The **eqnlines** Package
Source Code Documentation

Niklas Beisert

Institut für Theoretische Physik
Eidgenössische Technische Hochschule Zürich
Wolfgang-Pauli-Strasse 27, 8093 Zürich, Switzerland

`nbeisert@itp.phys.ethz.ch`

v1.0.1: 2026-03-23

`https://ctan.org/pkg/eqnlines`
`https://github.com/nbeisert/latex-pkg-nb`

Abstract

eqnlines is a LaTeX 2_ε package providing a framework for typesetting single- and multi-line equations which extends the established equation environments of LaTeX and the amsmath package with many options for convenient adjustment of the intended layout. In particular, the package adds flexible schemes for numbering, horizontal alignment and semi-automatic punctuation, and it improves upon the horizontal and vertical spacing options. The extensions can be used and adjusted through optional arguments and modifiers to the equation environments as well as global settings.

Contents

1 Implementation

This appendix documents the implementation for the various components of the eqnlines package.

The code for the package is based on the amsmath package, see the reference manual for details. It was forked at version v2.17t dated 2024-11-05. Most of the code was substantially redesigned (macros renamed, reshuffled, enhanced), but many of the underlying mechanisms were preserved. The documentation thus contains excerpts from the amsmath package documentation explaining some details of the implementation.

Please note that the documentation is completed only for few sections in the present version. Various open issues are remarked.

2 General Support

In the following we describe general purpose supporting routines.

2.1 Debugging Messages

The package offers a verbose mode for debugging purposes. It outputs extra information on the current location within the code in order to track progress: **TODO:** describe

```
\def\eql@verbose@on{%
  \def\eql@verbose@info##1{\PackageInfo{eqnlines}{##1}}%
  \def\eql@verbose@infoarg##1##2{\eql@verbose@info{##1##2}}%
}
\def\eql@verbose@off{%
  \let\eql@verbose@info\@gobble
  \let\eql@verbose@infoarg\@gobbletwo
}
\eql@verbose@off
```

TODO: describe

```
\def\eql@verbose@msg@enterenv{entering \@currenvir}
\def\eql@verbose@msg@leaveenv{ leaving \@currenvir}
\def\eql@verbose@msg@start#1{starting \string#1}
\def\eql@verbose@msg@end#1{ \space ending \string#1}
\def\eql@verbose@msg@within#1{ \space within \string#1}
\def\eql@verbose@msg@enter#1{entering \string#1}
\def\eql@verbose@msg@leave#1{ leaving \string#1}
\def\eql@verbose@msg@startline@number{starting line \the\eql@row@}
\def\eql@verbose@msg@startline@new{starting new line}
```

2.2 Supporting Definitions

`\eql@false` (*bool*)
`\eql@true` (*bool*)

Rather than the standard LaTeX scheme of `\`*xxx*`false`, `\`*xxx*`true` and `\if`*xxx* for boolean variables *xxx*, we use a scheme where `\`*xxx* is either undefined or defined (to an empty macro) and is tested against by the ε-TeX conditional `\ifdefined\`*xxx*. In order to make the scheme more tangible, we define the two expected values for boolean variables:

```
\let\eql@false\@undefined
\let\eql@true\@empty
```

TODO: for comparison within `\ifx`

```
\def\eql@relax{\relax}
```

TODO: describe

```
\def\eql@append#1#2{\edef#1{\unexpanded\expandafter{#1#2}}}
\def\eql@appendexpand#1#2{\edef#1{\unexpanded\expandafter{#1}#2}}
\def\eql@appendmacro#1#2{\eql@appendexpand#1{\unexpanded\expandafter{#2}}}
\def\eql@letcs#1{\expandafter\let\csname#1\endcsname}
```

2.3 Dollardollar Abstraction

`l@dollardollar@begin`
`eql@dollardollar@end`

As of 2025 LaTeX defines `\dollardollar@begin` and `\dollardollar@end` to represent (and adjust) the beginning and end of bare TeX display equations (‘`$$`’). For the time being, we make sure to revert to ‘`$$`’ if these macros are not yet available:

```
\ifdefined\dollardollar@begin
  \def\eql@dollardollar@begin{\dollardollar@begin}
  \def\eql@dollardollar@end{\dollardollar@end}
\else
  \def\eql@dollardollar@begin{$$}
  \def\eql@dollardollar@end{$$}
\fi
```

2.4 Look-Ahead in Alignment

Scanning for optional arguments `[...]` or modifiers such as ‘`*`’ using the LaTeX `\@ifnextchar` mechanism has two challenges within aligned equations: a square bracket or star may well be part of the intended mathematical expression and the look-ahead could trip upon an alignment character ‘`&`’ which inadvertently triggers to enter the next alignment column.

`eql@ifnextchar@loose`
`eql@ifnextchar@tight`

To address the first challenge, we can force the special characters to follow immediately the macro invocation. For clarity, we copy LaTeX’s original `\@ifnextchar` in `\kernel@ifnextchar` which skips over spaces as `\eql@ifnextchar@loose`. We replicate the amsgen version `\new@ifnextchar` that does not skip over spaces as `\eql@ifnextchar@loose`. The space before `#1` allows to look-ahead for spaces as well:

```
\let\eql@ifnextchar@loose\kernel@ifnextchar
\long\def\eql@ifnextchar@tight#1#2#3{%
  \let\reserved@d= #1%
  \def\reserved@a{#2}%
  \def\reserved@b{#3}%
  \futurelet\@let@token\eql@ifnch@tight
}
\def\eql@ifnch@tight{%
  \ifx\@let@token\reserved@d
    \let\reserved@b\reserved@a
  \fi
  \reserved@b
}
```

`\eql@atxi`
`\eql@atxii`

Capture ‘`@`’ as a character (catcode 12) rather than a letter (catcode 11) as `\eql@atxii` so that we can look-ahead for ‘`@`’ with both `\makeatother` and `\makeatletter` modes:

```
\let\eql@atxi=@
\begingroup
  \makeatother
  \let\tmp=@%
  \makeatletter
  \global\let\eql@atxii\tmp
\endgroup
```

`eql@ifnextgobble@...`
`\eql@ifstar@...`
`\eql@testopt@...`
`\eql@teststaropt@...`

