The ccool package[*]

Erwann Rogard[†]

Released 2020/04/12

Abstract

ccool stands for Custom COntent Oriented for LaTeX, a concept pioneered by cool[1][1]. This is done using a minimalist interface built upon xparse[4]. Specifically, `\Ccool<`⟨*name*⟩`>` begins a series of instructions alternating between 'text' and macro definitions, that themselves optionally expand using predefined or inline rules. For example,

```
\Ccool<Math>[Let~]
i{\mathbb{#1}}{ Nat = N, Real = R }*s{{~\rm{and}~}}
[~denote the natural and real numbers.]{}
```

expands to: "Let $\mathbb{N}$ and $\mathbb{R}$ denote the natural and real numbers." As a side effect, `$\Nat<Math>$` encodes "$\mathbb{N}$" (and likewise for `\Real`). `Math` being the default for ⟨*name*⟩, `<Math>` can be dropped. Optionally, the macros can be written to a file, and read, which can be useful for typesetting documents sharing the same notation.

Contents

[*]This file describes version v1.8, last revised 2020/04/12.
[†]firstname dot lastname AusTria gmail dot com
[1]Whereas cool provided predefined macros, ccool is tool for making macros, hence "custom".

Part I
Usage

Convention

1. Loosely, those of [2] and [4], for example as to the meaning of ⟨*token list*⟩.
2. If unspecified, the environment in which a macro must be declared is `document`.

`\usepackage`

```
\usepackage{ccool}
```

Requirement

1. `ccool.sty` is in the path of the LaTeX engine. See Part III, section 4.
2. Declare it in the *preamble*

`\Ccool`

```
\Ccool<⟨tl₁⟩>
[⟨tl₂⟩]
i{⟨code₁⟩}
{⟨kvl₁⟩}
+
*
s{{⟨tl₃⟩}|{⟨tl₃⟩}{⟨tl₄⟩}|{⟨tl₃⟩}{⟨tl₄⟩}{⟨tl₅⟩}}
o{⟨code₂⟩}
[⟨tl₆⟩]
```

Requirement ⟨*kvl*$_1$⟩ is specified (all others optional).

⟨`tl`$_1$⟩

Example `Math`, `ModelA`, `ModelB`

Semantics Identifies a group of macros

⟨`tl`$_2$⟩

Example `Let~`

Semantics Expands ⟨*tl*$_2$⟩

⟨*`code`*$_1$⟩

Example `\mathbb{#1}`

Semantics

1. $\langle val_i \rangle \leftarrow \langle code_1 \rangle$ applied to $\langle val_i \rangle$

$\langle$`kvl`$_1\rangle$

Example `Elems={\omega_1, \dots, \omega_n}, Sample=\Omega`

Semantics

2. `\`$\langle$`key`$_i\rangle$`<`$\langle tl_1\rangle$`>` $\leftarrow \langle val_i \rangle$ defined in step 1, using `Expans` for expansion.
3. If `Write`, writes the input used by step 2 to `File`

`+`

Other Needed to make `\Ccool`'s side effect within a *local group* persist thereafter

Semantics Appends step 2 and step 3 to `\CcoolHook`

`*`

Semantics

4. Expands $\langle code_2\rangle$ applied to the list created in step 1, using the separator specified by $\langle tl_3\rangle$, $\langle tl_4\rangle$, $\langle tl_5\rangle$.

$\langle$`tl`$_3\rangle$

Example `{~\in~}`

$\langle$`tl`$_4\rangle$

Example `{,~}`

$\langle$`tl`$_5\rangle$

Example `{~\&~}`

$\langle$`code`$_2\rangle$

Example `$\left\{#1\right\}$`

$\langle$`tl`$_6\rangle$

Semantics `\Ccool<`$\langle tl_1\rangle$`>[`$\langle tl_6\rangle$`]`

`\CcoolClear` `\CcoolClear<`$\langle$`keyval list`$\rangle$`>`

Semantics Clears any data created by `\Ccool{`$\langle tl_1\rangle$`}`, for all $\langle tl_1\rangle$ in $\langle$*keyval list*$\rangle$

`\CcoolHook` `\CcoolHook`

Example `\AfterEndEnvironment{theorem}{\CcoolHook}`

`\CcoolLambda` `\CcoolLambda[⟨integer⟩]{⟨code⟩}`

Example `\Ccool{ EvalAt = \CcoolLambda{(#1)} }`

Semantics Creates a lambda expression with ⟨*integer*⟩ arguments for ⟨*code*⟩

`\CcoolOption` `\CcoolOption{⟨kvl0⟩}`

Semantics Set default options for `\Ccool`

`Expans`

Default `xo`

Syntax Either of `eo, ee, ex, xe, xo, xe, xx`

`File`

Default `ccool\pdfcreationdate`

Syntax Expands to a valid *path*

`Inner`

Default `####1`

Semantics Default for ⟨$code_1$⟩

Syntax Use `####1` as the argument to be replaced

`Name`

Default `Math`

Semantics Default for ⟨tl_1⟩

`Outer`

Default `\ensuremath{####1}`

Semantics Default for ⟨$code_2$⟩

Syntax Use `####1` as the argument to be replaced

`Separ`

Default `{ {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } }`

Semantics Default for separators' parameter

Syntax That of 'separators' in [2, Section 8 of l3seq]

`Write`

Default `\BooleanFalse`

Syntax *Boolean*

`\CcoolRead` `\CcoolRead[⟨path⟩]`

Other The default for ⟨*path*⟩ is the last write-file (see ⟨kvl_1⟩)

Semantics

1. Reads the definitions in ⟨*path*⟩.
2. Writes to `ccool.log`: 'read from ⟨*path*⟩'

`\CcoolVers` `\CcoolVers`

Semantics Expands to the package's version

Do's and dont's

1.

