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 .
2. Declare it in the *preamble*

\Ccool

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

Requirement $\langle kvl_1 \rangle$ is specified (all others optional).

$\langle tl_1 \rangle$

Example `Math, ModelA, ModelB`

Semantics Identifies a group of macros

$\langle tl_2 \rangle$

Example `Let~`

Semantics Expands $\langle tl_2 \rangle$

$\langle code_1 \rangle$

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. $\backslash\langle key_i \rangle<\langle tl_1 \rangle> \leftarrow \langle val_i \rangle$ 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 ⟨*code₂*⟩ applied to the list created in step 1, using the separator specified by ⟨*tl₃*⟩, ⟨*tl₄*⟩, ⟨*tl₅*⟩.

⟨*tl₃*⟩

Example {~\in~}

⟨*tl₄*⟩

Example {,~}

⟨*tl₅*⟩

Example {~\&~}

⟨*code₂*⟩

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

⟨*tl₆*⟩

Semantics **\Ccool**<⟨*tl₁*⟩>[⟨*tl₆*⟩]

\CcoolClear \CcoolClear<⟨*keyval list*⟩>

Semantics Clears any data created by **\Ccool**{⟨*tl₁*⟩}, for all ⟨*tl₁*⟩ in ⟨*keyval list*⟩

\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 $\langle code_1 \rangle$

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

Name

Default `Math`

Semantics Default for $\langle tl_1 \rangle$

Outer

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

Semantics Default for $\langle code_2 \rangle$

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 $\langle path \rangle$ is the last write-file (see $\langle kvl_1 \rangle$)

Semantics

1. Reads the definitions in $\langle path \rangle$.
2. Writes to `ccool.log`: 'read from $\langle path \rangle$'

`\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 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{{\ \&\ }{,\ }{\ \&\ }}\\
%
```

- -

$x \mathbin{@} y$
$x \% y \mathbin{@} z$
$x \& y$
$x \& y$
$x \& y \& z$
$x, y \& z$
$x, y \& z$

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 $\Omega$ $\Field$ $\Meas$
%
```

$$\Omega \; \mathcal{F} \; \mathcal{P}$$

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 LaTeXcommunity[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] User erwann at tex.stackexchange.com, 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

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.

16

17

Part IV
Implementation

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

1 `aux`

`\__ccool_aux_inner_set:n` #1 : ⟨code⟩

```
4 \cs_new_protected:Nn \__ccool_aux_inner_set:n
5 {
6   \cs_gset:Npn \__ccool_aux_inner:n ##1 {#1}
7   \cs_generate_variant:Nn \__ccool_aux_inner:n { e }
8 }
```

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

`\__ccool_aux_key:w` #1 : ⟨ key ⟩
#2 : ⟨ value ⟩

```
9  \cs_new_protected:Npn \__ccool_aux_key:w #1 = #2 \q_stop
10 {
11   \seq_gput_right:Nx \g__ccool_aux_key_seq { \tl_trim_spaces:n{#1} }
12 }
```

(*End definition for* `\__ccool_aux_key:w`.)

`\__ccool_aux_key:n` #1 : ⟨ key = value ⟩

```
13 \cs_new_protected:Nn \__ccool_aux_key:n
14 {
15   \__ccool_aux_key:w #1 \q_stop
16 }
```

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

`\__ccool_aux_key:N` #1 : ⟨ seq ⟩

```
17 \cs_new_protected:Nn \__ccool_aux_key:N
18 {
19   \seq_gclear_new:N \g__ccool_aux_key_seq
20   \seq_map_function:NN #1 \__ccool_aux_key:n
21 }
```

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

`\__ccool_aux_outer_set:n` #1 : ⟨ inline code ⟩

```
22 \cs_new_protected:Nn \__ccool_aux_outer_set:n
23 {
24   \cs_gset:Npn \__ccool_aux_outer:n ##1 {#1}
25 }
```

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

`\__ccool_aux_prop:nn`

```
26 \prop_new:N \g__ccool_aux_prop
27 \cs_new_protected:Nn \__ccool_aux_prop:nn
28 {
29   \prop_gput:Nnn \g__ccool_aux_prop{#1}{#2}
30 }
31 \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* ⟩

```
32 \tl_new:N \g__ccool_option_expans_tl
33 \cs_new_protected:Npn \__ccool_aux_prop:w #1 = #2 \q_stop
34 {
35   \exp_args:Nx
36   \use:c{__ccool_aux_prop:\g__ccool_option_expans_tl}
37   { \tl_trim_spaces:n{#1} }
38   { \__ccool_aux_inner:n{ \tl_trim_spaces:n{#2} } }
39 }
```

(*End definition for* `\__ccool_aux_prop:w`.)

`\__ccool_aux_prop:n` #1 : ⟨ *key = value* ⟩

