

# oops, an object oriented practical scribe’s package.\*

Erwann Rogard†

Released 2020/04/10

## Abstract

`oops` is a package for L<sup>A</sup>T<sub>E</sub>X (hence “scribe”) for generating macro definitions as the need arises in the document, and to organize them along two dimensions: functions and objects, hence “OO”. This is done using a minimalist interface built upon `xparse`[4]. Specifically, `\Ops<object>` begins a series of instructions alternating between ‘text’ and definitions, that themselves optionally expand using predefined or inline rules. For example,

```
\Ops<Math>[Let~]{Space=\Omega}*[-denote the sample space]{}
```

expands to: “Let  $\Omega$  denote the sample space”. As a side effect, `\Space<Math>$` encodes “ $\Omega$ ”. `Math` being the default for `<object>`, it can be dropped. `DEBUG` Optionally, the definitions can be written to a file, and restored, which can be useful for typesetting documents sharing the same notational conventions. Altogether, “practical”.

## Contents

<b>I</b>	<b>Usage</b>	<b>3</b>
<b>1</b>	<b>Convention</b>	<b>3</b>
<b>2</b>	<b>Loading the package</b>	<b>3</b>
<b>3</b>	<b>\Ops</b>	<b>3</b>
3.1	<code>&lt;tl<sub>1</sub>&gt;</code> . . . . .	3
3.2	<code>[tl<sub>2</sub>]</code> . . . . .	4
3.3	<code>i{code<sub>1</sub>}</code> . . . . .	4
3.4	<code>{kv<sub>1</sub>}</code> . . . . .	4
3.5	<code>+</code> . . . . .	4
3.6	<code>*</code> . . . . .	4
3.7	<code>s{tl<sub>3</sub>}tl<sub>3</sub>tl<sub>4</sub>tl<sub>3</sub>tl<sub>4</sub>tl<sub>5</sub>}</code> . . . . .	4
3.8	<code>o{code<sub>2</sub>}</code> . . . . .	4
3.9	<code>[tl<sub>6</sub>]</code> . . . . .	4
<b>4</b>	<b>\OpsClear</b>	<b>5</b>

---

\*This file describes version v1.4, last revised 2020/04/10.

†firstname dot lastname AusTria gmail dot com

5	\OpsDebug	5
6	\OpsHook	5
7	\OpsOption	5
	7.10 Expans . . . . .	5
	7.11 Inner . . . . .	5
	7.12 Name . . . . .	5
	7.13 Outer . . . . .	5
	7.14 Separ . . . . .	5
	7.15 Write . . . . .	5
8	\OpsRead	5
8	Do's and dont's	5
<b>II</b>	<b>Listing</b>	<b>7</b>
	Listing 1.	7
	Listing 2.	7
	Listing 3.	8
	Listing 4.	8
	Listing 6.	9
	Listing 7.	9
	Listing 8.	9
<b>III</b>	<b>Other</b>	<b>11</b>
1	Acknowledgment	11
2	Also see	11
3	Install	11
4	Issue	11
5	Support	11
6	Testing	11
<b>IV</b>	<b>Implementation</b>	<b>13</b>
1	aux	13

<b>2</b>	<b>log</b>	<b>15</b>
<b>3</b>	<b>make_key</b>	<b>16</b>
<b>4</b>	<b>make_oops</b>	<b>17</b>
<b>5</b>	<b>msg</b>	<b>19</b>
<b>6</b>	<b>option</b>	<b>19</b>
<b>7</b>	<b>prop</b>	<b>20</b>
<b>8</b>	<b>seq</b>	<b>21</b>
<b>9</b>	<b>Front-end</b>	<b>22</b>
<b>10</b>	<b>Misc</b>	<b>24</b>

## Part I

# Usage

This part describes .

## Convention

1. Loosely, those of [3] and [4], for example as to the meaning of  $\langle token\ list \rangle$ .
2. If unspecified, the environment in which a function must be declared is **document**.
3. Where  $\langle tl_1 \rangle$  is an optional argument, its default is **Math**.