   Don't: `\Ccool{ A = a, B = b }[Hello, world!]`.

   Do: `\Ccool{ A = a, B = b }[Hello, world!]{}`, or
   `\Ccool{ A = a, B = b } Hello, world!`

2.

   Don't: `$\⟨key_i⟩<x$`.

   Do: `$\⟨key_i⟩{<}x$`

3.

   Don't: `[a, b)`

   Do: `{[}a, b{)}`

4.

   Don't: `\Ccool{ F = \cal F }`.

   Do: `\Ccool{ F = \cal{F} }` or `\Ccool{ F = \mathcal{F} }`

5. Also see Part III, section 3

Part II
Listing

Listing 1.

```
%     \CcoolVers
%
```

2020/04/12 v1.8 cool — A tool for encoding mathematical notation

Listing 2. Preamble[a]

[a]These are the settings to replicate the listings. For exhaustivity, check the `documentation` section of `ccool.dtx`.

```
%     \usepackage{amsmath, amsthm, commath}
%     \usepackage[T1]{fontenc}% \char`[
%
```

Listing 3. Separators

```
%     \CcoolOption{
%     ^^A% spaces betw. inner and outer brackets matter!->
%     Separ={{\ \char`@\ }{\ \%\ }{\ \char`@\ }}}
%     \Ccool<Test>{ X = x, Y = y }*[\\]
%     { X = x, Y = y, Z = z }*[\\]
%     { X = x, Y = y }*s{{\ \&\ }}[\\]
%     { X = x, Y = y }*s{{\ \&\ }{,\ }}[\\]
%     { X = x, Y = y, Z = z }*s{{\ \&\ }}[\\]
%     { X = x, Y = y, Z = z }*s{{\ \&\ }{,\ }}[\\]
%     { X = x, Y = y, Z = z }*s{{\ \&\ }{,\ }{\ \&\ }}\\
%
```

x @ y
x % y @ z
x & y
x & y
x & y & z
$x,\ y$ & z
$x,\ y$ & z

Listing 4. Hello, world![a]

[a]If this looks arcane, it's for the purpose of testing.

```
%     \CcoolOption{ Separ = {{}{.}{.}}, Outer = {####1} }
%     \CcoolOption{ Write = \BooleanTrue }
%     \Ccool<Test>
%     { KeyA = {.}, KeyB = {!}, KeyC = {\%} }[]
%     { KeyD = {d}, KeyE = {\%} }[]i{\{#1\}}
%     { KeyF = {H}, KeyG = {e}, KeyH = {l} }*[]
%     { KeyI = {\%}, KeyJ = {\%}, KeyK = {\%} }[.\{l\}.\{o\}]
%     { KeyL = {l}, KeyM = {\char`[}, KeyN = {\char`]} }[]
%     { KeyO = {o}, KeyP = {\%}, KeyQ = {\%} }[{,\ }]
%     { KeyR = {w}, KeyS = {o}, KeyT = {r} }*s{{}{}{}}o{{\char`[}#1}[]
%     { KeyU = {\%}, KeyV = {\%}, KeyW = {\%} }[]
%     { KeyX = {\%}, KeyY = {\%}, KeyZ = {\KeyB<Test>} }\nobreak
%     \KeyL<Test>\KeyD<Test>\KeyZ<Test>\KeyN<Test>\\
%     \CcoolOption{ Write = \BooleanFalse }
%
```

{H}.{e}.{l}.{l}.{o}, [world!]

Listing 5. Listing 4 read from file.

```
%     \CcoolRead
%     \KeyF<Test>\KeyA<Test>\nobreak
%     \KeyG<Test>\KeyA<Test>\nobreak
%     \KeyH<Test>\KeyA<Test>\nobreak
%     \KeyH<Test>\KeyA<Test>\nobreak
%     {\{}\nobreak\KeyO<Test>{\}},{\ }\nobreak
%     \KeyM<Test>\KeyR<Test>\nobreak
%     \KeyO<Test>\nobreak
%     \KeyT<Test>\nobreak
%     \KeyL<Test>\nobreak
%     \KeyD<Test>\nobreak
%     \KeyZ<Test>\nobreak
%     \KeyN<Test>\nobreak
%
```

{H}.{e}.{l}.{l}.{o}, [world!]

Listing 6. Probability space

```
%     \CcoolOption{ Write = \BooleanTrue }
%     \Ccool[Let~]
%     { Space = \Omega, Field = \mathcal{F}, Meas = \mathcal{P} }
%     *s{{,}}o{$\{#1\}$}
%     [~denote the probability space, where~]{ PowerSet = { 2^{\Space} } }
%     [$\Field\subset \PowerSet$.]
%     {}
%     \CcoolOption{ Write = \BooleanFalse }
%
```

Let $\{\Omega, \mathcal{F}, \mathcal{P}\}$ denote the probability space, where $\mathcal{F} \subset 2^{\Omega}$.