```
40 \cs_new_protected:Nn \__ccool_aux_prop:n
41 {
42   \__ccool_aux_prop:w #1 \q_stop
43 }
```

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

`\__ccool_aux_prop:N` #1 : ⟨*keyval list*⟩

```
44 \cs_new_protected:Nn \__ccool_aux_prop:N
45 {
46   \prop_gclear_new:N \g__ccool_aux_prop
47   \seq_if_empty:NTF #1
48   { \c_empty_tl }
49   {
50     \seq_map_function:NN #1 \__ccool_aux_prop:n
51   }
52 }
```

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

`\__ccool_aux_separ:nn` #1 : ⟨ *int* ⟩
#2 : ⟨ *tokens* ⟩

```
53 \cs_new:Nn \__ccool_aux_separ:nn
54 {
55   \int_case:nnTF {#1}
56   {
57     {1}
58     { \prg_replicate:nn{ 3 }{#2} }
59     {2}
60     {
```

```
61        { \use_i:nn #2 }
62        { \use_ii:nn #2 }
63        { \use_i:nn #2 }
64      }
65      {3}{#2}
66    }
67    { \c_empty_tl }
68    {
69      \msg_error:nnnn { __erw }
70      { separ }
71      { \exp_not:N \__ccool_aux_separ:nn }
72      {#2}
73    }
74  }
75  \cs_generate_variant:Nn \__ccool_aux_separ:nn { e }
```

(End definition for __ccool_aux_separ:nn.)

__ccool_aux_separ:n   #1 : ⟨ *tokens* ⟩

```
76  \cs_new:Nn \__ccool_aux_separ:n
77  {
78    \__ccool_aux_separ:en{ \tl_count:n{#1} }{#1}
79  }
```

(End definition for __ccool_aux_separ:n.)

__ccool_aux_val:Nn   #1 : ⟨ *seq* ⟩
#2 : ⟨ *tl var name* ⟩

```
80  \cs_new_protected:Nn \__ccool_aux_val:Nn
81  {
82    \seq_gclear_new:N \g__ccool_aux_val_seq
83    \__ccool_seq_from_prop:NNn \g__ccool_aux_val_seq #1 { \__ccool_prop_name:n{#2} }
84  }
```

(End definition for __ccool_aux_val:Nn.)

2   `lambda`

__ccool_lambda:nn   [7]

```
85  \cs_new_protected:Npn \__ccool_lambda:nn #1 #2
86  {
87    \exp_args:NNx
88    \DeclareDocumentCommand \__ccool_lambda_expression
89    { \prg_replicate:nn { #1 } { m } } }
90    {#2}
91    \__ccool_lambda_expression
92  }
```

(End definition for __ccool_lambda:nn.)

3 log

__ccool_log_close:

```
93 \iow_new:N \g__ccool_log_iow
94 \AtEndDocument{\iow_close:N \g__ccool_log_iow}
95 \bool_set_false:N \g__ccool_log_open_bool
96 \cs_new_protected:Nn \__ccool_log_close:
97 {
98   \iow_close:N \g__ccool_log_iow
99   \bool_gset_false:N \g__ccool_log_open_bool
100 }
```

(*End definition for* __ccool_log_close:.)

__ccool_log_open:

```
101 \tl_new:N \g__ccool_log_file_tl
102 \cs_new_protected:Nn \__ccool_log_open:
103 {
104   \tl_gset:Nx \g__ccool_log_to_tl{\g__ccool_log_file_tl}
105   \iow_open:Nn \g__ccool_log_iow {\g__ccool_log_to_tl}
106   \bool_gset_true:N \g__ccool_log_open_bool
107 }
```

(*End definition for* __ccool_log_open:.)

__ccool_log_read:n **#1** : ⟨*path*⟩

```
108 \cs_new_protected:Nn \__ccool_log_read:n
109 {
110   \file_input:n{#1}
111   \tl_log:n{read~from~#1}
112 }
113 \cs_generate_variant:Nn \__ccool_log_read:n { e }
```

(*End definition for* __ccool_log_read:n.)

__ccool_log_read:

```
114 \cs_new_protected:Nn \__ccool_log_read:
115 {
116   \__ccool_log_read:e{\g__ccool_log_to_tl}
117 }
```

(*End definition for* __ccool_log_read:.)

__ccool_log_write:n

