oops, an object oriented practical scribe's package.[*]

Erwann Rogard[†]

Released 2020/04/10

Abstract

oops is a package for LaTeX (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, `\Oops<`⟨*object*⟩`>` begins a series of instructions alternating between 'text' and definitions, that themselves optionally expand using predefined or inline rules. For example,

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

expands to: "Let Ω denote the sample space". As a side effect, `$\Space<Math>$` encodes"Ω". `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

[*]This file describes version v1.4, last revised 2020/04/10.
[†]firstname dot lastname AusTria gmail dot com

Part I
Usage

This part describes .

Convention

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

`\usepackage`

```
\usepackage{oops}
```

Environment Preamble

Requirement `oops.sty` is in the path of the LaTeX engine. See Part III, section 5.

`\Oops`

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

Requirement ⟨kvl_1⟩ 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 `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 step 1.

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

`+`

Other Needed to make the side effect `\Oops`' within a local group persist thereafter

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

`*`

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 `{~\in~}`

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

Example `{,~}`

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

Example `{~\&~}`

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

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

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

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

`\OopsClear` `\OopsClear<`$\langle$*keyval list*$\rangle$`>`

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

`\OopsDebug`

Semantics See Part IV

`\OopsHook` `\OopsHook`

Example `\AfterEndEnvironment{theorem}{\OopsHook}`

`\OopsOption` `\OopsOption{`$\langle$*kvl0*$\rangle$`}`

Semantics Set default options for `\Oops`

`Expans`

Default `xo`

Semantics Controls expansion in step 2

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

`Inner`

Semantics Default for $\langle code_1\rangle$

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

`Name`

Semantics Default for $\langle tl_1\rangle$

`Outer`

Semantics Default for $\langle code_2\rangle$

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

`Separ`

Semantics Default for `{`$\langle tl_3\rangle$`}`|`{`$\langle tl_3\rangle$`}{`$\langle tl_4\rangle$`}`|`{`$\langle tl_3\rangle$`}{`$\langle tl_4\rangle$`}{`$\langle tl_5\rangle$`}`

Syntax That of 'separators' in [3, Section 8 of l3seq]

`Write`

Syntax $\langle$*boolean*$\rangle$

`\OopsRead` `\OopsRead[`$\langle$*path*$\rangle$`]`

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 `oops.log`: 'read from $\langle$*path*$\rangle$'

Do's and dont's

1.

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

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

2.

   Don't: `$\`$\langle$*key*$_i\rangle$`<x$`.

   Do: `$\`$\langle$*key*$_i\rangle$`{<}x$`

3.

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

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

4.

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

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

5. Also see Part III, section 4

Part II
Listing

Warning: To reproduce the listings in a LaTeX document, use the same setting as thoat 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.[1]

Listing 1.

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

```
%       \OopsOption{ Separ = {{}{.}{.}}, Outer = {####1} }
%       \OopsOption{ Write = \BooleanTrue }
%       \Oops<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>\\
%       \OopsClear
```

[1] 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`

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

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

Listing 3.

```
%     \OopsRead
%     \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
%     \OopsClear<Test>
%
```

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

Listing 4.

```
%     \OopsOption{Inner, Separ, Outer}
%     \Oops[We call~]{Elems={{\omega_{1}}, {\dots}, {\omega_{n}}}}*
%     [~the elementary events, and ]{Space=\Omega}
%     [\begin{equation*}\Space=(\Elems)\end{equation*}~the sample space.]
%     {}
%     \OopsClear
%
```

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

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

the sample space.

Listing 5.