---

<code>\usepackage</code>	<code>\usepackage{oops}</code>
--------------------------	--------------------------------

---

**Environment** Preamble

**Requirement** `oops.sty` is in the path of the L<sup>A</sup>T<sub>E</sub>X engine. See [Part III, section 5](#).

---

<code>\Ooops</code>	$\backslash\mathrm{Ooops}\langle tl_1 \rangle$ $[\langle tl_2 \rangle]$ $\mathrm{i}\{\langle code_1 \rangle\}$ $\{\langle kvl_1 \rangle\}$ $+$ $*$ $\mathrm{s}\{\{\langle tl_3 \rangle\} \{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\} \{\langle tl_3 \rangle\}\{\langle tl_4 \rangle\}\{\langle tl_5 \rangle\}\}$ $\mathrm{o}\{\langle code_2 \rangle\}$ $[\langle tl_6 \rangle]$
---------------------	---

---

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

$\langle tl_1 \rangle$

**Example** Math, ModelA, ModelB

**Semantics** Registers a new object, if applicable

$\langle tl_2 \rangle$

**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** Elms={ $\omega_1$ ,  $\dots$ ,  $\omega_n$ }, Sample= $\Omega$

**Semantics** 2.  $\langle key_i \rangle \langle tl_1 \rangle \leftarrow \langle val_i \rangle$  defined in step **1**.

3. If Write= $\text{BooleanTrue}$ , writes the definitions made in step **2** to file  $\text{oops}\langle digits \rangle.\text{tex}$ , where  $\langle digits \rangle = \text{pdfdate}$

+

**Other** Needed to make the side effect  $\backslash\text{Ops}$ ' within a local group persist thereafter

**Semantics** Appends step **2** and step **3** to  $\backslash\text{OpsHook}$

\*

**Semantics** 5. Expands  $\langle code_2 \rangle$  applied to the list created in step **1**, using the separator specified by  $\langle tl_3 \rangle$ ,  $\langle tl_4 \rangle$ , and  $\langle tl_5 \rangle$ .

$\langle tl_3 \rangle$

**Example**  $\{\sim\text{in}\sim\}$

$\langle tl_4 \rangle$

**Example**  $\{, \sim\}$

$\langle tl_5 \rangle$

**Example**  $\{\sim\&\sim\}$

$\langle code_2 \rangle$

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

$\langle tl_6 \rangle$

**Semantics**  $\backslash\text{Ops}\langle tl_1 \rangle[\langle tl_6 \rangle]$

<hr/> <hr/>	<code>\OpsClear&lt;⟨keyval list⟩&gt;</code>
	<b>Semantics</b> Clears any data created by <code>\Ops{⟨tl<sub>1</sub>⟩}</code> , for all <code>⟨tl<sub>1</sub>⟩</code> in <code>⟨keyval list⟩</code>
<hr/> <hr/>	
<hr/> <hr/>	<code>\OpsDebug</code>
	<b>Semantics</b> See <a href="#">Part IV</a>
<hr/> <hr/>	
<hr/> <hr/>	<code>\OpsHook</code>
	<b>Example</b> <code>\AfterEndEnvironment{theorem}{\OpsHook}</code>
<hr/> <hr/>	
<hr/> <hr/>	<code>\OpsOption{⟨kv10⟩}</code>
	<b>Semantics</b> Set default options for <code>\Ops</code>
Expans	
	<b>Default</b> <code>xo</code>
	<b>Semantics</b> Controls expansion in step <a href="#">2</a>
	<b>Syntax</b> Either of <code>eo</code> , <code>ee</code> , <code>ex</code> , <code>xe</code> , <code>xo</code> , <code>xe</code> , <code>xx</code>
Inner	
	<b>Semantics</b> Default for <code>⟨code<sub>1</sub>⟩</code>
	<b>Syntax</b> Use <code>####1</code> as the argument to be replaced
Name	
	<b>Semantics</b> Default for <code>⟨tl<sub>1</sub>⟩</code>
Outer	
	<b>Semantics</b> Default for <code>⟨code<sub>2</sub>⟩</code>
	<b>Syntax</b> Use <code>####1</code> as the argument to be replaced
Separ	
	<b>Semantics</b> Default for <code>{⟨tl<sub>3</sub>⟩} {⟨tl<sub>3</sub>⟩}{⟨tl<sub>4</sub>⟩} {⟨tl<sub>3</sub>⟩}{⟨tl<sub>4</sub>⟩}{⟨tl<sub>5</sub>⟩}</code>
	<b>Syntax</b> That of ‘separators’ in <a href="#">[3, Section 8 of l3seq]</a>
Write	
	<b>Syntax</b> <code>⟨boolean⟩</code>
<hr/> <hr/>	
<hr/> <hr/>	<code>\OpsRead[⟨path⟩]</code>
	<b>Other</b> The default for <code>⟨path⟩</code> is the last write-file (see <code>⟨kv1⟩</code> )
	<b>Semantics</b> 1. Reads the definitions in <code>⟨path⟩</code> . 2. Writes to <code>oops.log</code> : ‘read from <code>⟨path⟩</code> ’

## Do's and don't's

1.