```
118 \tl_new:N \g__ccool_log_to_tl
119 \cs_new_protected:Nn \__ccool_log_write:n
120 {
121   \bool_if:nTF{ \g__ccool_log_open_bool }
122   {
123     \iow_now:Nn \g__ccool_log_iow {#1}
124     \tl_log:n{ write~to~#1 }
125   }
126   { \msg_error:nnn{ __ccool }{ iow }{ \g__ccool_log_iow }  }
127 }
128 \cs_generate_variant:Nn \__ccool_log_write:n { e }
```

(*End definition for* __ccool_log_write:n.)

4 `make_key`

#1 : ⟨ *token* ⟩
#2 : ⟨ *key* ⟩

```
129 \cs_new_protected:Nn \__ccool_make_key:Nn
130 {
131   \exp_args:NNx
132   \ProvideDocumentCommand{#1}
133   { D<>{\g__ccool_option_name_tl} }
134   {
135     \__ccool_prop_item:nn{##1}{#2}
136   }
137 }
138 \cs_generate_variant:Nn \__ccool_make_key:Nn {c}
```

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

#1 : ⟨ *key* ⟩

```
139 \cs_new_protected:Nn \__ccool_make_key:n
140 {
141   \__ccool_make_key:cn{#1}{#1}
142 }
143 \cs_generate_variant:Nn \__ccool_make_key:n { e }
```

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

#1 : ⟨ *seq* ⟩

```
144 \cs_new_protected:Nn \__ccool_make_key:N
145 {
146   \seq_map_function:NN #1 \__ccool_make_key:e
147 }
```

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

5 `make_ccool`

```
148 \cs_new_protected:Nn \__ccool_make_ccool_exp:nnn
149 {
150   \__ccool_aux_val:Nn \g__ccool_aux_key_seq {#1}
151   \__ccool_aux_outer_set:n{#3}
152   \__ccool_aux_outer:n
153   {
154     \exp_args:NNf
155     \__ccool_seq_use:Nn
156     \g__ccool_aux_val_seq
157     {#2}
158   }
159 }
```

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

`\__ccool_make_ccool_key:nnn`

```
160 \cs_new_protected:Nn \__ccool_make_ccool_key:nnn
161 {
162   \__ccool_prop_if_exist:nTF{#1}
163   { \c_empty_tl }
164   { \__ccool_prop_new:n{#1} }
165   \exp_args:No \__ccool_aux_inner_set:n{#2}
166   \seq_set_from_clist:Nn \g__ccool_aux_keyval_seq {#3}
167   \__ccool_aux_prop:N \g__ccool_aux_keyval_seq
168   \__ccool_prop_append:Nn \g__ccool_aux_prop {#1}
169   \__ccool_aux_key:N \g__ccool_aux_keyval_seq
170   \__ccool_make_key:N \g__ccool_aux_key_seq
171 }
```

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

`\__ccool_make_ccool_sideeffect:nnn` [9]

```
172 \cs_new_protected:Nn \__ccool_make_ccool_sideeffect:nnn
173 {
174   \__ccool_make_ccool_key:nnn{#1}{#2}{#3}
175   \bool_if:nTF{ \g__ccool_log_open_bool }
176   {%^^A https://tex.stackexchange.com/questions/536597
177     \__ccool_log_write:n
178     {
179       \begingroup
180       \def \__ccool_log_entry { \Ccool<#1>i{#2}{#3} } \expandafter
181       \endgroup \__ccool_log_entry
182     }
183   }{\c_empty_tl}
184 }
```

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

`\__ccool_make_ccool:nnnn`
#1 : ⟨ *token list* ⟩
#2 : ⟨ *seq₁* ⟩
#3 : ⟨ *seq₂* ⟩
#4 : ⟨ *prop* ⟩

```
185 \def\CcoolHook{\c_empty_tl}
186 \cs_new_protected:Npn \__ccool_make_ccool:nnnn #1 #2 #3 #4
187 {
188   \exp_args:NNx \DeclareDocumentCommand \Ccool
189   {%^^A       2   3           4 5 6     7 8                   9
190     D<>{#1} +o E{ i }{{#2}} m t+ s E{ s o }{{#3}{#4}} +o
191   }
192   {
193     \IfValueT{##2}{##2}
194     \__ccool_make_ccool_sideeffect:nnn{##1}{##3}{##4}
195     \IfBooleanT{##6}
196     {
197       \__ccool_make_ccool_exp:nnn{##1}{##7}{##8}
198     }
199     \bool_if:nTF{##5}
200     {
201       \gappto{\CcoolHook}
```