Listing 7. Listing 6 read from file.

```
%     \CcoolRead \tab $\Omega$ $\Field$ $\Meas$
%
```

Ω $\mathcal{F}$ $\mathcal{P}$

Listing 8. Mittelwertsatz für n Variable[3, 17.3]

```
%     \CcoolOption{ Write = \BooleanTrue }
%     \newtheorem{theorem}{Theorem}
%     \AfterEndEnvironment{theorem}{\CcoolHook}
%     \Ccool i{\mathbb{#1}}
%     { N = { N } , R = { R } }+[]
%     { Grad = { \operatorname{grad} } }+
%     [\begin{theorem}
%       [Mittelwertsatz f\"ur $n$ Variable]Es~sei~]
%       { OffMenge = {D}, Ci = {C^{1}}, Strecke = { [x_0,x] } }+
%       [$n\in\N$,~$\OffMenge\subseteq\N^n$ eine offene Menge und
    $f\in\Ci(\OffMenge,\R)$.
%       Dann gibt es auf jeder Strecke $\Strecke\subset\OffMenge$ einen
    Punkt $\xi\in\Strecke$,~]
%       { Steig = { \frac{ f(x)-f(x_0) }{ x-x_0 } }, Punkt = { \xi } }+
%       [so dass gilt
%       \begin{equation*}
%         \Steig = \Grad f(\Punkt)^{\top}
%       \end{equation*}
%     \end{theorem}]
%     {}
%     (Check: $\N$, $\Punkt$)
%     \CcoolOption{ Write = \BooleanFalse }
%
```

Theorem 1 (Mittelwertsatz für n Variable) *Es sei $n \in \mathbb{N}$, $D \subseteq \mathbb{N}^n$ eine offene Menge und $f \in C^1(D, \mathbb{R})$. Dann gibt es auf jeder Strecke $[x_0, x] \subset D$ einen Punkt $\xi \in [x_0, x]$, so dass gilt*

$$\frac{f(x) - f(x_0)}{x - x_0} = \operatorname{grad} f(\xi)^{\top}$$

(Check: $\mathbb{N}$, ξ)

Listing 9. Listing 8 read from file.

```
%     \CcoolRead \tab $\N$ $\R$ $\OffMenge$ $\Ci$ $\Strecke$
%
```

$\mathbb{N}$ $\mathbb{R}$ D C^1 $[x_0, x]$

Listing 10. Lambda expression.

```
%     \CcoolOption{ Write = \BooleanTrue }
%     \Ccool{ EvalAt = \CcoolLambda{(#1)}, ApplyOp =
    \CcoolLambda[2]{#1[#2]} }
%     [Supposons une fonction $f\EvalAt{t}$, et \'etudions le probl\`eme
    o\`u la fonctionnelle $\ApplyOp{S}{f}$ est donn\'ee par\dots]{}
%     \CcoolOption{ Write = \BooleanFalse }
%
```

Supposons une fonction $f(t)$, et étudions le problème où la fonctionnelle $S[f]$ est donnée par…

Listing 11. Listing 10 read from file.

```
%     \CcoolRead \tab $f\EvalAt{t}$, $\ApplyOp{S}{f}$
%
```

$f(t)$, $S[f]$

Listing 12. CUSUM statistic[5]

```
%     \newtheorem{definition}{Definition}
%     \AfterEndEnvironment{definition}{\CcoolHook}
%
%     \CcoolOption{ Write = \BooleanTrue }
%     \Ccool{ SuchThat = { ;~ }, Time = { t }, Process = { \xi }, StopT =
    { T }, EvalAt = \CcoolLambda{(#1)}  }
%     [The CUSUM statistic process and the corresponding one-sided CUSUM
    stopping time are defined as follows:
%     \begin{definition}\label{the CUSUM statistic}. Let~]
%       { Scale = { \lambda }, Real = {\mathcal{R}} }+*s{{~\in~}}[~and~]
%       { CUSUMthresh = { \nu } }+*o{$#1\in\Real^{+}$.}
%       [~Define the following processes:]
%       { LogWald = { u },  CUSUMst = { \StopT_{c} }, CUSUM = { y },
    LogWaldInf = { m } }+
%       [\begin{enumerate}
%       \item{$\LogWald_{\Time}\EvalAt{ \Scale } = \Scale\Process_{\Time}
    - \frac{1}{2}\Scale^2\Time$;
%         $\LogWaldInf_{\Time}\EvalAt{ \Scale } = \inf_{ 0\le s \le \Time
    }\CUSUM_{s} \EvalAt{ \Scale }$.}
%       \item{$\CUSUM_{\Time}\EvalAt{ \Scale } =
    \LogWaldInf_{\Time}\EvalAt{ \Scale } - \LogWald_{\Time}\EvalAt{
    \Scale }\ge0$, which is the CUSUM statistic process.}
%       \item{$\CUSUMst \EvalAt{ \Scale, \LogWaldInf } = \inf\left[ \Time
    \ge 0 \SuchThat \CUSUM_{\Time}\EvalAt{\Scale} \ge \LogWaldInf
    \right]$, which is the CUSUM stopping time.}
    \end{enumerate}\end{definition}\par]{}
%
%     (Check: $\Scale$, $\CUSUM$)
%     \CcoolOption{ Write = \BooleanFalse }
```

```
%
```

The CUSUM statistic process and the corresponding one-sided CUSUM stopping time are defined as follows:

Definition 1 . *Let* $\lambda \in \mathcal{R}$ *and* $\nu \in \mathcal{R}^+$. *Define the following processes:*

1. $u_t(\lambda) = \lambda\xi_t - \frac{1}{2}\lambda^2 t$; $m_t(\lambda) = \inf_{0\leq s\leq t} y_s(\lambda)$.
2. $y_t(\lambda) = m_t(\lambda) - u_t(\lambda) \geq 0$, *which is the CUSUM statistic process.*
3. $T_c(\lambda, m) = \inf\,[t \geq 0; \, y_t(\lambda) \geq m]$, *which is the CUSUM stopping time.*

(Check: λ, y)

Listing 13. Listing 12 read from file.

```
%     \CcoolRead \tab $\Time $ $\Process$ $\Scale$ $\Real$ $\CUSUMthresh$
    $\LogWald$  $\CUSUMst$ $\CUSUM$ $\LogWaldInf$
%
```

$t\ \xi\ \lambda\ \mathcal{R}\ \nu\ u\ T_c\ y\ m$

Part III
Other

1 Acknowledgment

This work has benefited from Q&A's from the LaTeX community[6]. Specific attributions are made in the implementation and References.