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

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

2.

Don't: `$\langle key_i \rangle <x$`.

Do: `$\langle key_i \rangle \{<\}x$`

3.

Don't: `\Oups[[a, b)]`.

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

4.

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

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

5. Also see [Part III, section 4](#)

## Part II

# Listing

**Warning:** To reproduce the listings in a L<sup>A</sup>T<sub>E</sub>X document, use the same setting as thout of the documentation portion of oops.dtx (such as \documentclass, \usepackage, and \newtcblisting), and remove any ^^A. Any deviation from the original may require tinkering.<sup>1</sup>

Listing 1.

```
% \OpsOption{
% ^^A% spaces betw. inner and outer brackets matter!->
% Separ={\ \char`@\ }{\ \%\ }{\ \char`@\ }}
% \Ops<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 2.

```
% \OpsOption{ Separ = {\}\{.\}\{.}\}, Outer = {####1} }
% \OpsOption{ Write = \BooleanTrue }
% \Ops<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>\
% \OpsClear
```

<sup>1</sup>For instance, in testing v1.1, I realized \usepackage[T1]{fontenc} was needed, to work with \documentclass{article} in place of \documentclass[full]{l3doc}, hence added it to the documentation portion of oops.dtx

```
% \OpsOption{ Write = \BooleanFalse }
%
```

---

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

### Listing 3.

```
% \OpsRead
% \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
% \OpsClear<Test>
%
```

---

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

### Listing 4.

```
% \OpsOption{Inner, Separ, Outer}
% \Ops[We call~]{Elems={\omega_1}, {\dots}, {\omega_n}}}*
% [~the elementary events, and ]{Space=\Omega}
% [\begin{equation*}\Space=(\Elems)\end{equation*}~the sample space.]
% {}
% \OpsClear
%
```

---

We call  $\omega_1, \dots, \omega_n$  the elementary events, and

$$\Omega = (\omega_1, \dots, \omega_n)$$

the sample space.

### Listing 5.

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



```
% {}
% \OpsClear
% \OpsOption{ Write = \BooleanFalse }
%
```

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

#### Listing 6.

```
% \OpsRead \tab $\Omega$ $\Field$ $\Meas$
% \OpsClear
%
```

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

#### Listing 7.

```
% \OpsOption{ Write = \BooleanTrue }
% \newtheorem{theorem}{Theorem}
% \AfterEndEnvironment{theorem}{\OpsHook}
% \Ops 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] } }+
% [$n \in \mathbb{N}, \sim \text{OffMenge} \subseteq \mathbb{N}^n$ eine offene Menge und
% $f \in C^1(\text{OffMenge}, \mathbb{R})$].
% Dann gibt es auf jeder Strecke $\text{Strecke} \subseteq \text{OffMenge}$ einen
% Punkt $\xi \in \text{Strecke}$,~
% { Steig = { \frac{ f(x)-f(x_0) }{ x-x_0 } } , Punkt = { \xi } }+
% [so dass gilt
% \begin{equation*}
% \text{Steig} = \text{Grad } f(\text{Punkt})^{\top}
% \end{equation*}
% \end{theorem}]
% {}
% \OpsClear
% \OpsOption{ Write = \BooleanFalse }
%
```

**Theorem 1 (Mittelwertsatz für  $n$  Variable)** Es sei  $n \in \mathbb{N}$ ,  $D \subseteq \mathbb{R}^n$  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

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

Listing 8.

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

---


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

## Part III

# Other

### 1 Acknowledgment

This work has benefited from Q&A's from the L<sup>A</sup>T<sub>E</sub>X community, see here: <https://tex.stackexchange.com/users/112708/erwann?tab=questions>. Specific references are made in Part IV. Listing 4 and Listing 5 are from [1]. Listing 7 is from tcolbox[5, 17.3].

### 2 Also see

1. The package cool[2]

### 3 Install

Compiling `oops.dtx` (under Unix, `$tex oops.dtx`) will generate `oops.sty` and `oops.pdf`

### 4 Issue

1. **Don't:** `Inner={\####1\}`  
**Symptom:** `\OpsRead` fails  
**Do:** `Inner={\char'\####1\char'}`

### 5 Support

This package is available from <https://www.ctan.org/pkg/oops> and <https://github.com/rogard/oops>.

### 6 Testing

It's not possible to check the expansion of a certain class of macros against predefined values[6]. Instead, one can check that Part II, as generated in section 3 on one's own machine, agrees with `bench.pdf` available at <https://github.com/rogard/oops>,

## References

- [1] A.N. Shiryaev *Probability* Springer, 1995
- [2] Nick Setzer *The cool package*, 2005 <https://www.ctan.org/pkg/cool>
- [3] The L<sup>A</sup>T<sub>E</sub>X3 Project Team *The L<sup>A</sup>T<sub>E</sub>X3 interfaces* <http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3kernel/interface3.pdf>

- [4] The L<sup>A</sup>T<sub>E</sub>X3 Project Team *The xparse package* <http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3packages/xparse.pdf>
- [5] Thomas F. Sturm *The tcolorbox package* <http://www.texdoc.net/texmf-dist/doc/latex/tcolorbox/tcolorbox.pdf>
- [6] <https://tex.stackexchange.com/a/534100/112708>

## Part IV

# Implementation