```
202        {
203          \__ccool_make_ccool_sideeffect:nnn{##1}{##3}{##4}
204        }
205      }
206      {\c_empty_tl}
207      \IfValueT{##9}
208      {
209        \exp_not:n{ \Ccool<##1>[##9] }
210      }
211    }
212  }
```

(End definition for __ccool_make_ccool:nnnn.*)*

6   `msg`

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

7   `option`

__ccool_aux_inner:n   #1 : ⟨*code*⟩

```
219  \cs_new_protected:Nn \__ccool_option_inner:n
220  {
221    \tl_gset:Nn \g__ccool_option_inner_tl {#1}
222  }
223  \__ccool_option_inner:n
224  {
225    \msg_warning:nnn{ __ccool }{ unset }{ \exp_not:N \g__ccool_option_inner_tl }
226  }
```

(End definition for __ccool_aux_inner:n.*)*

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

```
227  \cs_new:Nn \__ccool_option_name:n
228  {
229    \tl_gset:Nn \g__ccool_option_name_tl{#1}
230  }
231  \__ccool_option_name:n
232  {
233    \msg_error:nnx{ __ccool }
234    { generic }
235    { \exp_not:N\g__ccool_option_name_tl~undefined }
236  }
```

(End definition for __ccool_option_name:n.*)*

`\__ccool_option_outer:n`   #1 : ⟨ *inline code* ⟩

```
237 \cs_new_protected:Nn \__ccool_option_outer:n
238 {
239   \tl_gset:Nn \g__ccool_option_outer_tl {#1}
240 }
241 \__ccool_option_outer:n
242 {
243   \msg_warning:nnn{ __ccool }{ unset }{ \exp_not:N \g__ccool_option_outer_tl }
244 }
```

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

`\__ccool_option_separ:n`   #1 : {⟨ *tl₁* ⟩}{⟨ *tl₂* ⟩}{⟨ *tl₃* ⟩}

```
245 \cs_new_protected:Nn \__ccool_option_separ:n
246 {
247   \cs_gset:Npn \g__ccool_option_separ_tl {#1}
248 }
249 \__ccool_option_separ:n
250 {
251   \msg_warning:nnn{ __ccool }{ unset }{ \exp_not:N \g__ccool_option_separ_tl }
252 }
```

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

8 prop

`\__ccool_prop_append:NN`   #1 : ⟨ *prop₁* ⟩
#2 : ⟨ *prop₂* ⟩

```
253 \cs_new_protected:Npn \__ccool_prop_append:NN #1 #2
254 {
255   \cs_set:Nn \__ccool_prop_append:nn
256   {
257     \prop_gput:Nnx #1 {##1}{ \prop_item:Nn #2{##1} }
258   }
259   \prop_map_function:NN #2 \__ccool_prop_append:nn
260 }
261 \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* ⟩

```
262 \cs_new_protected:Nn \__ccool_prop_append:Nn
263 {
264   \__ccool_prop_append:cN{ \__ccool_prop_name:n {#2} } #1
265 }
```

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

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

```
266 \cs_new_protected:Nn \__ccool_prop_clear_new:n
267 {
268   \exp_args:No \prop_clear_new:c{ \__ccool_prop_name:n {#1} }
269 }
```

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

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

```
270 \cs_new_protected:Nn \__ccool_prop_clear_new_map:n
271 {
272   \seq_set_from_clist:Nn \g__ccool_aux_key_seq {#1}
273   \seq_map_function:NN \g__ccool_aux_key_seq \__ccool_prop_clear_new:n
274 }
```

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

`\__ccool_prop_if_exist:nTF`   #1 : ⟨*tl₁*⟩
   #2 : ⟨*tl₂*⟩
   #3 : ⟨*tl₃*⟩

```
275 \cs_new:Nn \__ccool_prop_if_exist:nTF
276 {
277   \prop_if_exist:cTF{ \__ccool_prop_name:n {#1} }{#2}{#3}
278 }
```

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

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

```
279 \cs_new:Nn \__ccool_prop_item:nn
280 {
281   \prop_item:cn { \__ccool_prop_name:n {#1} } {#2}
282 }
```

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

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

```
283 \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* ⟩

```
284 \cs_new_protected:Nn \__ccool_prop_new:n
285 {
286   \prop_new:c{ \__ccool_prop_name:n {#1} }
287 }
```

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

9 `seq`

`\__ccool_seq_from_prop:NNn`

#1 : ⟨ *seq₁* ⟩
#2 : ⟨ *seq₂* ⟩ (keys)
#3 : ⟨ *prop* ⟩

```
288 \cs_new_protected:Nn \__ccool_seq_from_prop:NNn
289 {
290   \cs_set_protected:Nn \__ccool_seq_from_prop:n
291   {
292     \seq_gput_right:No #1 { \prop_item:cn{#3}{##1} }
293   }
294   \seq_map_function:NN #2 \__ccool_seq_from_prop:n
295 }
```

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

`\__ccool_erw_seq_use:Nn`

