DocOnce: Document Once, Include Anywhere

Hans Petter Langtangen[1,2]

[1]Center for Biomedical Computing, Simula Research Laboratory
[2]Department of Informatics, University of Oslo

Oct 27, 2015

1 Some DocOnce Features

- Strong support for texts with much math and code.
- Same source can produce a variety of output formats:
  - traditional LaTeX B/W documents for printing
  - color LaTeX PDF documents
  - color LaTeX PDF documents for viewing on small phones
  - Sphinx HTML documents with 20+ different designs
  - Plain HTML, Boostrap HTML, or with a template, or with another template, or solarized
  - HTML for Google or Wordpress blog posts
  - MediaWiki (Wikipedia, Wikibooks, etc)
  - Markdown
  - IPython notebook

  Other formats include plain untagged text (for email), Creole wiki (for Bitbucket wikis), Google wiki (for Googlecode), reStructuredText, and Epytext.
- Integration with Mako enables use of variables, functions, if-tests, and loops to parameterize the text and generate various versions of the text for different purposes.
- Computer code can be copied directly from parts of source code files.

- Running text can quickly be edited to slide formats (reveal.js and deck.js, based on HTML5+CSS3).
- Special exercise environments with support for hints, answers, subexercises, etc.
- Automatic inline embedding of YouTube and Vimeo movies.
- Good support for admonitions in various LaTeX and HTML styles for warnings, questions, hints, remarks, summaries, etc.

2 What Does DocOnce Look Like?

DocOnce text looks like ordinary text (much like Markdown[1]), but there are some almost invisible text constructions that allow you to control the formating. Here are some examples.

- Bullet lists automatically arise from lines starting with `*`, or `o` if the list is to be enumerated.
- *Emphasized words* are surrounded by `*`. **Words in boldface** are surrounded by underscores.
- Words from computer code are enclosed in backticks and then typeset `verbatim (in a monospace font)`.
- Section and paragraph headings are recognied special decorating characters (= or _) before and after the heading. The length of the decoration determines the level of the section.
- Blocks of computer code are included by surrounding the blocks with `!bc` (begin code) and `!ec` (end code) tags on separate lines.
- Blocks of computer code can also be imported from source files.
- Blocks of LaTeX mathematics are included by surrounding the blocks with `!bt` (begin TeX) and `!et` (end TeX) tags on separate lines.
- There is support for both LaTeX and text-like inline mathematics such that formulas make sense also when not rendered by LaTeX or MathJax.
- Figures and movies with captions, simple tables, URLs with links, index list, labels and references are supported. YouTube and Vimeo videos are automatically embedded in web documents.

[1] In fact, DocOnce allows basic GitHub/extended Markdown syntax as input. This is attractive for newcomers from Markdown or writers who also write Markdown documents (or uses Markdown frequently at GitHub).

- The abstract of a document starts with *Abstract* as paragraph heading, and all text up to the next heading makes up the abstract,
- Special comment lines are not visible in the output.
- Comments to authors can be inserted throughout the text and made visible or invisible as desired.
- There is an exercise environment with many advanced features.
- With a preprocessor, Preprocess or Mako, one can include other documents (files), large portions of text can be defined in or out of the text, and tailored format-specific constructs can easily be included. With Mako a single text can output its program examples in two or more languages.

2.1 What Can DocOnce Be Used For?

LaTeX is ideal for articles, thesis, and books, but the PDF files does not look fresh and modern on tablets and phones or big computer screens. For the latter type of media you need HTML-based documents with strong support for nice layouts. Tools like Sphinx, Markdown, or plain HTML with Bootstrap are then more appropriate than LaTeX, but involves a very different syntax. DocOnce lets you write one text in one place and then generate the most appropriate language for the media you want to target. DocOnce also has many extra features for supporting large documents with much code and mathematics, not found in any of other publishing tool.