```

1 <@@=oops>
2 \NeedsTeXFormat{LaTeX2e}[2019/10/01]
3 \ExplSyntaxOn

```

### 1 aux

```

\__oops_aux_inner_set:n #1 : <code>

4 \cs_new_protected:Nn \__oops_aux_inner_set:n
5 {
6   \cs_gset:Npn \__oops_aux_inner:n ##1 {#1}
7   \cs_generate_variant:Nn \__oops_aux_inner:n { e }
8 }

(End definition for \__oops_aux_inner_set:n.)

\__oops_aux_key:w #1 : <key>
#2 : <value>

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

(End definition for \__oops_aux_key:w.)

\__oops_aux_key:n #1 : <key = value>

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

(End definition for \__oops_aux_key:n.)

\__oops_aux_key:N #1 : <seq>

17 \cs_new_protected:Nn \__oops_aux_key:N
18 {
19   \seq_gclear_new:N \g__oops_aux_key_seq
20   \seq_map_function:NN #1 \__oops_aux_key:n
21 }

(End definition for \__oops_aux_key:N.)

\__oops_aux_outer_set:n #1 : <inline code>

22 \cs_new_protected:Nn \__oops_aux_outer_set:n
23 {
24   \cs_gset:Npn \__oops_aux_outer:n ##1 {#1}
25 }

(End definition for \__oops_aux_outer_set:n.)

\__oops_aux_prop:w #1 : <key>

```

```

#2 : < value >
26 \prop_new:N \g__oops_aux_prop
27 \cs_new_protected:Nn \__oops_aux_prop:nn
28 {
29   \prop_gput:Nnn \g__oops_aux_prop{#1}{#2}
30 }
31 \cs_generate_variant:Nn \__oops_aux_prop:nn { eo, ee, ex, xo, xe, xx }
32 \tl_new:N \g__oops_option_expans_tl
33 \cs_new_protected:Npn \__oops_aux_prop:w #1 = #2 \q_stop
34 {
35   \exp_args:Nx
36   \use:c{__oops_aux_prop:\g__oops_option_expans_tl}
37   { \tl_trim_spaces:n{#1} }
38   { \__oops_aux_inner:n{ \tl_trim_spaces:n{#2} } }
39   %%^A { \__oops_aux_inner:e{ \tl_trim_spaces:n{#2} } }% DEBUG
40 }

```

(End definition for \\_\_oops\_aux\_prop:w.)

```

\__oops_aux_prop:n #1 : < key = value >
41 \cs_new_protected:Nn \__oops_aux_prop:n
42 {
43   \__oops_aux_prop:w #1 \q_stop
44 }

```

(End definition for \\_\_oops\_aux\_prop:n.)

```

\__oops_aux_prop:N #1 : < keyval list >
45 \cs_new_protected:Nn \__oops_aux_prop:N
46 {
47   \prop_gclear_new:N \g__oops_aux_prop
48   \seq_if_empty:NTF #1
49   { \c_empty_tl }
50   {
51     \seq_map_function:NN #1 \__oops_aux_prop:n
52   }
53 }

```

(End definition for \\_\_oops\_aux\_prop:N.)

```

\__oops_aux_separ:nn #1 : < int >
#2 : < tokens >
54 \cs_new:Nn \__oops_aux_separ:nn
55 {
56   \int_case:nnTF {#1}
57   {
58     {1}
59     { \prg_replicate:nn{ 3 }{#2} }
60     {2}
61     {
62       { \use_i:nn #2 }
63       { \use_ii:nn #2 }
64       { \use_i:nn #2 }
65     }

```

```

66     {3}{#2}
67   }
68   { \c_empty_tl }
69   {
70     \msg_error:nnnn { __erw }
71     { separ }
72     { \exp_not:N \__oops_aux_separ:nn }
73     {#2}
74   }
75 }
76 \cs_generate_variant:Nn \__oops_aux_separ:nn { e }

```

(End definition for \\_\_oops\_aux\_separ:nn.)