We introduce a collection of look-ahead macros which do or do not skip over spaces. The macros `\eql@ifstar@...` and `\eql@testopt@...` replicate the LaTeX counterparts `\@ifstar` and `\@testopt`. The macros `\eql@ifnextgobble@...` work like `\@ifnextchar`, but also gobble the specific character if found; one might define `\eql@ifstar@...` as `\eql@ifnextgobble@...*`. The macros `\eql@teststaropt@...` tests for combinations of ‘`*`’ and optional arguments `[...]`:

```
\long\def\eql@ifnextgobble@loose#1#2{\eql@ifnextchar@loose#1{\@firstoftwo{#2}}}
\long\def\eql@ifnextgobble@tight#1#2{\eql@ifnextchar@tight#1{\@firstoftwo{#2}}}
\long\def\eql@ifstar@loose#1{\eql@ifnextchar@loose*{\@firstoftwo{#1}}}
\long\def\eql@ifstar@tight#1{\eql@ifnextchar@tight*{\@firstoftwo{#1}}}
\long\def\eql@ifat@loose#1#2{\eql@ifnextgobble@loose{@}{#1}{%
  \eql@ifnextgobble@loose\eql@atxii{#1}{#2}}}
\long\def\eql@ifat@tight#1#2{\eql@ifnextgobble@tight{@}{#1}{%
  \eql@ifnextgobble@tight\eql@atxii{#1}{#2}}}
```

```
\long\def\eql@testopt@loose#1#2{\eql@ifnextchar@loose[{#1}{#1[{#2}]}}%]
\long\def\eql@testopt@tight#1#2{\eql@ifnextchar@tight[{#1}{#1[{#2}]}}%]
\long\def\eql@teststaropt@loose#1#2#3{%
  \eql@ifstar@loose{\eql@testopt@loose{#1}{#3}}{\eql@testopt@loose{#2}{#3}}}
\long\def\eql@teststaropt@tight#1#2#3{%
  \eql@ifstar@tight{\eql@testopt@tight{#1}{#3}}{\eql@testopt@tight{#2}{#3}}}
\long\def\eql@teststaroropt@loose#1#2#3{%
  \eql@ifstar@loose{#1}{\eql@testopt@loose{#2}{#3}}}
\long\def\eql@teststaroropt@tight#1#2#3{%
  \eql@ifstar@tight{#1}{\eql@testopt@tight{#2}{#3}}}
\long\def\eql@gobbleopt[#1]{}
\long\def\eql@gobbleoptone[#1]#2{}
```

TODO: describe

```
\def\eql@testopt@default{\eql@testopt@default}
```

TODO: describe

```
\let\eql@parseopt@warn@env\@empty
\let\eql@parseopt@warn@cr\@empty
```

TODO: describe

```
\def\eql@parseopt@env{%
  \let\eql@parseopt@warn\eql@parseopt@warn@env\eql@parseopt}
\def\eql@parseopt@cr{%
  \let\eql@parseopt@warn\eql@parseopt@warn@cr\eql@parseopt}
```

TODO: describe

```
\def\eql@parseopt#1#2{%
  \def\eql@parseopt@case{#1}%
  \def\eql@parseopt@end{#2}%
  \eql@parseopt@peek
}
\def\eql@parseopt@peek{%
  \futurelet\eql@parseopt@token\eql@parseopt@select
}
\def\eql@parseopt@select{%
  \let\eql@parseopt@next\eql@parseopt@other
  \ifx\eql@parseopt@token\@sptoken
    \let\eql@parseopt@next\eql@parseopt@end
  \fi
  \eql@parseopt@case
  \eql@parseopt@next
}
\def\eql@parseopt@other{\eql@parseopt@warn\eql@parseopt@end}
\def\eql@parseopt@gobble#1{\eql@parseopt@peek}
```

`\eql@spbgroup`
`\eql@spegroup`
`\eql@srbgroup`
`\eql@sregroup`

The second challenge is addressed by enclosing the look-ahead in spurious groups[1] which protect against triggering ‘&’. The macros `\eql@spbgroup` and `\eql@spegroup` open and close a spurious group. For some reason, the look-ahead mechanism requires further protections by inserting `\relax` at the beginning and by resetting `\@let@token` at the end. These adjustments are included in the macros `\eql@srbgroup` and `\eql@spegroup`:

```
\def\eql@spbgroup{\iffalse{\fi\ifnum0=`}\fi}
```

---

[1] See `https://www.latex-project.org/cgi-bin/ltxbugs2html?pr=latex/3040`, `https://www.latex-project.org/cgi-bin/ltxbugs2html?pr=amslatex/1834` and `https://tex.stackexchange.com/questions/9897/showcase-of-brace-tricks-egroup-iffalse-fi-etc`.

```
\def\eql@spegroup{\ifnum0=`{\fi\iffalse}\fi}
\def\eql@srbgroup{\relax\iffalse{\fi\ifnum0=`}\fi}
\def\eql@sregroup{\let\@let@token\relax\ifnum0=`{\fi\iffalse}\fi}
```

`\eql@ampprotect`
`\eql@ampprotecttwo`

The macros `\eql@ampprotect` and `\eql@ampprotecttwo` inject the opening and closing of spurious groups into the look-ahead mechanism:

```
\long\def\eql@ampprotect#1#2{\eql@srbgroup#1{\eql@sregroup#2}}
\long\def\eql@ampprotecttwo#1#2#3{%
  \eql@srbgroup#1{\eql@sregroup#2}{\eql@sregroup#3}}
```

`...@ampsafe`

We introduce a collection of ‘`&`’-safe look-ahead macros:

```
\def\eql@ifnextchar@loose@ampsafe#1{%
  \eql@ampprotecttwo{\eql@ifnextchar@loose#1}}
\def\eql@ifnextchar@tight@ampsafe#1{%
  \eql@ampprotecttwo{\eql@ifnextchar@tight#1}}
\def\eql@ifstar@loose@ampsafe{\eql@ampprotecttwo\eql@ifstar@loose}
\def\eql@ifstar@tight@ampsafe{\eql@ampprotecttwo\eql@ifstar@tight}
\def\eql@testopt@loose@ampsafe{\eql@ampprotect\eql@testopt@loose}
\def\eql@testopt@tight@ampsafe{\eql@ampprotect\eql@testopt@tight}
\def\eql@teststaropt@loose@ampsafe{\eql@ampprotecttwo\eql@teststaropt@loose}
\long\def\eql@teststaropt@tight@ampsafe{%
  \eql@ampprotecttwo\eql@teststaropt@tight}
```