2 Install

Compiling `ccool.dtx`[2] will generate `ccool.sty` and `ccool.pdf`

3 Issue

1. **Don't:** `Inner={\{####1\}`

   Symptom: `\CcoolRead` fails

   Do: `Inner={\char`{####1\char`}}`

4 Support

This package is available from `https://www.ctan.org/pkg/ccool` and `https://github.com/rogard/ccool`.

5 Testing

5.1 Technicality

Not possible to compile-check the expansion of a certain class of macros against predefined values[8]. Instead, one can visually check Part II, as generated in section 2 on one's own machine, against that of the repository for the same version.

5.2 Platform

1.
```
Linux laptop 4.15.0-20-generic #21-Ubuntu SMP Tue Apr 24
↪  06:16:15 UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
```

5.3 Engine

1.
```
pdfTeX 3.14159265-2.6-1.40.20 (TeX Live 2019)
```

5.4 Results

1. ccool v1.8 satisfactory on platform 1 and engine 1

[2]Under Unix, `$tex ccool.dtx`

5.5 Other

Check [5] for using ccool with llncs

References

[1] Nick Setzer *The cool package*, 2005, `https://www.ctan.org/pkg/cool`

[2] The LaTeX3 Project Team *The LaTeX3 interfaces*, 2019, `http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3kernel/interface3.pdf`

[3] Thomas F. Sturm *The tcolorbox package*, 2019, `http://www.texdoc.net/texmf-dist/doc/latex/tcolorbox/tcolorbox.pdf`

[4] The LaTeX3 Project Team *The xparse package*, 2020, `http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3packages/xparse.pdf`

[5] Erwann Rogard and Olympia Hadjiliadis *Typesetting a math thesis with ccool*, 2020, `https://github.com/rogard/ccool/blob/master/thesis.pdf`

[6] `https://tex.stackexchange.com/users/112708/erwann?tab=questions`

[7] @sean-allred's answer to "How to create lambda expressions?", `https://tex.stackexchange.com/a/188053/112708`

[8] "Checking a function's expansion against a string", `https://tex.stackexchange.com/a/534100`

[9] @frougon's answer to "Journaling calls to a function []", `https://tex.stackexchange.com/a/536620`

Change History

v1.0
- General: Initial version . . . 14

v1.1
- General: Added: `Save` . . . 14
  - Added: Listing 1., 2., 3., 4., 6., and 9. . . . 14
  - Added:`\OopsRestore` . . . 14
  - Added:`\OopsTest` . . . 14
  - Deleted: Listing 1-5 from v1.0 . . . 14
  - Fixed: apparent anomaly in v1.0's Listing 4, see Listing 3 . . . 14
  - Replaced: `\OopsOptions` by `\OopsOption` . . 14
  - Replaced: `{`$\langle kvl_2\rangle$`}` by <kvl_2> given that option type `G` not recommended[4] 14
  - Replaced: `GenericObject` by `Name` 14
  - Replaced: `Separators` by `Separ` . . 14
  - Revamped: much of the implementation . . . 14

v1.2
- General: Deleted: `\OopsTest` . . . 14
  - Deleted: $\langle kvl_2\rangle$ and $\langle code_2\rangle$ . . . 14
  - Deleted: Listing 2-3 from v1.1. . . . 14
  - Replaced: `\OopsClear{`$\langle tl_1\rangle$`}` by `\OopsClear[`$\langle keyval\ list\rangle$`]` . . . 14
  - Replaced: `\Restore` by `\Read` . . . 14
  - Replaced: `\Save` by `\Write` . . . 14

v1.3
- General: Replaced: `\OopsNew` by `\Oops` 14
  - Replaced: `{`$\langle tl_1\rangle$`}` and `[`$\langle tl_1\rangle$`]` by `<`$\langle tl_1\rangle$`>` . . . 14

v1.4

General: Added: section 9 14

Added: \OopsDebug 14

Added: \OopsHook 14

Added: Expans (for debugging' sake, but...) 14

Added: Listing 1., 2., and 3. 14

Deleted: Listing 1., and 2. 14

Replaced: s{{⟨tl_3⟩}{⟨tl_4⟩}{⟨tl_5⟩}} by s{{⟨tl_3⟩}|{⟨tl_3⟩}{⟨tl_4⟩}|{⟨tl_3⟩}{⟨tl_4⟩}{⟨tl_5⟩}}} 14

v1.5

General: Added: File 14

Deleted: dependence on datetime . 14

v1.6

General: Added: Listing 2 (preamble) 14

Renamed: \OopsClear to \CcoolClear 14

Renamed: \OopsDebug to \CcoolDebug 14

Renamed: \OopsHook to \CcoolHook 14

Renamed: \OopsOption to \CcoolOption 14

Renamed: \OopsRead to \CcoolRead 14

Renamed: \Oops to \Ccool 14

Renamed: oops to ccool (better describes the purpose) 14

v1.7

General: Added: Legends to listings . 14

Added: Listing 12 (CUSUM) 14

Deleted: \CcoolDebug 14

Deleted: Listing 5 from v1.6 14

v1.8

General: Added: \CcoolLambda 14

Added: \CcoolVers 14

Added: Listing 10, Listing 11 14

Added: Listing 1 14

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Part IV
Implementation

```
⟨@@=ccool⟩
\NeedsTeXFormat{LaTeX2e}[2019/10/01]
\ExplSyntaxOn
```

1 aux

__ccool_aux_inner_set:n

#1 : ⟨*code*⟩

```
\cs_new_protected:Nn \__ccool_aux_inner_set:n
{
  \cs_gset:Npn \__ccool_aux_inner:n ##1 {#1}
  \cs_generate_variant:Nn \__ccool_aux_inner:n { e }
}
```

(*End definition for* __ccool_aux_inner_set:n.)