```
296 %        \begin{arguments}
297 %          \item \meta{ seq }
298 %          \item \meta{ tokens }
299 %        \end{arguments}
300 \cs_new:Nn \__ccool_seq_use:Nn
301 {
302   \exp_last_unbraced:NNf
303   \seq_use:Nnnn #1
304   \__ccool_aux_separ:n{#2}
305 }
```

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

10 Front-end

```
306 \keys_define:nn { __ccool }
307 {
308   Expans .multichoices:nn =
309   { eo, ee, ex, xo, xe, xx }
310   { \tl_gset_eq:NN \g__ccool_option_expans_tl \l_keys_choice_tl },
311   Expans .default:n = { xo },
312   Expans .initial:n = { xo },
313   File .code:n = { \tl_gset:Nn  \g__ccool_log_file_tl{ \exp_not:n{ #1 } } },
314   File .default:n = { ccool\pdfcreationdate },
315   File .initial:n = { ccool\pdfcreationdate },
316   Name .code:n={
317     \__ccool_option_name:n{#1}
318     \exp_last_unbraced:Nf
319     \__ccool_make_ccool:nnnn
320     {
321       { \g__ccool_option_name_tl }
322       { \g__ccool_option_inner_tl }
323       { \g__ccool_option_separ_tl }
324       { \g__ccool_option_outer_tl }
325     }
326   },
```

```
327  Name .value_required:n = false,
328  Name .default:n = { Math },
329  Name .initial:n = { Math },
330  Inner .code:n={
331    \__ccool_option_inner:n{#1}
332    \exp_last_unbraced:Nf
333    \__ccool_make_ccool:nnnn
334    {
335      { \g__ccool_option_name_tl }
336      { \g__ccool_option_inner_tl }
337      { \g__ccool_option_separ_tl }
338      { \g__ccool_option_outer_tl }
339    }
340  },
341  Inner .value_required:n = false,
342  Inner .default:n = {####1},
343  Inner .initial:n = {####1},
344  Outer .code:n={
345    \__ccool_option_outer:n{#1}
346    \exp_last_unbraced:Nf
347    \__ccool_make_ccool:nnnn
348    {
349      { \g__ccool_option_name_tl }
350      { \g__ccool_option_inner_tl }
351      { \g__ccool_option_separ_tl }
352      { \g__ccool_option_outer_tl }
353    }
354  },
355  Outer .value_required:n = false,
356  Outer .default:n = { \ensuremath{####1} },
357  Outer .initial:n = { \ensuremath{####1} },
358  Write .code:n = {
359    \bool_if:nTF{#1}
360    {\__ccool_log_open:}
361    {\__ccool_log_close:}
362  },
363  Write .value_required:n = false,
364  Write .default:n = \BooleanFalse,
365  Write .initial:n = \BooleanFalse,
366  Separ .code:n={
367    \__ccool_option_separ:n{#1}
368    \exp_last_unbraced:Nf
369    \__ccool_make_ccool:nnnn
370    {
371      { \g__ccool_option_name_tl }
372      { \g__ccool_option_inner_tl }
373      { \g__ccool_option_separ_tl }
374      { \g__ccool_option_outer_tl }
375    }
376  },
377  Separ .value_required:n = false,
378  Separ .default:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } },
379  Separ .initial:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } }
380 }
```

\CcoolClear #1 : ⟨ *tl var name* ⟩

```
381 \NewDocumentCommand{ \CcoolClear }
382 { D<>{\g__ccool_option_name_tl} }
383 {
384   \__ccool_prop_clear_new_map:n{#1}
385 }
```

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

\CcoolLambda

```
386 \ProvideDocumentCommand \CcoolLambda { O{1} m }
387 {
388   \__ccool_lambda:nn { #1 } { #2 }
389 }
```

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

\CcoolOption

```
390 \NewDocumentCommand{ \CcoolOption }
391 { m }
392 {
393   \keys_set:nn{ __ccool }{#1}
394 }
```

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

\CcoolRead

```
395 \NewDocumentCommand{\CcoolRead}
396 {o}
397 {
398   \IfValueTF{#1}
399   {\__ccool_log_read:e{#1}}
400   {\__ccool_log_read:}
401 }
```

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

\CcoolVers

```
402 \NewDocumentCommand{\CcoolVers}
403 {}
404 {\use:c{ver@ccool.sty}}
```

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

11 Misc

```
405 \ExplSyntaxOff
```