`\eql@amproof`
`\eql@amprevert`

We may want to replace LaTeX’s definitions `\@ifnextchar`, `\@ifstar` and `\@testopt` to respect ‘`&`’ characters within aligned equations. This might make unrelated definitions with optional arguments and starred variants more robust in this context. The macro `\eql@amproof` overwrites the original definitions, and `\eql@amprevert` reverts the changes:

```
\let\eql@ifnextchar@org\@ifnextchar
\let\eql@ifstar@org\@ifstar
\let\eql@testopt@org\@testopt
\def\eql@amprevert{%
  \let\@ifnextchar\eql@ifnextchar@org
  \let\@testopt\eql@testopt@org
  \let\@ifstar\eql@ifstar@org
}
\def\eql@ampproof{%
  \let\@ifnextchar\eql@ifnextchar@loose@ampsafe
  \let\@testopt\eql@testopt@loose@ampsafe
  \let\@ifstar\eql@ifstar@loose@ampsafe
}
```

2.5 Error Messages

`\eql@error`
`\eql@warning`

Main error and warning message function for the package:

```
\def\eql@error#1{\PackageError{eqnlines}{#1}{}}
\def\eql@warning{\PackageWarning{eqnlines}}
```

`\eql@error@mathmode`

Error messages concerning math mode:

```
\def\eql@warn@here#1{\eql@warning{\string#1 not allowed outside equations}}
\def\eql@error@mathmode#1{\eql@error{#1 allowed only in paragraph mode}}
```

`ql@warn@label@unused`
`@warn@label@multiple`
`\eql@warn@tag@unused`
`ql@warn@tag@multiple`
`eql@warn@name@unused`
`l@warn@name@multiple`
`\eql@warn@ref@unused`
`ql@warn@ref@multiple`

Warning messages concerning unused and multiply declared labels and tags:

```
\def\eql@warn@tags@unused#1#2{\eql@warning{Unused equation #1:
    #2 will be lost}}
\def\eql@warn@tags@multiple#1#2#3{\eql@warning{Multiple equation #1:
    previous #2 will be lost#3}}
\def\eql@warn@label@unused{\eql@warn@tags@unused{\string\label}%
    {label '\eql@tags@label'}}
\def\eql@warn@label@multiple#1{\eql@warn@tags@multiple{\string\label's}%
    {label '\eql@tags@label'}{ and replaced by '#1'}}
\def\eql@warn@name@unused{\eql@warn@tags@unused{label name}%
    {name declaration}}
\def\eql@warn@name@multiple{\eql@warn@tags@multiple{label names}%
    {name declaration}{}}
\def\eql@warn@tag@unused{\eql@warn@tags@unused{\string\tag}%
    {tag declaration}}
\def\eql@warn@tag@multiple{\eql@warn@tags@multiple{\string\tag's}%
    {tag declaration will be lost}{}}
\def\eql@warn@ref@unused{\eql@warn@tags@unused{tag label}%
    {tag label declaration}}
\def\eql@warn@ref@multiple{\eql@warn@tags@multiple{tag labels}%
    {tag label declaration}{}}
```

```
\def\eql@warn@parseopt{\eql@warning{Unknown modifier token:
    modifier parsing stopped}}
\def\eql@warn@parseopt@verbose{\eql@warning{Unknown modifier token:
    '\meaning\eql@parseopt@token'}}
```

2.6 amsmath Integration

`\eql@amsmath@after`
`\eql@amsmath@before`
`eql@amsmath@undefine`
`\eql@amsmath@let`

We need to overwrite certain macros from amsmath. The method `\eql@amsmath@after` executes argument `#1` after loading amsmath is loaded. It also runs the code if amsmath has already been loaded. Furthermore, loading amsmath requires certain macros to be undefined. To this end `\eql@amsmath@before` will execute argument `#1` before any future loading of amsmath. `\eql@amsmath@undefine` undefines a macro in this way and `\eql@amsmath@let` overwrites a macro of `\amsmath/`:

```
\def\eql@amsmath@after#1{\AddToHook{package/amsmath/after}{#1}}
\def\eql@amsmath@before#1{%
  \@ifpackageloaded{amsmath}{}{\AddToHook{package/amsmath/before}{#1}}}
\def\eql@amsmath@undefine#1{\eql@amsmath@before{\let#1\@undefined}}
\def\eql@amsmath@let#1#2{\eql@amsmath@undefine#1\let#1#2}
```

TODO: temporary fix for development stages

```
\@ifpackageloaded{amsmath}{}{%
  \DeclareHookRule{package/amsmath/after}%
    {eqnlines}{after}{latex-lab-testphase-math}}
```

2.7 PDF Tagging Support

`\eql@tagging@...`

Proper PDF tagging[2] support requires a LaTeX version at least of 2025. For the time being, we define an abstraction layer so that the package will collaborate with LaTeX versions around 2020: **TODO:** adjust to further developments

```
\let\eql@tagging@on\eql@false
\IfFormatAtLeastTF{2025-06-01}{%
  \csname tag_if_active:T\endcsname{\let\eql@tagging@on\eql@true}}{}
```