__ccool_aux_key:w

#1 : ⟨ *key* ⟩
#2 : ⟨ *value* ⟩

```
\cs_new_protected:Npn \__ccool_aux_key:w #1 = #2 \q_stop
{
  \seq_gput_right:Nx \g__ccool_aux_key_seq { \tl_trim_spaces:n{#1} }
}
```

(*End definition for* __ccool_aux_key:w.)

__ccool_aux_key:n

#1 : ⟨ *key* = *value* ⟩

```
\cs_new_protected:Nn \__ccool_aux_key:n
{
  \__ccool_aux_key:w #1 \q_stop
}
```

(*End definition for* __ccool_aux_key:n.)

__ccool_aux_key:N

#1 : ⟨ *seq* ⟩

```
\cs_new_protected:Nn \__ccool_aux_key:N
{
  \seq_gclear_new:N \g__ccool_aux_key_seq
  \seq_map_function:NN #1 \__ccool_aux_key:n
}
```

(*End definition for* __ccool_aux_key:N.)

__ccool_aux_outer_set:n

#1 : ⟨ *inline code* ⟩

```
\cs_new_protected:Nn \__ccool_aux_outer_set:n
{
  \cs_gset:Npn \__ccool_aux_outer:n ##1 {#1}
}
```

(*End definition for* __ccool_aux_outer_set:n.)

__ccool_aux_prop:nn

```
\prop_new:N \g__ccool_aux_prop
\cs_new_protected:Nn \__ccool_aux_prop:nn
{
  \prop_gput:Nnn \g__ccool_aux_prop{#1}{#2}
}
\cs_generate_variant:Nn \__ccool_aux_prop:nn { eo, ee, ex, xo, xe, xx }
```

(*End definition for* __ccool_aux_prop:nn.)

__ccool_aux_prop:w

#1 : ⟨ *key* ⟩
#2 : ⟨ *value* ⟩

```
\tl_new:N \g__ccool_option_expans_tl
\cs_new_protected:Npn \__ccool_aux_prop:w #1 = #2 \q_stop
{
  \exp_args:Nx
  \use:c{__ccool_aux_prop:\g__ccool_option_expans_tl}
  { \tl_trim_spaces:n{#1} }
  { \__ccool_aux_inner:n{ \tl_trim_spaces:n{#2} } }
}
```

(*End definition for* __ccool_aux_prop:w.)

__ccool_aux_prop:n

#1 : ⟨ *key* = *value* ⟩

```
\cs_new_protected:Nn \__ccool_aux_prop:n
{
  \__ccool_aux_prop:w #1 \q_stop
}
```

(*End definition for* __ccool_aux_prop:n.)

__ccool_aux_prop:N

#1 : ⟨*keyval list*⟩

```
\cs_new_protected:Nn \__ccool_aux_prop:N
{
  \prop_gclear_new:N \g__ccool_aux_prop
  \seq_if_empty:NTF #1
  { \c_empty_tl }
  {
    \seq_map_function:NN #1 \__ccool_aux_prop:n
  }
}
```

(*End definition for* __ccool_aux_prop:N.)

__ccool_aux_separ:nn

#1 : ⟨ *int* ⟩
#2 : ⟨ *tokens* ⟩

```
\cs_new:Nn \__ccool_aux_separ:nn
{
  \int_case:nnTF {#1}
  {
    {1}
    { \prg_replicate:nn{ 3 }{#2} }
    {2}
    {
      { \use_i:nn #2 }
      { \use_ii:nn #2 }
      { \use_i:nn #2 }
    }
    {3}{#2}
  }
  { \c_empty_tl }
  {
    \msg_error:nnnn { __erw }
    { separ }
    { \exp_not:N \__ccool_aux_separ:nn }
    {#2}
  }
}
\cs_generate_variant:Nn \__ccool_aux_separ:nn { e }
```

(*End definition for* `\__ccool_aux_separ:nn`.)

`\__ccool_aux_separ:n`

`#1` : ⟨ *tokens* ⟩

```
\cs_new:Nn \__ccool_aux_separ:n
{
  \__ccool_aux_separ:en{ \tl_count:n{#1} }{#1}
}
```

(*End definition for* `\__ccool_aux_separ:n`.)

`\__ccool_aux_val:Nn`

`#1` : ⟨ *seq* ⟩
`#2` : ⟨ *tl var name* ⟩

```
\cs_new_protected:Nn \__ccool_aux_val:Nn
{
  \seq_gclear_new:N \g__ccool_aux_val_seq
  \__ccool_seq_from_prop:NNn \g__ccool_aux_val_seq #1 { \__ccool_prop_name:n{#2} }
}
```

(*End definition for* `\__ccool_aux_val:Nn`.)

2 lambda

`\__ccool_lambda:nn` [7]

```
\cs_new_protected:Npn \__ccool_lambda:nn #1 #2
{
  \exp_args:NNx
  \DeclareDocumentCommand \__ccool_lambda_expression
  { \prg_replicate:nn { #1 } { m } }
  {#2}
  \__ccool_lambda_expression
}
```

(*End definition for* `\__ccool_lambda:nn`.)

3 log

__ccool_log_close:

```
\iow_new:N \g__ccool_log_iow
\AtEndDocument{\iow_close:N \g__ccool_log_iow}
\bool_set_false:N \g__ccool_log_open_bool
\cs_new_protected:Nn \__ccool_log_close:
{
  \iow_close:N \g__ccool_log_iow
  \bool_gset_false:N \g__ccool_log_open_bool
}
```

(*End definition for* __ccool_log_close:.)