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

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

Listing 6.

```
%           \OopsRead \tab $\Omega$ $\Field$ $\Meas$
%           \OopsClear
%
```

Ω $\mathcal{F}$ $\mathcal{P}$

Listing 7.

```
%       \OopsOption{ Write = \BooleanTrue }
%       \newtheorem{theorem}{Theorem}
%       \AfterEndEnvironment{theorem}{\OopsHook}
%       \Oops 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}]
%       {}
%       \OopsClear
%       \OopsOption{ 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*

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

Listing 8.

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

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

Part III
Other

1 Acknowledgment

This work has benefited from Q&A's from the LaTeXcommunity, 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:** `\newtheorem{definition}{Definition}`;
   and `\Oops[\begin{definition}]{` ⟨key_i⟩ `=` ⟨val_i⟩ `}`

   Symptom: `ERROR: Undefined control sequence.` (not systematic) for ⟨key_i⟩

   Do: `\Oops  \meta{\pkgkey}  = \meta{\pkgval}  \begin{definition}]{`

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

   Symptom: `\OopsRead` fails

   Do: `Inner={\char‘{####1\char‘}}`

   See: Listing 1

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 LaTeX3 Project Team *The LaTeX3 interfaces* http://ftp.math.purdue.edu/mirrors/ctan.org/macros/latex/contrib/l3kernel/interface3.pdf

[4] The LaTeX3 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

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

1 aux

__oops_aux_inner_set:n

#1 : ⟨*code*⟩

```
\cs_new_protected:Nn \__oops_aux_inner_set:n
{
  \cs_gset:Npn \__oops_aux_inner:n ##1 {#1}
  \cs_generate_variant:Nn \__oops_aux_inner:n { e }
}
```

(*End definition for* __oops_aux_inner_set:n.)

__oops_aux_key:w

#1 : ⟨ *key* ⟩
#2 : ⟨ *value* ⟩

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

(*End definition for* __oops_aux_key:w.)

__oops_aux_key:n

#1 : ⟨ *key* = *value* ⟩

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

(*End definition for* __oops_aux_key:n.)

__oops_aux_key:N

#1 : ⟨ *seq* ⟩

```
\cs_new_protected:Nn \__oops_aux_key:N
{
  \seq_gclear_new:N \g__oops_aux_key_seq
  \seq_map_function:NN #1 \__oops_aux_key:n
}
```

(*End definition for* __oops_aux_key:N.)

__oops_aux_outer_set:n

#1 : ⟨ *inline code* ⟩

```
\cs_new_protected:Nn \__oops_aux_outer_set:n
{
  \cs_gset:Npn \__oops_aux_outer:n ##1 {#1}
}
```

(*End definition for* __oops_aux_outer_set:n.)

__oops_aux_prop:w

#1 : ⟨ *key* ⟩

#2 : ⟨ *value* ⟩

```
\prop_new:N \g__oops_aux_prop
\cs_new_protected:Nn \__oops_aux_prop:nn
{
  \prop_gput:Nnn \g__oops_aux_prop{#1}{#2}
}
\cs_generate_variant:Nn \__oops_aux_prop:nn { eo, ee, ex, xo, xe, xx }
\tl_new:N \g__oops_option_expans_tl
\cs_new_protected:Npn \__oops_aux_prop:w #1 = #2 \q_stop
{
  \exp_args:Nx
  \use:c{__oops_aux_prop:\g__oops_option_expans_tl}
  { \tl_trim_spaces:n{#1} }
  { \__oops_aux_inner:n{ \tl_trim_spaces:n{#2} } }
%^^A  { \__oops_aux_inner:e{ \tl_trim_spaces:n{#2} } }% DEBUG
}
```

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

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

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

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

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

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

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

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

```
\cs_new:Nn \__oops_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 \__oops_aux_separ:nn }
    {#2}
  }
}
\cs_generate_variant:Nn \__oops_aux_separ:nn { e }
```

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

`\__oops_aux_separ:n`

#1 : ⟨ *tokens* ⟩

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

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

`\__oops_aux_val:Nn`

#1 : ⟨ *seq* ⟩
#2 : ⟨ *tl var name* ⟩

```
\cs_new_protected:Nn \__oops_aux_val:Nn
{
  \seq_gclear_new:N \g__oops_aux_val_seq
  \__oops_seq_from_prop:NNn \g__oops_aux_val_seq #1 { \__oops_prop_name:n{#2} }
}
```

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

2 log

`\__oops_log_close:`