---

[2] see `https://latex3.github.io/tagging-project/`

```
\ifdefined\eql@tagging@on
  \def\eql@tagging@mathsave{%
    \UseTaggingSocket{math/luamml/save/nNn}{{}\displaystyle{mtd}}}
  \def\eql@tagging@mathaddlast{%
    \UseTaggingSocket{math/luamml/mtable/finalizecol}{last}}
  \def\eql@tagging@tagbegin{%
    \UseTaggingSocket{math/display/tag/begin}}
  \def\eql@tagging@tagend{%
    \UseTaggingSocket{math/display/tag/end}}
  \def\eql@tagging@tagsave{%
    \UseTaggingSocket{math/luamml/mtable/tag/save}}
  \def\eql@tagging@tagaddbox{%
    \setbox\z@\copy\eql@tagbox@%
    \UseTaggingSocket{math/luamml/mtable/tag/set}}
  \def\eql@tagging@tablesaveinner{%
    \UseExpandableTaggingSocket{math/luamml/mtable/innertable/save}}
  \def\eql@tagging@tableaddinner{%
    \UseTaggingSocket{math/luamml/mtable/innertable/finalize}}
  \def\eql@tagging@tablesavelines{%
    \UseExpandableTaggingSocket{math/luamml/mtable/finalize}{gather}}
  \def\eql@tagging@tablesavealign{%
    \UseExpandableTaggingSocket{math/luamml/mtable/finalize}{align}}
  \def\eql@tagging@alignleft{%
    \UseTaggingSocket{math/luamml/mtable/aligncol}{left}}
  \def\eql@tagging@aligncenter{%
    \UseTaggingSocket{math/luamml/mtable/aligncol}{center}}
  \def\eql@tagging@alignright{%
    \UseTaggingSocket{math/luamml/mtable/aligncol}{right}}
```

We need to get hold of the equation body in all cases so that we can feed it into the tagging mechanism:

```
  \let\eql@single@doscan\eql@true
  \let\eql@scan@body\eql@scan@body@rescan
```

`\eql@tagging@start`
`\eql@tagging@end`

We need to activate tagging for display equations for environments and for enclosures `\[`...`\]` and `\<`...`\>`. The tagging interface registration macro `\RegisterMathEnvironment` will work only partially for our cases, hence we replicate code from `\math_register_halign_env:nn`. Make sure collection is not yet active (`\l__math_collected_bool`). Then feed collected environment name, options and body into `\__math_process:nn`. Indicate the start of a display equation:

```
  \def\eql@tagging@start{%
    \csname bool_if:NF\expandafter\endcsname
      \csname l__math_collected_bool\endcsname{%
      \edef\eql@tmp{{\@currenvir}{[\unexpanded\expandafter{\eql@tagging@opt}]%
        \the\eql@scan@reg@}}%
      \csname __math_process:nn\expandafter\endcsname\eql@tmp
      \@kernel@math@registered@begin
      \csname bool_set_true:N\expandafter\endcsname
        \csname l__math_collected_bool\endcsname
    }%
  }
  \def\eql@tagging@end{}
  \def\eql@tagging@register@luamml#1{%
    \AddToHook{package/luamml/after}{%
      \eql@letcs{c__luamml_label_#1_tl}\@empty}}
  \def\eql@tagging@register@env{\csname math_register_env:n\endcsname}
```

When tagging is deactivated, provide empty definitions:

```
\else
  \let\eql@tagging@mathsave\@empty
  \let\eql@tagging@mathaddlast\@empty
  \let\eql@tagging@tagbegin\@empty
  \let\eql@tagging@tagend\@empty
  \let\eql@tagging@tagsave\@empty
  \let\eql@tagging@tagaddbox\@empty
  \let\eql@tagging@tablesaveinner\@empty
  \let\eql@tagging@tableaddinner\@empty
  \let\eql@tagging@tablesavelines\@empty
  \let\eql@tagging@tablesavealign\@empty
  \let\eql@tagging@alignleft\@empty
  \let\eql@tagging@aligncenter\@empty
  \let\eql@tagging@alignright\@empty
  \let\eql@tagging@start\@empty
  \let\eql@tagging@end\@empty
  \let\eql@tagging@register@luamml\@gobble
  \let\eql@tagging@register@env\@gobble
\fi
```

2.8 Key-Value Processing

The package uses the keyval mechanism to parse key-value pairs to specify adjustments to the behaviour of the equations environments:

```
\RequirePackage{keyval}
```

Value Selection.

`\eql@decide@select` Some parameter values take values in a given set, e.g. `true` vs. `false` or `left` vs. `right`. The macro `\eql@decide@select` is a general purpose selector. Arguments `#1` and `#2` describe the category and key which are used only towards error messages. Argument `#3` contains the value and argument `#4` is a list of values and corresponding actions in the format

`{{{`*val1a*`,`*val1b*`,` . . . `}{`*act1*`},{{`*val2a*`,`*val2b*`,` . . . `}{`*act2*`},` . . . `}`.

The (single) value `\relax` matches everything (can be used for handling generic values after specific ones). If no corresponding value is found in the list, an error message is invoked. Single expansion is applied to the list of values:

```
\def\eql@decide@relax{\eql@tmpb:=\relax}
\def\eql@decide@select#1#2#3#4{%
  \def\eql@tmpa{#3}%
  \let\eql@tmpd\@undefined
  \@for\eql@tmpc:=#4\do{%
    \ifdefined\eql@tmpd\else
      \edef\eql@tmpb{\noexpand\eql@tmpb:=\expandafter\@firstoftwo\eql@tmpc}%
      \ifx\eql@tmpb\eql@decide@relax
        \let\eql@tmpa\eql@relax
      \fi
      \expandafter\@for\eql@tmpb\do{%
        \ifx\eql@tmpa\eql@tmpb
          \edef\eql@tmpd{\unexpanded\expandafter\expandafter\expandafter{%
            \expandafter\@secondoftwo\eql@tmpc}}%
        \fi
      }%
  \fi
}%
\ifdefined\eql@tmpd
    \eql@tmpd
  \else
    \eql@error{undefined value ’#3’ for option ’#2’ of ’#1’}%
  \fi
}
```

Decide between `true` and `false` or related pairs of values:

```
\def\eql@decide@true{on,true,yes,enabled}
\def\eql@decide@false{off,false,no,disabled}
```

`\eql@decide@if`

```
\def\eql@decide@if#1#2#3#4#5{%
  \eql@decide@select{#1}{#2}{#3}{%
    {\eql@decide@true{#4}},%
    {\eql@decide@false{#5}}}}
```

`\eql@decide@bool` Store a boolean value into a conditional register:

```
\def\eql@decide@bool#1#2#3#4{%
  \eql@decide@if{#1}{#2}{#3}{\let#4\eql@true}{\let#4\eql@false}}
```

Key Declaration.