__ccool_log_open:

```
\tl_new:N \g__ccool_log_file_tl
\cs_new_protected:Nn \__ccool_log_open:
{
  \tl_gset:Nx \g__ccool_log_to_tl{\g__ccool_log_file_tl}
  \iow_open:Nn \g__ccool_log_iow {\g__ccool_log_to_tl}
  \bool_gset_true:N \g__ccool_log_open_bool
}
```

(*End definition for* __ccool_log_open:.)

__ccool_log_read:n   #1 : ⟨*path*⟩

```
\cs_new_protected:Nn \__ccool_log_read:n
{
  \file_input:n{#1}
  \tl_log:n{read~from~#1}
}
\cs_generate_variant:Nn \__ccool_log_read:n { e }
```

(*End definition for* __ccool_log_read:n.)

__ccool_log_read:

```
\cs_new_protected:Nn \__ccool_log_read:
{
  \__ccool_log_read:e{\g__ccool_log_to_tl}
}
```

(*End definition for* __ccool_log_read:.)

__ccool_log_write:n

```
\tl_new:N \g__ccool_log_to_tl
\cs_new_protected:Nn \__ccool_log_write:n
{
  \bool_if:nTF{ \g__ccool_log_open_bool }
  {
    \iow_now:Nn \g__ccool_log_iow {#1}
    \tl_log:n{ write~to~#1 }
  }
  { \msg_error:nnn{ __ccool }{ iow }{ \g__ccool_log_iow }  }
}
\cs_generate_variant:Nn \__ccool_log_write:n { e }
```

(*End definition for* __ccool_log_write:n.)

4 make_key

__ccool_make_key:Nn

#1 : ⟨ *token* ⟩
#2 : ⟨ *key* ⟩

```
\cs_new_protected:Nn \__ccool_make_key:Nn
{
  \exp_args:NNx
  \ProvideDocumentCommand{#1}
  { D<>{\g__ccool_option_name_tl} }
  {
    \__ccool_prop_item:nn{##1}{#2}
  }
}
\cs_generate_variant:Nn \__ccool_make_key:Nn {c}
```

(*End definition for* `\__ccool_make_key:Nn`*.*)

__ccool_make_key:n

#1 : ⟨ *key* ⟩

```
\cs_new_protected:Nn \__ccool_make_key:n
{
  \__ccool_make_key:cn{#1}{#1}
}
\cs_generate_variant:Nn \__ccool_make_key:n { e }
```

(*End definition for* `\__ccool_make_key:n`*.*)

__ccool_make_key:N

#1 : ⟨ *seq* ⟩

```
\cs_new_protected:Nn \__ccool_make_key:N
{
  \seq_map_function:NN #1 \__ccool_make_key:e
}
```

(*End definition for* `\__ccool_make_key:N`*.*)

5 make_ccool

__ccool_make_ccool_exp:nnn

```
\cs_new_protected:Nn \__ccool_make_ccool_exp:nnn
{
  \__ccool_aux_val:Nn \g__ccool_aux_key_seq {#1}
  \__ccool_aux_outer_set:n{#3}
  \__ccool_aux_outer:n
  {
    \exp_args:NNf
    \__ccool_seq_use:Nn
    \g__ccool_aux_val_seq
    {#2}
  }
}
```

(*End definition for* `\__ccool_make_ccool_exp:nnn`*.*)

__ccool_make_ccool_key:nnn

```
\cs_new_protected:Nn \__ccool_make_ccool_key:nnn
{
  \__ccool_prop_if_exist:nTF{#1}
  { \c_empty_tl }
  { \__ccool_prop_new:n{#1} }
  \exp_args:No \__ccool_aux_inner_set:n{#2}
  \seq_set_from_clist:Nn \g__ccool_aux_keyval_seq {#3}
  \__ccool_aux_prop:N \g__ccool_aux_keyval_seq
  \__ccool_prop_append:Nn \g__ccool_aux_prop {#1}
  \__ccool_aux_key:N \g__ccool_aux_keyval_seq
  \__ccool_make_key:N \g__ccool_aux_key_seq
}
```

(*End definition for* `\__ccool_make_ccool_key:nnn`*.*)

__ccool_make_ccool_sideeffect:nnn [9]

```
\cs_new_protected:Nn \__ccool_make_ccool_sideeffect:nnn
{
  \__ccool_make_ccool_key:nnn{#1}{#2}{#3}
  \bool_if:nTF{ \g__ccool_log_open_bool }
  {%^^A https://tex.stackexchange.com/questions/536597
    \__ccool_log_write:n
    {
      \begingroup
      \def \__ccool_log_entry { \Ccool<#1>i{#2}{#3} } \expandafter
      \endgroup \__ccool_log_entry
    }
  }{\c_empty_tl}
}
```

(*End definition for* `\__ccool_make_ccool_sideeffect:nnn`*.*)

__ccool_make_ccool:nnnn

#1 : ⟨ *token list* ⟩
#2 : ⟨ seq_1 ⟩
#3 : ⟨ seq_2 ⟩
#4 : ⟨ *prop* ⟩

```
\def\CcoolHook{\c_empty_tl}
\cs_new_protected:Npn \__ccool_make_ccool:nnnn #1 #2 #3 #4
{
  \exp_args:NNx \DeclareDocumentCommand \Ccool
  {%^^A      2    3         4 5  6    7 8              9
    D<>{#1} +o E{ i }{{#2}} m t+ s E{ s o }{{#3}{#4}} +o
  }
  {
    \IfValueT{##2}{##2}
    \__ccool_make_ccool_sideeffect:nnn{##1}{##3}{##4}
    \IfBooleanT{##6}
    {
      \__ccool_make_ccool_exp:nnn{##1}{##7}{##8}
    }
    \bool_if:nTF{##5}
    {
      \gappto{\CcoolHook}
      {
        \__ccool_make_ccool_sideeffect:nnn{##1}{##3}{##4}
      }
    }
    {\c_empty_tl}
    \IfValueT{##9}
    {
      \exp_not:n{ \Ccool<##1>[##9] }
    }
  }
}
```

(*End definition for* `\__ccool_make_ccool:nnnn`.)