2.2 Basic Syntax

Here is an example of some simple text written in the DocOnce format:

```
======= First a Section Heading with 7 = Characters =======

===== Then a Subsection Heading with 5 = Characters =====

=== Finally a Subsubection Heading with 3 = Characters ===

You can also have paragraphs with a paragraph heading surrounded
by double underscores are the beginning of a line.

__This is a paragraph heading.__
And here comes the text.

===== A Subsection with Sample Text =====
label{my:first:sec}

Ordinary text looks like ordinary text, but must always start at the
beginning of lines. Tags used for _boldface_ words, *emphasized*
words, and `computer` words look natural in plain text.  Quotations
appear inside double backticks and double single quotes, as in ``this
example''.
```

```
Below the section title we have a *label*, which can be used to
refer to Section ref{my:first:sec}.
References to equations, such as (ref{myeq1}), work in the same
LaTeX-inspired way.

Lists are typeset as you would do in email,

  * item 1
  * item 2,
    perhaps with a 2nd line
  * item 3

Note the consistent use of indentation (as in Python programming!).
Lists can also have automatically numbered items instead of bullets,

  o item 1
  o item 2
  o item 3,
    but be careful with the indentation of the next lines!

__Hyperlinks.__
URLs with a link word are possible, as in "hpl": "http://folk.uio.no/hpl".
If the word is just URL, the URL itself becomes the link name,
as in URL: "tutorial.do.txt". DocOnce distinguishes between paper
and screen output. In traditional paper output, in PDF generated from LaTeX
generated from DocOnce, the URLs of links appear as footnotes.
With screen output, all links are clickable hyperlinks, except in
the plain text format which does not support hyperlinks.

__Inline comments.__
DocOnce also allows inline comments of the form [name: comment] (with
a space after `name:`), e.g., such as [hpl: here I will make some
remarks to the text]. Inline comments can be removed from the output
by a command-line argument (see Section ref{doconce2formats} for an
example). Inline comments can also be used for detailed editing of
text, much like track changes in word, to illustrate how a text
is revised. (However, for seeing how others have revised the text, I
strongly recommend using Git for version control and running `git diff`
on the appropriate versions, or you can click on differences at
GitHub if the files are hosted there.)

__Footnotes.__ Adding a footnote[^footnote] is also possible.

[^footnote]: The syntax for footnotes is borrowed from Extended Markdown.

__Tables.__
Tables are also written in the plain text way, e.g.,

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

The characters `c`, `r`, and `l` can be inserted, as illustrated above,
for aligning the headings and the columns (center, right, left).
One can also use `X` for potentially very long text in a column (will be
left-adjusted).
```

```
# Lines beginning with # are comment lines.
```

The DocOnce text above results in the following little document:

3 First a Section Heading with 7 = Characters

3.1 Then a Subsection Heading with 5 = Characters

Finally a Subsubection Heading with 3 = Characters. You can also have paragraphs with a paragraph heading surrounded by double underscores are the beginning of a line.

This is a paragraph heading. And here comes the text.

3.2 A Subsection with Sample Text

Ordinary text looks like ordinary text, but must always start at the beginning of lines. Tags used for **boldface** words, *emphasized* words, and `computer` words look natural in plain text. Quotations appear inside double backticks and double single quotes, as in "this example".

Below the section title we have a *label*, which can be used to refer to Section 3.2. References to equations, such as '(1)', work in the same LaTeX-inspired way.

Lists are typeset as you would do in email,

- item 1
- item 2, perhaps with a 2nd line
- item 3

Note the consistent use of indentation (as in Python programming!). Lists can also have automatically numbered items instead of bullets,

1. item 1
2. item 2
3. item 3, but be careful with the indentation of the next lines!

Hyperlinks. URLs with a link word are possible, as in hpl. If the word is just URL, the URL itself becomes the link name, as in `tutorial.do.txt`. DocOnce distinguishes between paper and screen output. In traditional paper output, in PDF generated from LaTeX generated from DocOnce, the URLs of links appear as footnotes. With screen output, all links are clickable hyperlinks, except in the plain text format which does not support hyperlinks.

Inline comments. DocOnce also allows inline comments of the form **name 1**: comment (with a space after `name:`), e.g., such as **hpl 2**: here I will make some remarks to the text . Inline comments can be removed from the output by a command-line argument (see Section 4 for an example). Inline comments can also be used for detailed editing of text, much like track changes in word, to illustrate how a text is revised. (However, for seeing how others have revised the text, I strongly recommend using Git for version control and running `git diff` on the appropriate versions, or you can click on differences at GitHub if the files are hosted there.)