```

\__oops_aux_separ:n #1 : < tokens >
77 \cs_new:Nn \__oops_aux_separ:n
78 {
79   \__oops_aux_separ:en{ \tl_count:n{#1} }{#1}
80 }

```

(End definition for \\_\_oops\_aux\_separ:n.)

```

\__oops_aux_val:Nn #1 : < seq >
#2 : < tl var name >
81 \cs_new_protected:Nn \__oops_aux_val:Nn
82 {
83   \seq_gclear_new:N \g__oops_aux_val_seq
84   \__oops_seq_from_prop:NNn \g__oops_aux_val_seq #1 { \__oops_prop_name:n{#2} }
85 }

```

(End definition for \\_\_oops\_aux\_val:Nn.)

## 2 log

```

\__oops_log_close:
86 \iow_new:N \g__oops_log_iow
87 \AtEndDocument{\iow_close:N \g__oops_log_iow}
88 \bool_set_false:N \g__oops_log_open_bool
89 \cs_new_protected:Nn \__oops_log_close:
90 {
91   \iow_close:N \g__oops_log_iow
92   \bool_gset_false:N \g__oops_log_open_bool
93 }

```

(End definition for \\_\_oops\_log\_close:.)

```

\__oops_log_open:
94 \cs_new_protected:Nn \__oops_log_open:
95 {
96   \tl_gset:Nx \g__oops_log_to_tl{oops\pdfdate}
97   \iow_open:Nn \g__oops_log_iow {\g__oops_log_to_tl}
98   \bool_gset_true:N \g__oops_log_open_bool
99 }

```

(End definition for \\_oops\_log\_open:.)

```
\_oops_log_read:n #1 : <path>
100 \cs_new_protected:Nn \_oops_log_read:n
101 {
102   \file_input:n{#1}
103   \tl_log:n{read~from~#1}
104 }
105 \cs_generate_variant:Nn \_oops_log_read:n { e }
```

(End definition for \\_oops\_log\_read:n.)

```
\_oops_log_read:
106 \cs_new_protected:Nn \_oops_log_read:
107 {
108   \_oops_log_read:e{\g__oops_log_to_tl}
109 }
```

(End definition for \\_oops\_log\_read:.)

```
\_oops_log_write:n
110 \tl_new:N \g__oops_log_to_tl
111 \cs_new_protected:Nn \_oops_log_write:n
112 {
113   \bool_if:nTF{ \g__oops_log_open_bool }
114   {
115     \iow_now:Nn \g__oops_log_iow {#1}
116     \tl_log:n{ write~to~#1 }
117   }
118   { \msg_error:nnn{ __oops }{ iow }{ \g__oops_log_iow } }
119 }
120 \cs_generate_variant:Nn \_oops_log_write:n { e }
```

(End definition for \\_oops\_log\_write:n.)

### 3 make\_key

```
\_oops_make_key:Nn #1 : < token >
#2 : < key >
121 \cs_new_protected:Nn \_oops_make_key:Nn
122 {
123   \exp_args:NNx
124   \ProvideDocumentCommand{#1}
125   { D<>{\g__oops_option_name_tl} }
126   {
127     \_oops_prop_item:nn{##1}{#2}
128   }
129 }
130 \cs_generate_variant:Nn \_oops_make_key:Nn { c }
```

(End definition for \\_oops\_make\_key:Nn.)



```

__oops_make_key:n #1 : < key >

131 \cs_new_protected:Nn __oops_make_key:n
132 {
133   __oops_make_key:cn{#1}{#1}
134 }
135 \cs_generate_variant:Nn __oops_make_key:n { e }

(End definition for __oops_make_key:n.)

```

```

__oops_make_key:N #1 : < seq >

136 \cs_new_protected:Nn __oops_make_key:N
137 {
138   \seq_map_function:NN #1 __oops_make_key:e
139 }

(End definition for __oops_make_key:N.)

```

## 4 make\_oops

```

__oops_make_oops_exp:nnn

140 \cs_new_protected:Nn __oops_make_oops_exp:nnn
141 {
142   __oops_aux_val:Nn \g__oops_aux_key_seq {#1}
143   __oops_aux_outer_set:n{#3}
144   __oops_aux_outer:n
145   {
146     \exp_args:NNf
147     __oops_seq_use:Nn
148     \g__oops_aux_val_seq
149     {#2}
150   }
151 }