```
\iow_new:N \g__oops_log_iow
\AtEndDocument{\iow_close:N \g__oops_log_iow}
\bool_set_false:N \g__oops_log_open_bool
\cs_new_protected:Nn \__oops_log_close:
{
  \iow_close:N \g__oops_log_iow
  \bool_gset_false:N \g__oops_log_open_bool
}
```

(*End definition for* `\__oops_log_close:`*.*)

`\__oops_log_open:`

```
\cs_new_protected:Nn \__oops_log_open:
{
  \tl_gset:Nx \g__oops_log_to_tl{oops\pdfdate}
  \iow_open:Nn \g__oops_log_iow {\g__oops_log_to_tl}
  \bool_gset_true:N \g__oops_log_open_bool
}
```

(*End definition for* `\__oops_log_open:`.)

`\__oops_log_read:n` #1 : ⟨*path*⟩

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

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

`\__oops_log_read:`

```
\cs_new_protected:Nn \__oops_log_read:
{
  \__oops_log_read:e{\g__oops_log_to_tl}
}
```

(*End definition for* `\__oops_log_read:`.)

`\__oops_log_write:n`

```
\tl_new:N \g__oops_log_to_tl
\cs_new_protected:Nn \__oops_log_write:n
{
  \bool_if:nTF{ \g__oops_log_open_bool }
  {
    \iow_now:Nn \g__oops_log_iow {#1}
    \tl_log:n{ write~to~#1 }
  }
  { \msg_error:nnn{ __oops }{ iow }{ \g__oops_log_iow }  }
}
\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* ⟩

```
\cs_new_protected:Nn \__oops_make_key:Nn
{
  \exp_args:NNx
  \ProvideDocumentCommand{#1}
  { D<>{\g__oops_option_name_tl} }
  {
    \__oops_prop_item:nn{##1}{#2}
  }
}
\cs_generate_variant:Nn \__oops_make_key:Nn {c}
```

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

__oops_make_key:n #1 : ⟨ *key* ⟩

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

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

__oops_make_key:N #1 : ⟨ *seq* ⟩

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

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

4 make_oops

__oops_make_oops_exp:nnn

```
\cs_new_protected:Nn \__oops_make_oops_exp:nnn
{
  \__oops_aux_val:Nn \g__oops_aux_key_seq {#1}
  \__oops_aux_outer_set:n{#3}
  \__oops_aux_outer:n
  {
    \exp_args:NNf
    \__oops_seq_use:Nn
    \g__oops_aux_val_seq
    {#2}
  }
}
```

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

__oops_make_oops_key:nnn

```
\cs_new_protected:Nn \__oops_make_oops_key:nnn
{
  \__oops_prop_if_exist:nTF{#1}
  { \c_empty_tl }
  { \__oops_prop_new:n{#1} }
  \exp_args:No \__oops_aux_inner_set:n{#2}
  \seq_set_from_clist:Nn \g__oops_aux_keyval_seq {#3}
  \__oops_aux_prop:N \g__oops_aux_keyval_seq
  \__oops_prop_append:Nn \g__oops_aux_prop {#1}
  \__oops_aux_key:N \g__oops_aux_keyval_seq
  \__oops_make_key:N \g__oops_aux_key_seq
}
```

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

__oops_make_oops_sideeffect:nnn

```
\cs_new_protected:Nn \__oops_make_oops_sideeffect:nnn
{
  \__oops_make_oops_key:nnn{#1}{#2}{#3}
  \bool_if:nTF{ \g__oops_log_open_bool }
  {%^^A https://tex.stackexchange.com/questions/536597
    \__oops_log_write:n
    {
      \begingroup
      \def \__oops_log_entry { \Oops<#1>i{#2}{#3} } \expandafter
      \endgroup \__oops_log_entry
    }
  }{\c_empty_tl}
}
```