Footnotes. Adding a footnote[2] is also possible.

Tables. Tables are also written in the plain text way, e.g.,

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

The characters `c`, `r`, and `l` can be inserted, as illustrated above, for aligning the headings and the columns (center, right, left).

3.3 Mathematics

Inline mathematics, such as $\nu = \sin(x)$ is written exactly as in LaTeX:

```
$\nu = \sin(x)$
```

Blocks of mathematics are typeset with raw LaTeX, inside `!bt` and `!et` (begin TeX, end TeX) directives:

```
!bt
\begin{align}
{\partial u\over\partial t} &= \nabla^2 u + f,
label{myeq1}\\
{\partial v\over\partial t} &= \nabla\cdot(q(u)\nabla v) + g
\end{align}
!et
```

The result looks like this:

$$\frac{\partial u}{\partial t} = \nabla^2 u + f, \qquad (1)$$

$$\frac{\partial v}{\partial t} = \nabla\cdot(q(u)\nabla v) + g \qquad (2)$$

[2]The syntax for footnotes is borrowed from Extended Markdown.

Of course, such blocks only looks nice in formats with support for LaTeX mathematics (this includes `latex`, `pdflatex`, `html`, `sphinx`, `ipynb`, `pandoc`, and `mwiki`). Simpler formats have to just list the raw LaTeX syntax.

LaTeX writers who adopt DocOnce need to pay attention to the following:

- AMS LaTeX mathematics is supported, also for the `html`, `sphinx`, and `ipynb` formats.
- Only five equation environments can be used: `\[ ... \]`, `equation*`, `equation`, `align*`, and `align`.
- Newcommands in mathematical formulas are allowed (but not in the running text). Newcommands must be defined in files with names `newcommands*.tex`.
- MediaWiki (`mwiki`) does not support references to equations. DocOnce has extended Markdown, Sphinx, and HTML to better support equation referencing (across web pages for example).
- Only figures are floating elements in DocOnce, all other elements (code, tables, algorithms) are *inline* without numbers or labels for reference. The reason is that floating elements are in general not used in web documents, but we made an exception with figures and movies.

Remark.

Although DocOnce allows user-defined styles in the preamble of LaTeX output, HTML-based output cannot make use of such styles. If-else constructs for the preprocessor can be used to allow special LaTeX environments for LaTeX output and alternative typesetting for other formats, but it is recommended to stay away from special environments in the text and write in a simpler fashion. For example, DocOnce has no special construction for algorithms, so these must be simulated by lists or verbatim blocks. Other constructions that should be avoided include margin notes, special tables, and `subfigure` (combine image files to one file instead, via `doconce combine_images`).

3.4 Computer Code

You can have blocks of computer code, starting and ending with `!bc` and `!ec` instructions, respectively.

```
!bc pycod
from math import sin, pi
def myfunc(x):
    return sin(pi*x)
```

```
import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
!ec
```

Such blocks are formatted as

```
from math import sin, pi
def myfunc(x):
    return sin(pi*x)

import integrate
I = integrate.trapezoidal(myfunc, 0, pi, 100)
```

A code block must come after some plain sentence (at least for successful output to `sphinx`, `rst`, and formats close to plain text), not directly after a section/paragraph heading or a table.

Blocks of computer code has *named environments*, such as *pycod*. The *py* stands for Python and *cod* indicates a code snippet that cannot be run without more code. Another example is *fpro*, *f* for Fortran and *pro* for a complete program that will run as it stands. There is support for code in C, C++, Fortran, Java, Python, Perl, Ruby, JavaScript, HTML, and LaTeX,

One can also copy computer code directly from files, either the complete file or specified parts, e.g,

```
 @@@CODE src/myprog.py fromto: def regression\(@import mymod
```

The copying is based on regular expressions and not on line numbers, which makes the specifications much more robust during software and document developing. With the `@@@CODE` command, computer code is never duplicated in the documentation (important for the principle of avoiding copying information!) and once the software is updated, the next compilation of the document is up-to-date.