(End definition for __oops_make_oops_exp:nnn.)

```

```

__oops_make_oops_key:nnn

152 \cs_new_protected:Nn __oops_make_oops_key:nnn
153 {
154   __oops_prop_if_exist:nTF{#1}
155   { \c_empty_tl }
156   { __oops_prop_new:n{#1} }
157   \exp_args:No __oops_aux_inner_set:n{#2}
158   \seq_set_from_clist:Nn \g__oops_aux_keyval_seq {#3}
159   __oops_aux_prop:N \g__oops_aux_keyval_seq
160   __oops_prop_append:Nn \g__oops_aux_prop {#1}
161   __oops_aux_key:N \g__oops_aux_keyval_seq
162   __oops_make_key:N \g__oops_aux_key_seq
163 }

(End definition for __oops_make_oops_key:nnn.)

```

\\_oops\\_make\\_oops\\_sideeffect:nnn

```

164 \cs_new_protected:Nn \_oops\_make\_oops\_sideeffect:nnn
165 {
166   \_oops\_make\_oops\_key:nnn{#1}{#2}{#3}
167   \bool_if:nTF{ \g\_oops\_log\_open\_bool }
168   {%^^A https://tex.stackexchange.com/questions/536597
169     \_oops\_log\_write:n
170     {
171       \begingroup
172       \def \_oops\_log\_entry { \Ops<#1>i{#2}{#3} } \expandafter
173       \endgroup \_oops\_log\_entry
174     }
175   }\c\_empty\_tl}
176 }

```

(End definition for \\_oops\\_make\\_oops\\_sideeffect:nnn.)

\\_oops\\_make\\_oops:nnnn

```

#1 : < token list >
#2 : < seq1 >
#3 : < seq2 >
#4 : < prop >

177 \def\OpsHook{\c\_empty\_tl}
178 \cs_new_protected:Npn \_oops\_make\_oops:nnnn #1 #2 #3 #4
179 {
180   \exp_args:NNx \DeclareDocumentCommand \Ops
181   {%^^A      2      3      4 5 6      7 8      9
182     D<>{#1} +o E{ i }{{#2}} m t+ s E{ s o }{{#3}{#4}} +o
183   }
184   {
185     \IfValueT{##2}{##2}
186     \_oops\_make\_oops\_sideeffect:nnn{##1}{##3}{##4}
187     \IfBooleanT{##6}
188     {
189       \_oops\_make\_oops\_exp:nnn{##1}{##7}{##8}
190     }
191     \bool_if:nTF{##5}
192     {
193       \gappto{\OpsHook}
194       {
195         \_oops\_make\_oops\_sideeffect:nnn{##1}{##3}{##4}
196       }
197     }
198     {\c\_empty\_tl}
199     \IfValueT{##9}
200     {
201       \exp_not:n{ \Ops<##1>[##9] }
202     }
203   }
204 }

```

(End definition for \\_oops\\_make\\_oops:nnnn.)

## 5 msg

```

205 \msg_new:nnn {__oops}{ generic }{#1}
206 \msg_new:nnn {__oops}{ iow }{#1~is~closed~can't~write}
207 \msg_new:nnn {__oops}{ keyonly }{#1~does~not~take~values;~keyval~is~#2}
208 \msg_new:nnn {__oops}{ keywrong }{#1~does~not~recognize~key~#2}
209 \msg_new:nnn {__oops}{ separ }{#1~expects~1~to~3~items,~#2}
210 \msg_new:nnn {__oops}{ unset }{#1~unset}

```

## 6 option

```

\__oops_aux_inner:n #1: <code>

211 \cs_new_protected:Nn \__oops_option_inner:n
212 {
213   \tl_gset:Nn \g__oops_option_inner_tl {#1}
214 }
215 \__oops_option_inner:n
216 {
217   \msg_warning:nnn{ __oops }{ unset }{ \exp_not:N \g__oops_option_inner_tl }
218 }

```

(End definition for \\_\_oops\_aux\_inner:n.)

```

\__oops_option_name:n #1: <token list>

219 \cs_new:Nn \__oops_option_name:n
220 {
221   \tl_gset:Nn \g__oops_option_name_tl{#1}
222 }
223 \__oops_option_name:n
224 {
225   \msg_error:nnx{ __oops }
226   { generic }
227   { \exp_not:N\g__oops_option_name_tl~undefined }
228 }

```

(End definition for \\_\_oops\_option\_name:n.)