`\eql@define@key` For convenience, we define a wrapper for keyval's `\define@key` which accepts lists of categories and keys. We prepend the prefix `eql@` to all our categories so that we can hide it from the user in error messages:

```
\def\eql@define@key#1#2{%
  \eql@ifnextchar@loose[%]
    {\eql@definekey@opt{#1}{#2}}%
    {\eql@definekey@noopt{#1}{#2}}%
}
\def\eql@definekey@noopt#1#2#3{\eql@definekey@for{#1}{#2}{{#3}}}
\def\eql@definekey@opt#1#2[#3]#4{\eql@definekey@for{#1}{#2}{[#3]{#4}}}
\def\eql@definekey@for#1#2#3{%
  \def\eql@for@fn##1##2##3{\define@key{eql@##3}{##2}#3}%
  \edef\eql@for@vara{\noexpand\eql@for@vara:=#1}%
  \expandafter\@for\eql@for@vara\do{%
    \edef\eql@for@varb{\noexpand\eql@for@varb:=#2}%
    \expandafter\@for\eql@for@varb\do{%
      \edef\eql@for@call##1{%
        \noexpand\eql@for@fn{##1}{\eql@for@varb}{\eql@for@vara}}%
      \eql@for@call{##1}%
    }%
  }%
}
```

`\eql@setkeys` Our wrapper of keyval's `\setkeys` prepends the prefix `eql@` to the category, and it expands the list argument once:

```
\def\eql@setkeys#1#2{%
  \def\eql@tmp{\setkeys{eql@#1}}%
  \expandafter\eql@tmp\expandafter{#2}%
}
```

Options and Control Interface.

`\eql@nextopt` `\eql@nextopt@process` It can be convenient to add arguments to the following equations environment, e.g. towards defining modifier macros:

```
\let\eql@nextopt\@empty
\def\eql@nextopt@process#1{%
  \eql@setkeys{#1}\eql@nextopt
  \let\eql@tagging@opt\eql@nextopt
  \global\let\eql@nextopt\@empty
}
```

`\eqnaddopt`

```
\newcommand{\eqnaddopt}[1]{%
  \ifx\eql@nextopt\@empty
    \eql@append\eql@nextopt{#1}%
  \else
    \eql@append\eql@nextopt{,#1}%
  \fi
}
```

`\eqnlinesset` Process global configuration options including the package options:

```
\newcommand{\eqnlinesset}[1]{%
  \eql@setkeys{setup}{#1}%
  \ignorespaces
}
```

`\eql@control@default`

```
\protected\def\eql@control@default{%
  \eql@warn@here\eqncontrol
  \@gobble
}
\let\eqncontrol\eql@control@default
```

`\eqncontrol` Macro for general-purpose control within equations using key-value pairs:

```
\newcommand{\eql@control}[1]{%
  \relax
  \eql@setkeys{control}{#1}%
  \ignorespaces
}
```

2.9 Scanning the Equation Body

The multi-line equatiuon environment must scan its body twice: once to determine how wide the columns are and then to actually typeset them. This means that we must collect all text in this body before calling the environment macros. The mechanism and its description follows amsmath closely.

Token Register.

`\eql@scan@reg@` We start by defining a token register to hold the equation body.

```
\newtoks\eql@scan@reg@
```

`\eql@scan@body@dump`
`eql@scan@body@rescan`
`\eql@scan@body`

The macro `\eql@scan@body@dump` dumps the equation body from the register so that we do not have to pass it around in arguments. The macro `\eql@scan@body@rescan` rescans the tokens so that special commands such as `\verb` can be processed properly. The register `\eql@scan@body` holds the currently selected mode of operation: **TODO:** may skip expandafter before scantokens?

```
\def\eql@scan@body@dump{\the\eql@scan@reg@}
\def\eql@scan@body@rescan{%
  \expandafter\scantokens\expandafter{\the\eql@scan@reg@}}
\let\eql@scan@body\eql@scan@body@dump
```

`\eql@scan@addto`

We define a macro to append to the token register `\eql@scan@reg@`:

```
\long\def\eql@scan@addto#1{\eql@scan@reg@\expandafter{\the\eql@scan@reg@#1}}
```

Environment Body. The following mechanism scans the contents of an environment taking into account nested environments that may be contained in the body.

`\eql@scan@env`

The macro `\eql@scan@env` starts the scan for the `\end{...}` command of the current environment. The argument is a call-back macro to process the body in `\eql@scan@reg@`:

```
\def\eql@scan@env{%
  \eql@verbose@infoarg\eql@verbose@msg@enter\eql@scan@env
  \def\eql@scan@end{\expandafter\end\expandafter{\@currenvir}}%
  \eql@scan@reg@{}\def\eql@scan@stack{b}%
```

We call `\eql@scan@env@iterate` which will scan until the next occurrence of `\end` and then count the number of occurrences of `\begin` before `\end` in `\eql@scan@stack`. If we simply called `\eql@scan@env@iterate` directly, the error message for an unwanted `\par` token (usually from a blank line) would refer to `\eql@scan@env@iterate` which would not be illuminating. We use a little finesse to get a more intelligible error message: We use the actual environment name as the name of the temporary function that is `\let` to `\eql@scan@env@iterate`:

```
  \edef\eql@scan@iterate{\expandafter\noexpand\csname\@currenvir\endcsname}%
  \expandafter\let\expandafter\eql@scan@env@org\eql@scan@iterate
  \ifdefined\eql@scan@par
    \expandafter\let\eql@scan@iterate\eql@scan@env@iterate
  \else
    \expandafter\let\eql@scan@iterate\eql@scan@env@iterate@nopar
  \fi
  \eql@scan@iterate
}
```

`eql@scan@env@iterate`

`\eql@scan@env@iterate` takes two arguments: the first will consist of all text up to the next `\end` command, the second will be the `\end` command's argument. If there are any extra `\begin` commands in the body text, a marker is pushed onto a stack via `\eql@scan@env@count`. An empty state for this stack means that we have reached the `\end` that matches our original `\begin`. Otherwise we need to include the `\end` and its argument in the material that we are adding to our environment body accumulator:

```
\long\def\eql@scan@env@iterate#1\end#2{%
  \edef\eql@scan@stack{%
    \eql@scan@env@count#1\begin\end\expandafter\@gobble\eql@scan@stack}%
  \ifx\@empty\eql@scan@stack
    \@checkend{#2}%
    \eql@scan@addto{#1}%
    \expandafter\let\eql@scan@iterate\eql@scan@env@org
    \eql@verbose@infoarg\eql@verbose@msg@leave\eql@scan@env
    \expandafter\eql@scan@call
  \else
    \eql@scan@addto{#1\end{#2}}%
    \expandafter\eql@scan@iterate
  \fi
}
```