Inclusion of files. Another DocOnce document or any file can be included by writing `# #include "mynote.do.txt"` at the beginning of a line. DocOnce documents have extension `do.txt`. The `do` part stands for doconce, while the trailing `.txt` denotes a text document so that editors gives you plain text editing capabilities.

3.5 Macros (Newcommands), Cross-References, Index, and Bibliography

DocOnce supports a type of macros via a LaTeX-style *newcommand* construction. The newcommands are defined in files with names `newcommands*.tex`, using standard LaTeX syntax. Only newcommands for use inside math environments are supported.

Labels, corss-references, citations, and support of an index and bibliography are much inspired by LaTeX syntax, but DocOnce features no backslashes.

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

Here is an example:

```
===== My Section =====
label{sec:mysec}

idx{key equation} idx{$\u$ conservation}

We refer to Section ref{sec:yoursec} for background material on
the *key equation*. Here we focus on the extension

# \Ddt, \u and \mycommand are defined in newcommands_keep.tex

!bt
\begin{equation}
\Ddt{\u} = \mycommand{v},
label{mysec:eq:Dudt}
\end{equation}
!et
where $\Ddt{\u}$ is the material derivative of $\u$.
Equation (ref{mysec:eq:Dudt}) is important in a number
of contexts, see cite{Larsen_et_al_2002,Johnson_Friedman_2010a}.
Also, cite{Miller_2000} supports such a view.

As see in Figure ref{mysec:fig:myfig}, the key equation
features large, smooth regions *and* abrupt changes.

FIGURE: [fig/myfile, width=600 frac=0.9] My figure. label{mysec:fig:myfig}

===== References =====

BIBFILE: papers.pub
```

DocOnce applies Publish for specifying bibliographies because this tool has more functionality than BibTeX, but any BibTeX database can be automatically converted to the simple Publish format.

For further details on functionality and syntax we refer to the `doc/manual/manual.do.txt` file in the DocOnce source and a Sphinx version of this document.

4 From DocOnce to Other Formats

We refer to the manual for detailed information on how to compile a DocOnce document to various formats. Here we just give a glimpse of the possibilities.

4.1 Example: Making an HTML File

Suppose you have some DocOnce text in `mydoc.do.txt`. Here is how you compile that document to an HTML file `mydoc.html`, which can be viewed in a web browser:

```
Terminal> doconce format html mydoc --html_style=bootswatch_journal
```

There are lots of styles for HTML files, and `bootswatch_journal` is a fancy one. There are also lots of other command-line options for tailoring the compilation. Run `doconce format -help` to see a list of all options. Those that start with `--html_` are specific for the HTML output format.

4.2 Preprocessors

A DocOnce compilation has three stages:

1. The preprocessor Preprocess is applied to `mydoc.do.txt`, resulting in `tmp_preprocess__mydoc.do.txt`.
2. The preprocessor Mako is applied to `tmp_preprocess__mydoc.do.txt`, resulting in `tmp_mako__mydoc.do.txt`.
3. The text in `tmp_mako__mydoc.do.txt` is translated to the chosen output format.

The preprocessor stages are only run if you have applied preprocessor syntax. The Preprocess program allows you to include other files (usually other DocOnce files) in your document (nested includes are possible). You can also have if-else branching based on variables set on the command line. The Mako preprocessor is (much) more advanced and features if-else tests, loops, variables, and function calls. You can, e.g., write Python functions in Mako to make quite intelligent output (e.g., copy computer code from a certain directory based on a variable that tells which computer language the document is to apply). The preprocessors are definitely one of the strongest features of DocOnce.