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

__oops_make_oops:nnnn

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

```
\def\OopsHook{\c_empty_tl}
\cs_new_protected:Npn \__oops_make_oops:nnnn #1 #2 #3 #4
{
  \exp_args:NNx \DeclareDocumentCommand \Oops
  {%^^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}
    \__oops_make_oops_sideeffect:nnn{##1}{##3}{##4}
    \IfBooleanT{##6}
    {
      \__oops_make_oops_exp:nnn{##1}{##7}{##8}
    }
    \bool_if:nTF{##5}
    {
      \gappto{\OopsHook}
      {
        \__oops_make_oops_sideeffect:nnn{##1}{##3}{##4}
      }
    }
    {\c_empty_tl}
    \IfValueT{##9}
    {
      \exp_not:n{ \Oops<##1>[##9] }
    }
  }
}
```

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

5 msg

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

6 option

`\__oops_aux_inner:n` #1 : ⟨*code*⟩

```
\cs_new_protected:Nn \__oops_option_inner:n
{
  \tl_gset:Nn \g__oops_option_inner_tl {#1}
}
\__oops_option_inner:n
{
  \msg_warning:nnn{ __oops }{ unset }{ \exp_not:N \g__oops_option_inner_tl }
}
```

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

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

```
\cs_new:Nn \__oops_option_name:n
{
  \tl_gset:Nn \g__oops_option_name_tl{#1}
}
\__oops_option_name:n
{
  \msg_error:nnx{ __oops }
  { generic }
  { \exp_not:N\g__oops_option_name_tl~undefined }
}
```

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

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

```
\cs_new_protected:Nn \__oops_option_outer:n
{
  \tl_gset:Nn \g__oops_option_outer_tl {#1}
}
\__oops_option_outer:n
{
  \msg_warning:nnn{ __oops }{ unset }{ \exp_not:N \g__oops_option_outer_tl }
}
```

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

`\__oops_option_separ:n` #1 : {⟨ *token list*$_1$ ⟩}{⟨ *token list*$_2$ ⟩}{⟨ *token list*$_3$ ⟩}

```
\cs_new_protected:Nn \__oops_option_separ:n
{
  \cs_gset:Npn \g__oops_option_separ_tl {#1}
}
```

```
\__oops_option_separ:n
{
  \msg_warning:nnn{ __oops }{ unset }{ \exp_not:N \g__oops_option_separ_tl }
}
```

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

7 prop

`\__oops_prop_append:NN`
`\__oops_prop_append:cN`

#1 : ⟨ *prop*$_1$ ⟩
#2 : ⟨ *prop*$_2$ ⟩

```
\cs_new_protected:Npn \__oops_prop_append:NN #1 #2
{
  \cs_set:Nn \__oops_prop_append:nn
  {
    \prop_gput:Nnx #1 {##1}{ \prop_item:Nn #2{##1} }
  }
  \prop_map_function:NN #2 \__oops_prop_append:nn
}
\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* ⟩

```
\cs_new_protected:Nn \__oops_prop_append:Nn
{
  \__oops_prop_append:cN{ \__oops_prop_name:n {#2} } #1
}
```

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

`\__oops_prop_clear_new:n`

#1 : ⟨ *tl var name* ⟩

```
\cs_new_protected:Nn \__oops_prop_clear_new:n
{
  \exp_args:No \prop_clear_new:c{ \__oops_prop_name:n {#1} }
}
```

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

`\__oops_prop_clear_new_map:n`

#1 : ⟨ *keyval list* ⟩

```
\cs_new_protected:Nn \__oops_prop_clear_new_map:n
{
  \seq_set_from_clist:Nn \g__oops_aux_key_seq {#1}
  \seq_map_function:NN \g__oops_aux_key_seq \__oops_prop_clear_new:n
}
```

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

`\__oops_prop_if_exist:nTF`

#1 : ⟨*token list*$_1$⟩
#2 : ⟨*token list*$_2$⟩

#3 : ⟨*token list*$_3$⟩

```
\cs_new:Nn \__oops_prop_if_exist:nTF
{
  \prop_if_exist:cTF{ \__oops_prop_name:n {#1} }{#2}{#3}
}
```

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

`\__oops_prop_item:nn`

#1 : ⟨ *tl var name* ⟩
#2 : ⟨ *key* ⟩