`an@env@iterate@nopar` Version of `\eql@scan@env@iterate` which does not accept `\par` within the argument:

```
\def\eql@scan@env@iterate@nopar#1\end#2{\eql@scan@env@iterate#1\end{#2}}
```

`\eql@scan@env@count` When adding a piece of the current environment's contents to `\eql@scan@reg@`, we scan it to check for additional `\begin` tokens, and add a '`b`' to the stack for any that we find.

```
\long\def\eql@scan@env@count#1\begin#2{%
  \ifx\end#2\else b\expandafter\eql@scan@env@count\fi
}
```

The call-back macro `\eql@scan@env@cancel` ignores the body as well as the end clause for the environment:

```
\def\eql@scan@env@cancel{%
  \@namedef{end\@currenvir}{\ignorespacesafterend}%
  \eql@scan@end
}
```

Square Brackets. The following is a version of the above mechanism that scans for an equation body enclosed by `\[`...`\]` paying attention to potential further instances of the square bracket enclosures contained in the body.

`\eql@scan@sqr` Start scanning for `\]`:

```
\def\eql@scan@sqr{%
  \eql@verbose@infoarg\eql@verbose@msg@enter\eql@scan@sqr
  \def\eql@scan@end{\]}%
  \eql@scan@reg@{}\def\eql@scan@stack{b}%
  \let\eql@scan@sqr@org\[%\]
  \ifdefined\eql@scan@par
    \let\[\eql@scan@sqr@iterate%\]
  \else
    \let\[\eql@scan@sqr@iterate@nopar%\]
  \fi
  \[%\]
}
```

`eql@scan@sqr@iterate` Iterate until we find a balanced pairing of square brackets. Then call the call-back macro:

```
\long\def\eql@scan@sqr@iterate#1\]{%
  \edef\eql@scan@stack{%
    \eql@scan@sqr@count#1\[\]\expandafter\@gobble\eql@scan@stack}%
  \ifx\@empty\eql@scan@stack
    \let\[\eql@scan@sqr@org%\]
    \eql@scan@addto{#1}%
    \eql@verbose@infoarg\eql@verbose@msg@leave\eql@scan@sqr
    \expandafter\eql@scan@call
  \else
```

```
    \eql@scan@addto{#1\]}%
    \expandafter\[%\]
  \fi
}
```

`an@sqr@iterate@nopar` Version of `\eql@scan@sqr@iterate` which does not accept `\par` within the argument:

```
\def\eql@scan@sqr@iterate@nopar#1\]{\eql@scan@sqr@iterate#1\]}
```

`\eql@scan@sqr@count` Push a ‘`b`’ for every encountered instance of ‘`\[`’:

```
\long\def\eql@scan@sqr@count#1\[#2{%\]
  \ifx\]#2\else b\expandafter\eql@scan@sqr@count\fi
}
```

`l@scan@sqrang@cancel` The call-back macro `\eql@scan@sqrang@cancel` ignores the body and the closing bracket:

```
\def\eql@scan@sqrang@cancel{\ignorespaces}
```

Angle Brackets. The following is another version of the mechanism which scans for an equation body enclosed by `\<`...`\>`.

`\eql@scan@ang` Start scanning for `\>`:

```
\def\eql@scan@ang{%
  \eql@verbose@infoarg\eql@verbose@msg@enter\eql@scan@ang
  \def\eql@scan@end{\>}%
  \eql@scan@reg@{}\def\eql@scan@stack{b}%
  \let\eql@scan@ang@org\<%\>
  \ifdefined\eql@scan@par
    \let\<\eql@scan@ang@iterate%\>
  \else
    \let\<\eql@scan@ang@iterate@nopar%\>
  \fi
  \<%\>
}
```

`eql@scan@ang@iterate` Iterate until we find a balanced pairing of angle brackets:

```
\long\def\eql@scan@ang@iterate#1\>{%
  \edef\eql@scan@stack{%
    \eql@scan@ang@count#1\<\>\expandafter\@gobble\eql@scan@stack}%
  \ifx\@empty\eql@scan@stack
    \let\<\eql@scan@ang@org%\>
    \eql@scan@addto{#1}%
    \eql@verbose@infoarg\eql@verbose@msg@leave\eql@scan@ang
    \expandafter\eql@scan@call
  \else
    \eql@scan@addto{#1\>}%
    \expandafter\<%\>
  \fi
}
```

`an@ang@iterate@nopar` Version of `\eql@scan@ang@iterate` which does not accept `\par` within the argument:

```
\def\eql@scan@ang@iterate@nopar#1\>{\eql@scan@ang@iterate#1\>}
```

`\eql@scan@ang@count` Push a ‘`b`’ for every encountered instance of ‘`\<`’:

```
\long\def\eql@scan@ang@count#1\<#2{%\>
  \ifx\>#2\else b\expandafter\eql@scan@ang@count\fi
}
```

Interface. **TODO:** describe

`\eql@sqr@open` If already in math mode, ignore the enclosed contents.

```
\protected\def\eql@sqr@open{%
  \ifmmode
    \expandafter\eql@sqr@cancel
  \else
    \expandafter\eql@equations@sqr@open
  \fi
}
```

`\eql@sqr@close` Definition for ‘`\]`’: **TODO:** NOTE: `\protected` acts as `\relax` and starts a row in `\halign`, so we overwrite `\]` when starting.

```
\protected\def\eql@sqr@close{%
  \eql@error{'\string\]' may only close '\string\['}%\]
}
```

`\eql@sqr@cancel`

```
\def\eql@sqr@cancel{%
  \eql@error@mathmode{\string\[...\string\]}%
  \let\eql@scan@call\eql@scan@sqrang@cancel
  \eql@scan@sqr
}
```

`\eql@ang@open` Definition for ‘`\<`’. Forward to `equationsbox` if in math mode, otherwise to `equations`:

```
\protected\def\eql@ang@open{%
  \ifmmode
    \expandafter\eql@box@ang@open
  \else
    \expandafter\eql@equations@ang@open
  \fi
}
```

`\eql@ang@close` Definition for ‘`\>`’: **TODO:** NOTE: `\protected` acts as `\relax` and starts a row in `\halign`, so we overwrite `\>` when starting.

```
\protected\def\eql@ang@close{%
  \eql@error{'\string\>' may only close '\string\<'}%\>
}
```

3 Parameters and Registers

In the following, we collect parameter and register definitions.

