt

# substitute a phrase by another using regular expressions
doconce subst [-s -m -x --restore] regex-pattern regex-replacement file1 file2 ...
(-s is the re.DOTALL modifier, -m is the re.MULTILINE modifier,
 -x is the re.VERBOSE modifier, --restore copies backup files back again)

# replace a phrase by another literally
doconce replace from-text to-text file1 file2 ...
(exact text substitution)

# doconce replace using from and to phrases from file
doconce replace_from_file file-with-from-to file1 file2 ...
(exact text substitution, but a set of from-to relations)

# gwiki format requires substitution of figure file names by URLs
doconce gwiki_figsubst file.gwiki URL-of-fig-dir

# remove all inline comments in a doconce file
doconce remove_inline_comments file.do.txt

# create a directory for the sphinx format
doconce sphinx_dir author='John Doe' title='Long title' \
    short_title="Short title" version=0.1 \
    dirname=sphinx-rootdir theme=default logo=mylogo.png \
    do_file [do_file2 do_file3 ...]
(requires sphinx version >= 1.1)

# replace latex-1 (non-ascii) characters by html codes
doconce latin2html file.html

# walk through a directory tree and insert doconce files as
# docstrings in *.py files
doconce insertdocstr rootdir

# remove all files that the doconce format can regenerate
doconce clean

# print the header (preamble) for latex file
doconce latex_header

# print the footer for latex files
doconce latex_footer

# change encoding
doconce change_encoding utf-8 latin1 filename

# guess the encoding in a text
doconce guess_encoding filename

# transform a .bbl file to a .rst file with reST bibliography format
doconce bbl2rst file.bbl
```

```

# split a sphinx/rst file into parts
doconce format sphinx complete_file
doconce split_rst complete_file          # !split delimiters
doconce sphinx_dir complete_file
python automake_sphinx.py

# edit URLs to local files and place them in _static
doconce sphinxfix_local_URLs file.rst

# split an html file into parts according to !split commands
doconce split_html complete_file.html

# create slides from a (doconce) html file
doconce slides_html slide_type complete_file.html

# replace bullets in lists by colored bullets
doconce html_colorbullets file1.html file2.html ...

# grab selected text from a file
doconce grab --from[-] from-text [--to[-] to-text] somefile

# remove selected text from a file
doconce remove --from[-] from-text [--to[-] to-text] somefile

# run spellcheck on a set of files
doconce spellcheck [-d .mydict.txt] *.do.txt

# transform ptex2tex files (.p.tex) to ordinary latex file
# and manage the code environments
doconce ptex2tex mydoc -DMINTED pycod=minted sys=Verbatim \
    dat=\begin{quote}\begin{verbatim};\end{verbatim}\end{quote}

# make HTML file via pandoc from Markdown (.md) file
doconce md2html file

# make LaTeX file via pandoc from Markdown (.md) file
doconce md2latex file

# expand short cut commands to full form in files
doconce expand_commands file1 file2 ...

# combine several images into one
doconce combine_images image1 image2 ... output_file

# insert a table of exercises in a latex file myfile.p.tex
doconce latex_exercise_toc myfile

# list all labels in a document (for purposes of cleaning them up)
doconce list_labels myfile

# translate a latex document to doconce (requires usually manual fixing)
doconce latex2doconce latexfile

# check if there are problems with translating latex to doconce
doconce latex_dislikes latexfile

# typeset a doconce document with pygments (for pretty print of doconce itself)
doconce pygmentize myfile [pygments-style]

# generate a make.sh script for translating a doconce file to various formats
doconce makefile docname doconcefile [html sphinx pdflatex ...]

# fix common problems in bibtex files for publish import
doconce fix_bibtex4publish file1.bib file2.bib ...

# find differences between two files
doconce diff file1.do.txt file2.do.txt [diffprog]
(diffprog can be diff, pdiff, latexdiff, kdiff3, diff, ...)

```

```
# find differences between the last two Git versions of several files
doconce gitdiff file1 file2 file3 ...
```

295 0.18 Exercises

296 Doconce supports *Exercise*, *Problem*, *Project*, and *Example*. These are type-
297 set as ordinary sections and referred to by their section labels. Exercise, prob-
298 lem, project, or example sections contains certain *elements*:

- 299 • a headline at the level of a subsection containing one of the words "Exer-
300 cise:", "Problem:", "Project:", or "Example:", followed by a title (required)
- 301 • a label (optional)
- 302 • a solution file (optional)
- 303 • name of file with a student solution (optional)
- 304 • main exercise text (required)
- 305 • a short answer (optional)
- 306 • a full solution (optional)
- 307 • one or more hints (optional)
- 308 • one or more subexercises (subproblems, subprojects), which can also
309 contain a text, a short answer, a full solution, name student file to be
310 handed in, and one or more hints (optional)

311 A typical sketch of a a problem without subexercises goes as follows:

```
===== Problem: Derive the Formula for the Area of an Ellipse =====
label{problem:ellipsearea1}
file=ellipse_area.pdf
solution=ellipse_area1_sol.pdf

Derive an expression for the area of an ellipse by integrating
the area under a curve that defines half of the allipse.
Show each step in the mathematical derivation.