6 msg

```
\msg_new:nnn {__ccool}{ generic }{#1}
\msg_new:nnn {__ccool}{ iow }{#1~is~closed~can't~write}
\msg_new:nnn {__ccool}{ keyonly }{#1~does~not~take~values;~keyval~is~#2}
\msg_new:nnn {__ccool}{ keywrong }{#1~does~not~recognize~key~#2}
\msg_new:nnn {__ccool}{ separ }{#1~expects~1~to~3~items,~#2}
\msg_new:nnn {__ccool}{ unset }{#1~unset}
```

7 option

`\__ccool_aux_inner:n` #1 : ⟨*code*⟩

```
\cs_new_protected:Nn \__ccool_option_inner:n
{
  \tl_gset:Nn \g__ccool_option_inner_tl {#1}
}
\__ccool_option_inner:n
{
  \msg_warning:nnn{ __ccool }{ unset }{ \exp_not:N \g__ccool_option_inner_tl }
}
```

(*End definition for* `\__ccool_aux_inner:n`.)

`\__ccool_option_name:n` #1 : ⟨*token list*⟩

```
\cs_new:Nn \__ccool_option_name:n
{
  \tl_gset:Nn \g__ccool_option_name_tl{#1}
}
\__ccool_option_name:n
{
  \msg_error:nnx{ __ccool }
  { generic }
  { \exp_not:N\g__ccool_option_name_tl~undefined }
}
```

(*End definition for* `\__ccool_option_name:n`.)

__ccool_option_outer:n

#1 : ⟨ *inline code* ⟩

```
\cs_new_protected:Nn \__ccool_option_outer:n
{
  \tl_gset:Nn \g__ccool_option_outer_tl {#1}
}
\__ccool_option_outer:n
{
  \msg_warning:nnn{ __ccool }{ unset }{ \exp_not:N \g__ccool_option_outer_tl }
}
```

(*End definition for* `\__ccool_option_outer:n`.)

__ccool_option_separ:n

#1 : {⟨ tl_1 ⟩}{⟨ tl_2 ⟩}{⟨ tl_3 ⟩}

```
\cs_new_protected:Nn \__ccool_option_separ:n
{
  \cs_gset:Npn \g__ccool_option_separ_tl {#1}
}
\__ccool_option_separ:n
{
  \msg_warning:nnn{ __ccool }{ unset }{ \exp_not:N \g__ccool_option_separ_tl }
}
```

(*End definition for* `\__ccool_option_separ:n`.)

8 prop

__ccool_prop_append:NN

#1 : ⟨ $prop_1$ ⟩
#2 : ⟨ $prop_2$ ⟩

```
\cs_new_protected:Npn \__ccool_prop_append:NN #1 #2
{
  \cs_set:Nn \__ccool_prop_append:nn
  {
    \prop_gput:Nnx #1 {##1}{ \prop_item:Nn #2{##1} }
  }
  \prop_map_function:NN #2 \__ccool_prop_append:nn
}
\cs_generate_variant:Nn \__ccool_prop_append:NN { cN }
```

(*End definition for* `\__ccool_prop_append:NN`.)

__ccool_prop_append:Nn

#1 : ⟨ *prop* ⟩
#2 : ⟨ *tl var name* ⟩

```
\cs_new_protected:Nn \__ccool_prop_append:Nn
{
  \__ccool_prop_append:cN{ \__ccool_prop_name:n {#2} } #1
}
```

(*End definition for* `\__ccool_prop_append:Nn`.)

__ccool_prop_clear_new:n

#1 : ⟨ *tl var name* ⟩

```
\cs_new_protected:Nn \__ccool_prop_clear_new:n
{
  \exp_args:No \prop_clear_new:c{ \__ccool_prop_name:n {#1} }
}
```

(*End definition for* `\__ccool_prop_clear_new:n`.)

`\__ccool_prop_clear_new_map:n` #1 : ⟨ *keyval list* ⟩

```
\cs_new_protected:Nn \__ccool_prop_clear_new_map:n
{
  \seq_set_from_clist:Nn \g__ccool_aux_key_seq {#1}
  \seq_map_function:NN \g__ccool_aux_key_seq \__ccool_prop_clear_new:n
}
```

(*End definition for* `\__ccool_prop_clear_new_map:n`.)

`\__ccool_prop_if_exist:nTF` #1 : ⟨tl_1⟩
#2 : ⟨tl_2⟩
#3 : ⟨tl_3⟩

```
\cs_new:Nn \__ccool_prop_if_exist:nTF
{
  \prop_if_exist:cTF{ \__ccool_prop_name:n {#1} }{#2}{#3}
}
```

(*End definition for* `\__ccool_prop_if_exist:nTF`.)

`\__ccool_prop_item:nn` #1 : ⟨ *tl var name* ⟩
#2 : ⟨ *key* ⟩

```
\cs_new:Nn \__ccool_prop_item:nn
{
  \prop_item:cn { \__ccool_prop_name:n {#1} } {#2}
}
```

(*End definition for* `\__ccool_prop_item:nn`.)