4.3 Output Formats

DocOnce can be translated to many formats. For documents with much mathematics and/or computer code the following formats are suitable:

- `latex`, `pdflatex`
- `html`
- `sphinx`
- `ipynb` (IPython/Jupyter notebooks)
- `matlabnb` (Matlab notebooks)

Other formats are

- Wikis: `gwiki` (Googlecode), `cwiki` (Creole), `mwiki` (MediaWiki)
- `plain` (pure ascii)
- `pandoc` (various Markdown formats)

4.4 Slides

DocOnce has strong support for writing slides, see the slides demo for examples. Each slide starts with a subsection heading (5 =), preceded by `!split` to indicate a new slide. Section headings are used to mark parts of the presentation. The slides are compiled as any other DocOnce file, but there is usually a second step where the text is modified to become a proper slide text in the chosen output format. We refer to the manual for details and the

Several popular slide formats are supported:

- LaTeX Beamer
- HTML5: reveal.js, deck.js, CSSS
- Markdown: Remark
- Plain HTML

5 Demos

The current text is generated from a DocOnce format stored in the file

```
doc/tutorial/tutorial.do.txt
```

The file `make.sh` in the `tutorial` directory of the DocOnce source code contains a demo of how to produce a variety of formats. The source of this tutorial, `tutorial.do.txt` is the starting point. Running `make.sh` and studying the various generated files and comparing them with the original `tutorial.do.txt` file, gives a quick introduction to how DocOnce is used in a real case.

A short scientific report demonstrates the vast choice of output formats and settings that are available.

There is also a demo of different slide formats.

6 Installation of DocOnce and its Dependencies

Below, we explain the manual installation of all software that may be needed when working with DocOnce documents. The impatient way to install everything that is needed is to use Anaconda Python and the `conda` program:

```
Terminal> conda install --channel johannr doconce
```

The `conda` package is available for Mac and Linux only.

If you do not want to use Anaconda and are on a Debian-based Linux computer (running, e.g., Ubuntu), you can instead run the Bash script `install_doconce.sh` or the equivalent Python script `install_doconce.py`. These scripts gives a comprehensive installation. Some users will prefer to install just what is needed for them, and this is explained below.

Version control systems are needed!

The coming installation instructions require that the version control systems Subversion, Mercurial, and Git are installed on your computer.

What about Mac and Windows?

DocOnce is primarily tested on GNU/Debian Linux systems, but also to a minor extent on Mac OS X. Experience with Windows is limited. Since most packages are Python-based and can be installed via `pip install` no problems should arise on Mac and Windows. However, some of the image processing tools and spell checking apply Unix-specific software.

You can omit reading the next sections if you rely on `conda` or `apt-get install` commands in the Bash script for installing DocOnce.

6.1 DocOnce

DocOnce itself is pure Python code hosted at https://github.com/hplgit/doconce. Installation can be done via

```
sudo pip install -e git+https://github.com/hplgit/doconce#egg=doconce
# or if doconce is already installed
sudo pip install -e git+https://github.com/hplgit/doconce#egg=doconce --upgrade
```

However, the recommended approach is to have a copy of the source on the local computer and run `setup.py`:

```
git clone git@github.com:hplgit/doconce.git
cd doconce
sudo python setup.py install
cd ..
```

Since DocOnce is frequently updated, it becomes necessary to ensure that you work with the most recent version:

```
cd doconce
git pull origin master
sudo python setup.py install
```

6.2 Dependencies

Producing HTML documents, plain text, pandoc-extended Markdown, and wikis can be done without installing any other software. However, if you want other formats as output (LaTeX, Sphinx, reStructuredText) and assisting utilities such as preprocesors, spellcheck, file differences, bibliographies, and so on, a lot of extra software must be installed.

Python v2.7. First you need Python version 2.7 and the `pip` installation program. Unless you already have these, we recommend to install a comprehensive Python bundle like Anaconda.

You do not need more software if you avoid using preprocessors, there is no bibliography, and you stick to the output formats LaTeX and HTML (you need of course LaTeX installed to process `.tex` files).

Preprocessors. If you make use of the Preprocess preprocessor, this program must be installed:

```
pip install -e \
   git+https://github.com/hplgit/preprocess#egg=preprocess --upgrade
```

A much more advanced alternative to Preprocess is Mako. Its installation is done by

```
pip install Mako
```

It is recommended to install both Preprocess and Mako.

Note that neither Preprocess nor Mako is run if you do not have preprocessor directives in your DocOnce source. That is, you only need this extra software if you make active use of preprocessors.