```

\__oops_option_outer:n #1: < inline code >

229 \cs_new_protected:Nn \__oops_option_outer:n
230 {
231   \tl_gset:Nn \g__oops_option_outer_tl {#1}
232 }
233 \__oops_option_outer:n
234 {
235   \msg_warning:nnn{ __oops }{ unset }{ \exp_not:N \g__oops_option_outer_tl }
236 }

```

(End definition for \\_\_oops\_option\_outer:n.)

```

\__oops_option_separ:n #1: {< token list1 >}{< token list2 >}{< token list3 >}}

237 \cs_new_protected:Nn \__oops_option_separ:n
238 {
239   \cs_gset:Npn \g__oops_option_separ_tl {#1}
240 }

```

```

241 \__oops_option_separ:n
242 {
243   \msg_warning:nnn{ __oops }{ unset }{ \exp_not:N \g__oops_option_separ_tl }
244 }

```

(End definition for \\_\_oops\_option\_separ:n.)

## 7 prop

```

\__oops_prop_append:NN #1 : < prop1 >
\__oops_prop_append:cN #2 : < prop2 >

245 \cs_new_protected:Npn \__oops_prop_append:NN #1 #2
246 {
247   \cs_set:Nn \__oops_prop_append:nn
248   {
249     \prop_gput:Nnx #1 {##1}{ \prop_item:Nn #2{##1} }
250   }
251   \prop_map_function:NN #2 \__oops_prop_append:nn
252 }
253 \cs_generate_variant:Nn \__oops_prop_append:NN { cN }

```

(End definition for \\_\_oops\_prop\_append:NN.)

```

\__oops_prop_append:Nn #1 : < prop >
#2 : < tl var name >

254 \cs_new_protected:Nn \__oops_prop_append:Nn
255 {
256   \__oops_prop_append:cN{ \__oops_prop_name:n {#2} } #1
257 }

```

(End definition for \\_\_oops\_prop\_append:Nn.)

```

\__oops_prop_clear_new:n #1 : < tl var name >

258 \cs_new_protected:Nn \__oops_prop_clear_new:n
259 {
260   \exp_args:No \prop_clear_new:c{ \__oops_prop_name:n {#1} }
261 }

```

(End definition for \\_\_oops\_prop\_clear\_new:n.)

```

\__oops_prop_clear_new_map:n #1 : < keyval list >

262 \cs_new_protected:Nn \__oops_prop_clear_new_map:n
263 {
264   \seq_set_from_clist:Nn \g__oops_aux_key_seq {#1}
265   \seq_map_function:NN \g__oops_aux_key_seq \__oops_prop_clear_new:n
266 }

```

(End definition for \\_\_oops\_prop\_clear\_new\_map:n.)

```

\__oops_prop_if_exist:nTF #1 : < token list1 >
#2 : < token list2 >

```

```

#3 :  $\langle token\ list_3 \rangle$ 
267 \cs_new:Nn \__oops_prop_if_exist:nTF
268 {
269   \prop_if_exist:cTF{ \__oops_prop_name:n {#1} }{#2}{#3}
270 }

```

(End definition for \\_\_oops\_prop\_if\_exist:nTF.)

```

\__oops_prop_item:nn #1 :  $\langle tl\ var\ name \rangle$ 
#2 :  $\langle key \rangle$ 
271 \cs_new:Nn \__oops_prop_item:nn
272 {
273   \prop_item:cn { \__oops_prop_name:n {#1} } {#2}
274 }

```

(End definition for \\_\_oops\_prop\_item:nn.)

```

\__oops_prop_name:n #1 :  $\langle tl\ var\ name \rangle$ 
275 \cs_new:Npn \__oops_prop_name:n #1{ __oops_#1 }

```

(End definition for \\_\_oops\_prop\_name:n.)

```

\__oops_prop_new:n #1 :  $\langle tl\ var\ name \rangle$ 
276 \cs_new_protected:Nn \__oops_prop_new:n
277 {
278   \prop_new:c{ \__oops_prop_name:n {#1} }
279 }

```

(End definition for \\_\_oops\_prop\_new:n.)

## 8 seq

```

\__oops_seq_from_prop:NNn #1 :  $\langle seq_1 \rangle$ 
#2 :  $\langle seq_2 \rangle$  (keys)
#3 :  $\langle prop \rangle$ 
280 \cs_new_protected:Nn \__oops_seq_from_prop:NNn
281 {
282   \cs_set_protected:Nn \__oops_seq_from_prop:n
283   {
284     \seq_gput_right:No #1 { \prop_item:cn{#3}{#1} }
285   }
286   \seq_map_function:NN #2 \__oops_seq_from_prop:n
287 }

```

(End definition for \\_\_oops\_seq\_from\_prop:NNn.)