3.1 Parameters

TODO: describe

TODO: maybe sort parameters into sections **TODO:** or sort parameters in sections here

`\eql@tagsleft` (*bool*)

The boolean parameter `\eql@tagsleft` specifies whether the tags are placed at the left or right margin:

```
\let\eql@tagsleft\eql@false
```

`eql@layoutleft` (*bool*)

The boolean parameter `\eql@layoutleft` specifies whether to use left or central alignment layout:

```
\let\eql@layoutleft\eql@false
```

`eql@layoutleftmargin`
`@layoutleftmarginmin`
`@layoutleftmarginmax`

The default width of the left margin in left alignment layout is specified by `\eql@layoutleftmargin`. It may be pushed down to `\eql@layoutleftmarginmin` and up to `\eql@layoutleftmarginmax`:

```
\def\eql@layoutleftmargin{\leftmargini}
\def\eql@layoutleftmarginmax{.5\maxdimen}
\def\eql@layoutleftmarginmin{\z@}
```

`\eql@mathstyle`

The math style to be used within cells is specified by `\eql@mathstyle`:

```
\let\eql@mathstyle\displaystyle
```

`l@tagmargin@` (*dimen*)
`argin@ratio@` (*dimen*)
`\eql@tagmargin@val`
`@tagmargin@threshold`

The intended margin width for tags in centeral alignment layout is stored in `\eql@tagmargin@` which is sourced by `\eql@tagmargin@val`. An undefined `\eql@tagmargin@val` will compute the margin width as the maximum width of tags (without separation). `\eql@tagmargin@ratio@` describes the maximum ratio of lines with tags to total number of lines for which `\eql@tagmargin@` is set to zero: **TODO:** threshold

```
\newdimen\eql@tagmargin@
\let\eql@tagmargin@val\@undefined
\newdimen\eql@tagmargin@ratio@
\eql@tagmargin@ratio@\p@
\def\eql@tagmargin@threshold{0.5}
```

`\eql@indent@` (*dimen*)

The currently selected indentation width is speficied by `\eql@indent@`. This dimension register is set to the macro `\eql@indent@val` when entering the equation environments:

```
\newdimen\eql@indent@
\def\eql@indent@val{2em}
```

`paddingleft@` (*dimen*)
`addingright@` (*dimen*)

The padding of an equation (column) is specified by `\eql@paddingleft@` and `\eql@paddingright@`. These dimension registers are set to the macros `\eql@paddingleft@val` and `\eql@paddingright@val`, respectively, when entering the equation environments:

```
\newdimen\eql@paddingleft@
\newdimen\eql@paddingright@
\let\eql@paddingleft@val\@undefined
\let\eql@paddingright@val\@undefined
```

`ql@display@linewidth`
`l@display@marginleft`
`@display@marginright`

TODO: describe

```
\let\eql@display@linewidth\@undefined
\let\eql@display@marginleft\@undefined
\let\eql@display@marginright\@undefined
```

`\eql@box@colsep`
`\eql@box@shortsep`
`\eql@box@condsep`
`\eql@box@matrixsep`

The macro `\eql@box@colsep` specifies the intercolumn separation for equation boxes: **TODO:** describe

```
\def\eql@box@colsep{2em}
\def\eql@box@shortsep{1em}
\def\eql@box@condsep{\eql@box@shortsep}
\def\eql@box@matrixsep{\eql@box@shortsep}
```

`\eql@break@line@sep`
`@break@line@shortsep`
`\eql@break@col@sep`
`l@break@col@shortsep`

TODO: describe

```
\def\eql@break@line@sep{2em minus 1em}
\def\eql@break@line@shortsep{1em}
\def\eql@break@col@sep{2em minus 1em}
\def\eql@break@col@shortsep{1em}
```

`\eql@spread@val`

The extra spread of equation lines is specified by `\eql@spread@val`:

```
\let\eql@spread@reset\eql@false
\def\eql@spread@val{\jot}
\newdimen\eql@spread@
```

`eql@tagfuzz@` (*dimen*)

The value `\eql@tagfuzz@` specifies the margin of error for comparing whether a tag fits a given equation line. We should not expect rounding errors in the fixed point arithmetic of TeX, nevertheless: **TODO:** probably do not need this due to fixed point arithmetic.

```
\newdimen\eql@tagfuzz@
\eql@tagfuzz@16sp\relax
```

`\eql@display@height`
`\eql@display@depth`

An equation will appear to the surrounding text with a fixed apparent height and depth specified by `\eql@display@height` and `\eql@display@depth`, respectively:

```
\def\eql@display@height\@undefined
\def\eql@display@depth\@undefined
```

`\eql@skip@mode@short`

The setting `\eql@skip@mode@short` specifies when a reduced amount of glue should be used around equations in case the text line above the equation fits in the space that is left available in the first equation line. Value `0` turns this feature off, value `1` reduces the glue above the equation, value `2` furthermore reduces the glue below a single equation line and value `3` also reduces the glue below multi-line equations:

```
\def\eql@skip@mode@short{2}

\def\eql@skip@mode@cont@above{2}
\def\eql@skip@mode@cont@below{0}

\def\eql@skip@mode@par@above{3}
\def\eql@skip@mode@par@below{0}

\def\eql@skip@mode@top@above{4}
\def\eql@skip@mode@top@below{0}

\newcount\eql@skip@mode@leave@
\let\eql@skip@force@leave\@undefined
```

`eql@skip@force@above` `eql@skip@force@below` `mode@above@` (*counter*) `mode@below@` (*counter*) 0: short, 1: long, 2: cont, 3: par, 4: top, 5: no, 6: med, 7: custom

```
\newcount\eql@skip@mode@above@
\newcount\eql@skip@mode@below@
\let\eql@skip@force@above\@undefined
\let\eql@skip@force@below\@undefined
\let\eql@skip@custom@above\@undefined
\let\eql@skip@custom@below\@undefined
```