`\__ccool_prop_name:n` #1 : ⟨ *tl var name* ⟩

```
\cs_new:Npn \__ccool_prop_name:n #1{ __ccool_#1 }
```

(*End definition for* `\__ccool_prop_name:n`.)

`\__ccool_prop_new:n` #1 : ⟨ *tl var name* ⟩

```
\cs_new_protected:Nn \__ccool_prop_new:n
{
  \prop_new:c{ \__ccool_prop_name:n {#1} }
}
```

(*End definition for* `\__ccool_prop_new:n`.)

9 seq

__ccool_seq_from_prop:NNn

#1 : ⟨ *seq*$_1$ ⟩
#2 : ⟨ *seq*$_2$ ⟩ (keys)
#3 : ⟨ *prop* ⟩

```
\cs_new_protected:Nn \__ccool_seq_from_prop:NNn
{
  \cs_set_protected:Nn \__ccool_seq_from_prop:n
  {
    \seq_gput_right:No #1 { \prop_item:cn{#3}{##1} }
  }
  \seq_map_function:NN #2 \__ccool_seq_from_prop:n
}
```

(*End definition for* `\__ccool_seq_from_prop:NNn`.)

__ccool_erw_seq_use:Nn

```
%      \begin{arguments}
%      \item \meta{ seq }
%      \item \meta{ tokens }
%      \end{arguments}
\cs_new:Nn \__ccool_seq_use:Nn
{
  \exp_last_unbraced:NNf
  \seq_use:Nnnn #1
  \__ccool_aux_separ:n{#2}
}
```

(*End definition for* `\__ccool_erw_seq_use:Nn`.)

10 Front-end

```
\keys_define:nn { __ccool }
{
  Expans .multichoices:nn =
  { eo, ee, ex, xo, xe, xx }
  { \tl_gset_eq:NN \g__ccool_option_expans_tl \l_keys_choice_tl },
  Expans .default:n = { xo },
  Expans .initial:n = { xo },
  File .code:n = { \tl_gset:Nn  \g__ccool_log_file_tl{ \exp_not:n{ #1 } } },
  File .default:n = { ccool\pdfcreationdate },
  File .initial:n = { ccool\pdfcreationdate },
  Name .code:n={
    \__ccool_option_name:n{#1}
    \exp_last_unbraced:Nf
    \__ccool_make_ccool:nnnn
    {
      { \g__ccool_option_name_tl }
      { \g__ccool_option_inner_tl }
      { \g__ccool_option_separ_tl }
      { \g__ccool_option_outer_tl }
    }
  },
  Name .value_required:n = false,
  Name .default:n = { Math },
  Name .initial:n = { Math },
  Inner .code:n={
    \__ccool_option_inner:n{#1}
    \exp_last_unbraced:Nf
    \__ccool_make_ccool:nnnn
    {
      { \g__ccool_option_name_tl }
      { \g__ccool_option_inner_tl }
      { \g__ccool_option_separ_tl }
      { \g__ccool_option_outer_tl }
    }
  },
  Inner .value_required:n = false,
  Inner .default:n = {####1},
  Inner .initial:n = {####1},
  Outer .code:n={
    \__ccool_option_outer:n{#1}
    \exp_last_unbraced:Nf
    \__ccool_make_ccool:nnnn
    {
      { \g__ccool_option_name_tl }
      { \g__ccool_option_inner_tl }
      { \g__ccool_option_separ_tl }
      { \g__ccool_option_outer_tl }
    }
  },
  Outer .value_required:n = false,
  Outer .default:n = { \ensuremath{####1} },
  Outer .initial:n = { \ensuremath{####1} },
  Write .code:n = {
    \bool_if:nTF{#1}
    {\__ccool_log_open:}
    {\__ccool_log_close:}
  },
  Write .value_required:n = false,
  Write .default:n = \BooleanFalse,
  Write .initial:n = \BooleanFalse,
  Separ .code:n={
    \__ccool_option_separ:n{#1}
    \exp_last_unbraced:Nf
    \__ccool_make_ccool:nnnn
    {
      { \g__ccool_option_name_tl }
      { \g__ccool_option_inner_tl }
      { \g__ccool_option_separ_tl }
      { \g__ccool_option_outer_tl }
    }
  },
  Separ .value_required:n = false,
  Separ .default:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } },
  Separ .initial:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } }
}
```

\CcoolClear #1 : ⟨ *tl var name* ⟩

```
\NewDocumentCommand{ \CcoolClear }
{ D<>{\g__ccool_option_name_tl} }
{
  \__ccool_prop_clear_new_map:n{#1}
}
```

(*End definition for* `\CcoolClear`*. This function is documented on page 5.*)

\CcoolLambda

```
\ProvideDocumentCommand \CcoolLambda { O{1} m }
{
  \__ccool_lambda:nn { #1 } { #2 }
}
```

(*End definition for* `\CcoolLambda`*. This function is documented on page 6.*)

\CcoolOption

```
\NewDocumentCommand{ \CcoolOption }
{ m }
{
  \keys_set:nn{ __ccool }{#1}
}
```

(*End definition for* `\CcoolOption`*. This function is documented on page 6.*)

\CcoolRead

```
\NewDocumentCommand{\CcoolRead}
{o}
{
  \IfValueTF{#1}
  {\__ccool_log_read:e{#1}}
  {\__ccool_log_read:}
}
```

(*End definition for* `\CcoolRead`*. This function is documented on page 7.*)

\CcoolVers

```
\NewDocumentCommand{\CcoolVers}
{}
{\use:c{ver@ccool.sty}}
```

(*End definition for* `\CcoolVers`*. This function is documented on page 7.*)

11 Misc

```
\ExplSyntaxOff
```