```

\__oops_erw_seq_use:Nn
288 % \begin{arguments}
289 % \item \meta{ seq }
290 % \item \meta{ tokens }
291 % \end{arguments}
292 \cs_new:Nn \__oops_seq_use:Nn
293 {

```

```

294 \exp_last_unbraced:NNf
295 \seq_use:Nnnn #1
296 \__oops_aux_separ:n{#2}
297 }

```

(End definition for \\_\_oops\_erw\_seq\_use:Nn.)

## 9 Front-end

```

298 \keys_define:nn { __oops }
299 {
300   Expans .multichoices:nn =
301   { eo, ee, ex, xo, xe, xx }
302   { \tl_gset_eq:NN \g__oops_option_expans_tl \l_keys_choice_tl },
303   Expans .default:n = { xo },
304   Expans .initial:n = { xo },
305   Name .code:n={
306     \__oops_option_name:n{#1}
307     \exp_last_unbraced:Nf
308     \__oops_make_oops:nnnn
309     {
310       { \g__oops_option_name_tl }
311       { \g__oops_option_inner_tl }
312       { \g__oops_option_separ_tl }
313       { \g__oops_option_outer_tl }
314     }
315   },
316   Name .value_required:n = false,
317   Name .default:n = { Math },
318   Name .initial:n = { Math },
319   Inner .code:n={
320     \__oops_option_inner:n{#1}
321     \exp_last_unbraced:Nf
322     \__oops_make_oops:nnnn
323     {
324       { \g__oops_option_name_tl }
325       { \g__oops_option_inner_tl }
326       { \g__oops_option_separ_tl }
327       { \g__oops_option_outer_tl }
328     }
329   },
330   Inner .value_required:n = false,
331   Inner .default:n = {####1},
332   Inner .initial:n = {####1},
333   Outer .code:n={
334     \__oops_option_outer:n{#1}
335     \exp_last_unbraced:Nf
336     \__oops_make_oops:nnnn
337     {
338       { \g__oops_option_name_tl }
339       { \g__oops_option_inner_tl }
340       { \g__oops_option_separ_tl }
341       { \g__oops_option_outer_tl }
342     }

```

```

343 },
344 Outer .value_required:n = false,
345 Outer .default:n = { \ensuremath{####1} },
346 Outer .initial:n = { \ensuremath{####1} },
347 Write .code:n = {
348   \bool_if:nTF{#1}
349   {\__oops_log_open:}
350   {\__oops_log_close:}
351 },
352 Write .value_required:n = false,
353 Write .default:n = \BooleanFalse,
354 Write .initial:n = \BooleanFalse,
355 Separ .code:n={
356   \__oops_option_separ:n{#1}
357   \exp_last_unbraced:Nf
358   \__oops_make_oops:nnnn
359   {
360     { \g__oops_option_name_tl }
361     { \g__oops_option_inner_tl }
362     { \g__oops_option_separ_tl }
363     { \g__oops_option_outer_tl }
364   }
365 },
366 Separ .value_required:n = false,
367 Separ .default:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } },
368 Separ .initial:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } }
369 }

```

**\OpsClear** #1 :  $\langle \textit{tl var name} \rangle$

```

370 \NewDocumentCommand{ \OpsClear }
371 { D<>{\g__oops_option_name_tl} }
372 {
373   \__oops_prop_clear_new_map:n{#1}
374 }

```

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

**\OpsDebug**

```

375 \NewDocumentCommand\OpsDebug{m}
376 {
377   \__oops_prop_if_exist:nTF{#1}
378   { \c_empty_tl }
379   { \__oops_prop_new:n{#1} }
380   \__oops_make_key:Nn \KeyA{KeyA}
381   \gappto{\OpsHook}
382   {
383     \__oops_prop_if_exist:nTF{#1}
384     { \c_empty_tl }
385     { \__oops_prop_new:n{#1} }
386     \__oops_make_key:Nn \KeyA{KeyA}
387   }
388 }

```

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

### **\OpsOption**

```
389 \NewDocumentCommand{ \OpsOption }
390 { m }
391 {
392   \keys_set:nn{ __oops }{#1}
393 }
```

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

### **\OpsRead**

```
394 \NewDocumentCommand{\OpsRead}
395 {o}
396 {
397   \IfValueTF{#1}
398   {\__oops_log_read:e{#1}}
399   {\__oops_log_read:}
400 }
```

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

## **10 Misc**

```
401 \ExplSyntaxOff
```