Bibliography. The Python package Publish is needed if you use a bibliography in your document (`cite` commands and a `BIBFILE:` specification). The installation is done by

```
pip install -e hg+https://bitbucket.org/logg/publish#egg=publish
```

Image file handling. Different output formats require different formats of image files. For example, PDF or PNG is used for `pdflatex`, PostScript for `latex`, while HTML needs JPEG, GIF, or PNG formats. DocOnce calls up programs from the ImageMagick suite for converting image files to a proper format if needed. The ImageMagick suite can be installed on all major platforms. On Debian Linux (including Ubuntu) systems one can simply write

```
sudo apt-get install imagemagick
```

The convenience program `doconce combine_images`, for combining several images into one, will use `montage` and `convert` from ImageMagick and the `pdftk`, `pdfnup`, and `pdfcrop` programs from the `texlive-extra-utils` Debian package. The latter gets installed by

```
sudo apt-get install texlive-extra-utils
```

Automatic image conversion from EPS to PDF calls up `epstopdf`, which can be installed by

```
sudo apt-get install texlive-font-utils
```

Spellcheck. The utility `doconce spellcheck` applies the `ispell` program for spellcheck. On Debian (including Ubuntu) it is installed by

```
sudo apt-get install ispell
```

Ptex2tex for LaTeX Output. Originally, DocOnce relied on the ptex2tex program for very flexible choice of typesetting of verbatim code blocks. A simplified version, `doconce ptex2tex`, is bundled with DocOnce. However, even greater flexibility is now offered by the `--latex_code_style=` option to `doconce format` so unless you already are a `ptex2tex` user, it is recommended to forget about `ptex2tex` and just use the `--latex_code_style=` option.

The stand-alone `ptex2tex` program is installed by

```
svn checkout http://ptex2tex.googlecode.com/svn/trunk/ ptex2tex
cd ptex2tex
sudo python setup.py install
```

It may happen that you need additional style files, you can run a script, `cp2texmf.sh`:

```
cd latex
sh cp2texmf.sh  # copy stylefiles to ~/texmf directory
cd ../..
```

This script copies some special stylefiles that that `ptex2tex` potentially makes use of. Some more standard stylefiles are also needed. These are installed by

```
sudo apt-get install texlive
```

on Debian Linux (including Ubuntu) systems. TeXShop on Mac comes with the necessary stylefiles (if not, they can be found by googling and installed manually in the `/texmf/tex/latex/misc` directory).

Note that the `doconce ptex2tex` command, which needs no installation beyond DocOnce itself, can be used as a simpler alternative to the `ptex2tex` program.

Pygments and the Minted Code Style. The *minted* LaTeX style is popular for typesetting code. This style requires the package Pygments to be installed. On Debian Linux, the simplest approach is to install `sphinx`:

```
pip install sphinx
```

All use of the minted style requires the `-shell-escape` command-line argument when running LaTeX, i.e., `pdflatex -shell-escape`.

Inline comments apply the `todonotes` LaTeX package if the option `--latex_todonotes` is given. The `todonotes` package requires several other packages: `xcolor`, `ifthen`, `xkeyval`, `tikz`, `calc`, `graphicx`, and `setspace`. The relevant Debian packages for installing all this are listed below.

LaTeX packages. Many LaTeX packages are potentially needed, depending on various constructions in the text and command-line options used when compiling DocOnce to LaTeX. The standard packages always required are `relsize`, `makeidx`, `setspace`, `color`, `amsmath`, `amsfonts`, `xcolor`, `bm`, `microtype`, `inputenc`, and `hyperref`. Optional packages that might be included in the `.tex` output are `minted`, `listings`, `fancyvrb`, `xunicode`, `inputenc`, `helvet`, `mathpazo`, `wrapfig`, `calc`, `ifthen`, `xkeyval`, `tikz`, `graphicx`, `setspace`, `shadow`, `disable`, `todonotes`, `lineno`, `xr`, `framed`, `mdframe`, `movie15`, `a4paper`, and `a6paper`.

Relevant Debian packages that gives you all of these LaTeX packages are

```
texlive
texlive-extra-utils
texlive-latex-extra
texlive-font-utils
```

Alternatively, one may pull in `texlive-full` to get all available style files.