!bhint
Wikipedia has the formula for the curve.
!ehint

!bhint
"Wolframalpha": "http://wolframalpha.com" can perhaps
compute the integral.
!ehint
```

312 If the exercise type (Exercise, Problem, Project, or Example) is enclosed in
313 braces, the type is left out of the title in the output. For example, the if the title
314 line above reads

```
===== {Problem}: Derive the Formula for the Area of an Ellipse =====
```

315 the title becomes just "Derive the ...".

316 An exercise with subproblems, answers and full solutions has this setup-up:

```

===== Exercise: Determine the Distance to the Moon =====
label{exer:moondist}

Intro to this exercise. Questions are in subexercises below.

!bsubex
Subexercises are numbered a), b), etc.

file=subexer_a.pdf

!bans
Short answer to subexercise a).
!eans

!bhint
First hint to subexercise a).
!ehint

!bhint
Second hint to subexercise a).
!ehint
!esubex

!bsubex
Here goes the text for subexercise b).

file=subexer_b.pdf

!bhint
A hint for this subexercise.
!ehint

!bsol
Here goes the solution of this subexercise.
!esol
!esubex

!bremarks
At the very end of the exercise it may be appropriate to summarize
and give some perspectives. The text inside the !bremarks-!eremarks
directives is always typeset at the end of the exercise.
!eremarks

!bsol
Here goes a full solution of the whole exercise.
!esol

```

317 By default, answers, solutions, and hints are typeset as paragraphs. The
 318 command-line arguments `--without_answers` and `--without_solutions` turn
 319 off output of answers and solutions, respectively, except for examples.

320 **0.19 Environments**

321 Doconce environments start with `benvirname!` and end with `eenvirname!`, where
 322 `envirname` is the name of the environment. Here is a listing of the environ-
 323 ments:

- 324 • `c`: computer code (or verbatim text)
- 325 • `t`: math blocks with \LaTeX syntax
- 326 • `subex`: sub-exercise
- 327 • `ans`: short answer to exercise or sub-exercise

- `sol`: full solution to exercise or sub-exercise
- `quote`: indented text
- `notice`, `summary`, `warning`, `question`, `hint`: admonition boxes with special icon and (frequently) background color
- `pop`: text to gradually pop up in slide presentations
- `slidecell`: indication of cells in a grid layout for elements on a slide

0.20 Preprocessing

Doconce documents may utilize a preprocessor, either `preprocess` and/or `mako`. The former is a C-style preprocessor that allows if-tests and including other files (but not macros with arguments). The `mako` preprocessor is much more advanced - it is actually a full programming language, very similar to Python.

The command `doconce format` first runs `preprocess` and then `mako`. Here is a typical example on utilizing `preprocess` to include another document, "comment out" a large portion of text, and to write format-specific constructions:

```
# #include "myotherdoc.do.txt"

# #if FORMAT in ("latex", "pdflatex")
\begin{table}
\caption{Some words... label{mytab}}
\begin{tabular}{lrr}
\hline\noalign{\smallskip}
\multicolumn{1}{c}{time} & \multicolumn{1}{c}{velocity} & \multicolumn{1}{c}{acceleration} \\
\hline
0.0 & 1.4186 & -5.01 \\
2.0 & 1.376512 & 11.919 \\
4.0 & 1.1E+1 & 14.717624 \\
\hline
\end{tabular}
\end{table}
# #else


| time | velocity | acceleration |
|------|----------|--------------|
| 0.0  | 1.4186   | -5.01        |
| 2.0  | 1.376512 | 11.919       |
| 4.0  | 1.1E+1   | 14.717624    |


# #endif

# #ifdef EXTRA_MATERIAL
...large portions of text...
# #endif
```

With the `mako` preprocessor the if-else tests have slightly different syntax. An example document contains some illustrations on how to utilize `mako` (clone the GitHub project and examine the Doconce source and the `doc/src/make.sh` script).

0.21 Resources

- Excellent "Sphinx Tutorial" by C. Reller: "<http://people.ee.ethz.ch/~creller/web/tricks/reST.html>"