```
\cs_new:Nn \__oops_prop_item:nn
{
  \prop_item:cn { \__oops_prop_name:n {#1} } {#2}
}
```

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

`\__oops_prop_name:n`

#1 : ⟨ *tl var name* ⟩

```
\cs_new:Npn \__oops_prop_name:n #1{ __oops_#1 }
```

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

`\__oops_prop_new:n`

#1 : ⟨ *tl var name* ⟩

```
\cs_new_protected:Nn \__oops_prop_new:n
{
  \prop_new:c{ \__oops_prop_name:n {#1} }
}
```

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

8 seq

`\__oops_seq_from_prop:NNn`

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

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

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

`\__oops_erw_seq_use:Nn`

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

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

9 Front-end

```
\keys_define:nn { __oops }
{
  Expans .multichoices:nn =
  { eo, ee, ex, xo, xe, xx }
  { \tl_gset_eq:NN \g__oops_option_expans_tl \l_keys_choice_tl },
  Expans .default:n = { xo },
  Expans .initial:n = { xo },
  Name .code:n={
    \__oops_option_name:n{#1}
    \exp_last_unbraced:Nf
    \__oops_make_oops:nnnn
    {
      { \g__oops_option_name_tl }
      { \g__oops_option_inner_tl }
      { \g__oops_option_separ_tl }
      { \g__oops_option_outer_tl }
    }
  },
  Name .value_required:n = false,
  Name .default:n = { Math },
  Name .initial:n = { Math },
  Inner .code:n={
    \__oops_option_inner:n{#1}
    \exp_last_unbraced:Nf
    \__oops_make_oops:nnnn
    {
      { \g__oops_option_name_tl }
      { \g__oops_option_inner_tl }
      { \g__oops_option_separ_tl }
      { \g__oops_option_outer_tl }
    }
  },
  Inner .value_required:n = false,
  Inner .default:n = {####1},
  Inner .initial:n = {####1},
  Outer .code:n={
    \__oops_option_outer:n{#1}
    \exp_last_unbraced:Nf
    \__oops_make_oops:nnnn
    {
      { \g__oops_option_name_tl }
      { \g__oops_option_inner_tl }
      { \g__oops_option_separ_tl }
      { \g__oops_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}
    {\__oops_log_open:}
    {\__oops_log_close:}
  },
  Write .value_required:n = false,
  Write .default:n = \BooleanFalse,
  Write .initial:n = \BooleanFalse,
  Separ .code:n={
    \__oops_option_separ:n{#1}
    \exp_last_unbraced:Nf
    \__oops_make_oops:nnnn
    {
      { \g__oops_option_name_tl }
      { \g__oops_option_inner_tl }
      { \g__oops_option_separ_tl }
      { \g__oops_option_outer_tl }
    }
  },
  Separ .value_required:n = false,
  Separ .default:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } },
  Separ .initial:n = { {\ }and{\ } } { ,{\ } } { ,{\ }and{\ } }
}
```

\OopsClear #1 : ⟨ *tl var name* ⟩

```
\NewDocumentCommand{ \OopsClear }
{ D<>{\g__oops_option_name_tl} }
{
  \__oops_prop_clear_new_map:n{#1}
}
```

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

\OopsDebug

```
\NewDocumentCommand\OopsDebug{m}
{
  \__oops_prop_if_exist:nTF{#1}
  { \c_empty_tl }
  { \__oops_prop_new:n{#1} }
  \__oops_make_key:Nn \KeyA{KeyA}
  \gappto{\OopsHook}
  {
    \__oops_prop_if_exist:nTF{#1}
    { \c_empty_tl }
    { \__oops_prop_new:n{#1} }
    \__oops_make_key:Nn \KeyA{KeyA}
  }
}
```

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

\OopsOption

```
\NewDocumentCommand{ \OopsOption }
{ m }
{
  \keys_set:nn{ __oops }{#1}
}
```

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

\OopsRead

```
\NewDocumentCommand{\OopsRead}
{o}
{
  \IfValueTF{#1}
  {\__oops_log_read:e{#1}}
  {\__oops_log_read:}
}
```

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

10 Misc

```
\ExplSyntaxOff
```