If you want to use the *anslistings* code environment with `ptex2tex` (`.ptex2tex.cfg` styles `Python_ANS`, `Python_ANSt`, `Cpp_ANS`, etc.) or `doconce ptex2tex` (`envir=ans` or `envir=ans:nt`), you need the `anslistings.sty` file. It can be obtained from the ptex2tex source. The same code style is in "modern DocOnce" just implemented by the command-line option

```
"--latex_code_style=default:lst[style=yellow2_fb]"
```

Sphinx or reStructuredText Output. Output to `sphinx` or `rst` requires the Sphinx software, installed by

```
pip install sphinx --upgrade
```

DocOnce comes with many Sphinx themes that are not part of the standard Sphinx source distribution:

- cloud and redcloud: `https://bitbucket.org/ecollins/cloud_sptheme`
- bootstrap: `https://github.com/ryan-roemer/sphinx-bootstrap-theme`
- solarized: `https://bitbucket.org/miiton/sphinxjp.themes.solarized`
- impressjs: `https://github.com/shkumagai/sphinxjp.themes.impressjs`
- sagecellserver: `https://github.com/kriskda/sphinx-sagecell`

Appropriate installation commands for these themes are

```
pip install -e hg+https://bitbucket.org/ecollins/cloud_sptheme#egg=cloud_sptheme
pip install -e git+https://github.com/ryan-roemer/sphinx-bootstrap-theme#egg=sphinx-bootstrap-theme
pip install -e hg+https://bitbucket.org/miiton/sphinxjp.themes.solarized#egg=sphinxjp.themes.solarize
pip install -e git+https://github.com/shkumagai/sphinxjp.themes.impressjs#egg=sphinxjp.themes.impress
pip install -e git+https://github.com/kriskda/sphinx-sagecell#egg=sphinx-sagecell
```

It can also be handy to have special typesetting of IPython sessions:

```
pip install -e git+https://bitbucket.org/hplbit/pygments-ipython-console#egg=pygments-ipython-console
```

To make OpenOffice or LibreOffice documents from `rst` output, you will need more software, typically the following on a Debian system:

```
sudo apt-get install unovonv libreoffice libreoffice-dmaths
```

Markdown and Pandoc Output. The DocOnce format `pandoc` outputs the document in various Markdown versions: the Pandoc extended Markdown format (which via the `pandoc` program can be translated to a range of other formats), strict Markdown, and GitHub-flavored Markdown. Installation of Pandoc, written in Haskell, is most easily done by

```
sudo apt-get install pandoc
```

on Debian (Ubuntu) systems.

Epydoc Output. When the output format is `epydoc` one needs that program too, installed by

```
svn co https://epydoc.svn.sourceforge.net/svnroot/epydoc/trunk/epydoc epydoc
cd epydoc
sudo make install
cd ..
```

Remark. Several of the packages above installed from source code are also available in Debian-based system through the `apt-get install` command. However, we recommend installation directly from the version control system repository as there might be important updates and bug fixes. For `svn` directories, go to the directory, run `svn update`, and then `sudo python setup.py install`. For Mercurial (`hg`) directories, go to the directory, run `hg pull; hg update`, and then `sudo python setup.py install`.

Analyzing file differences. The `doconce diff file1 file prog` command for illustrating differences between two files `file1` and `file2` using the program `prog` requires `prog` to be installed. By default, `prog` is `difflib` which comes with Python and is always present if you have DocOnce installed. Another choice, `diff`, should be available on all Unix/Linux systems. Other choices, their URL, and their `sudo apt-get install` command on Debian (Ubuntu) systems appear in the table below.

Program	URL	Debian/Ubuntu install
`pdiff`	a2ps wdiff	`sudo apt-get install a2ps wdiff texlive-latex-extra texlive-lat`
`latexdiff`	latexdiff	`sudo apt-get install latexdiff`
`kdiff3`	kdiff3	`sudo apt-get install kdiff3`
`diffuse`	diffuse	`sudo apt-get install diffuse`
`xxdiff`	xxdiff	`sudo apt-get install xxdiff`
`meld`	meld	`sudo apt-get install meld`
`tkdiff.tcl`	tkdiff	not in Debian