`\eql@skip@cont@above` The glue when an equation is at the top of a horizontal list is specified by `\eql@skip@cont@above`:

`\eql@skip@top@above` `\eql@skip@top@below` The glue when an equation is at the top of a vertical list is specified by `\eql@skip@top@above` and `\eql@skip@top@below`:

`\eql@skip@par@above` The glue when an equation starts a paragraph is specified by `\eql@skip@par@above`:

`\eql@skip@med@above` `\eql@skip@med@below` The surrounding glue for an equation with reduced spacing is given by `\eql@skip@med@above` and `\eql@skip@med@below`:

```
\def\eql@skip@long@above{\abovedisplayskip}
\def\eql@skip@long@below{\belowdisplayskip}
\def\eql@skip@short@above{\abovedisplayshortskip}
\def\eql@skip@short@below{\belowdisplayshortskip}
\def\eql@skip@cont@above{\eql@skip@short@above}
\def\eql@skip@cont@below{\eql@skip@short@below}
\def\eql@skip@par@above{\eql@skip@long@above}
\def\eql@skip@par@below{\eql@skip@long@below}
\def\eql@skip@top@above{\eql@skip@long@above}
\def\eql@skip@top@below{\eql@skip@long@below}
\def\eql@skip@med@above{\abovedisplayskip/2}
\def\eql@skip@med@below{\belowdisplayskip/2}
\def\eql@skip@tag@above{\z@skip}
\def\eql@skip@tag@below{\z@skip}
```

`l@colsepmin@` (*dimen*) `\eql@colsepmin@val` `\eql@colsepmax@val` The minimum intercolumn separatation is specified by `\eql@colsepmin@`. This dimension register is set to `\eql@colsepmin@val` when entering the equation environments to allow font-dependent values. Furthermore, `\eql@colsepmax@val` specifies the maximum intercolumn separation:

```
\newdimen\eql@colsepmin@
\def\eql@colsepmin@val{1em}
\def\eql@colsepmax@val{.5\maxdimen}
```

`@tagwidthmin@` (*dimen*) The minimum tag width is specified by `\eql@tagwidthmin@`:

```
\newdimen\eql@tagwidthmin@
\eql@tagwidthmin@\z@
```

`l@tagsepmin@` (*dimen*) The minimum separation between an equation and its tag is given by `\eql@tagsepmin@`. TEX's built-in value is half a quad[3] in font number 2. As the tag is processed in text mode, we use `0.5em` instead.

```
\newdimen\eql@tagsepmin@
\def\eql@tagsepmin@val{.5\fontdimen6\textfont\tw@}
```

---

[3] another half of a quad is left empty at the other end of the line.

ql@equations@sqr@opt
ql@equations@ang@opt
\eql@box@ang@opt

Store the default arguments for `\[...\]` and `\<...\>`, respectively:

```
\def\eql@equations@sqr@opt{equation}
\def\eql@equations@ang@opt{columns}
\def\eql@box@ang@opt{columns}
```

Multi-Line Align Mode.

```
\let\eql@columns@fulllength\eql@false
```

3.2 Registers

TODO: describe

General. **TODO:** describe

```
\newcount\eql@count@
\newdimen\eql@dimen@
\newskip\eql@skip@
```

TODO: describe

```
\let\eql@display@container\@empty
```

\eql@cellbox@ (*box*)
\eql@tagbox@ (*box*)
ql@cellwidth@ (*dimen*)
ql@prevwidth@ (*dimen*)
eql@tagwidth@ (*dimen*)
ql@prevdepth@ (*dimen*)
l@prevgraf@ (*counter*)

The box `\eql@cellbox@` holds the present alignment component and `\eql@tagbox@` the tag for the present line. The corresponding dimensions `\eql@cellwidth@` and `\eql@tagwidth@` hold their widths. `\eql@prevwidth@` holds the width of the previous alignment component: **TODO:** adjust

```
\newbox\eql@cellbox@
\newbox\eql@tagbox@
\newdimen\eql@cellwidth@
\newdimen\eql@prevwidth@
\newdimen\eql@tagwidth@
\newdimen\eql@prevdepth@
\newcount\eql@prevgraf@
```

l@totalwidth@ (*dimen*)
agwidth@max@ (*dimen*)
@totalheight@ (*dimen*)

```
\newdimen\eql@totalwidth@
\newdimen\eql@tagwidth@max@
\newdimen\eql@totalheight@
\newdimen\eql@topheight@
\newdimen\eql@bottomdepth@
```

@line@height@ (*dimen*)
l@line@depth@ (*dimen*)

The dimension registers `\eql@line@height@` and `\eql@line@depth@` keep track of the height and depth of the present line in an alignment:

```
\newdimen\eql@line@height@
\newdimen\eql@line@depth@
```

l@line@width@ (*dimen*)
l@line@avail@ (*dimen*)
eql@line@pos@ (*dimen*)
e@widthsep@ (*counter*)
e@availsep@ (*counter*)
ine@possep@ (*counter*)
@line@offset@ (*dimen*)
e@prevdepth@ (*dimen*)
e@interline@ (*dimen*)

```
\newdimen\eql@line@width@
\newdimen\eql@line@avail@
\newdimen\eql@line@pos@
```

```
\newcount\eql@line@availsep@
\newcount\eql@line@widthsep@
\newcount\eql@line@possep@
\newdimen\eql@line@offset@
\newdimen\eql@line@prevdepth@
\newdimen\eql@line@interline@
```

Rows and Columns.

`\eql@row@` (*counter*)
`l@totalrows@` (*counter*)
`ql@tagrows@` (*counter*)

TODO: tagrows `\eql@row@` counts the present row (1-based) and `\eql@totalrows@` holds the total number of rows:

```
\newcount\eql@row@
\newcount\eql@totalrows@
\newcount\eql@tagrows@
```

`\eql@column@`
`\eql@totalcolumns@`

```
\newcount\eql@column@
\newcount\eql@totalcolumns@
```

`\eql@colsep@` (*dimen*)

The dimension of the intercolumn separation for align environments is stored in `\eql@colsep@`:

```
\newdimen\eql@colsep@
```

`tercolumns@` (*counter*)

```
\newcount\eql@intercolumns@
```

Vertical Spacing Adjustments.

`y@firstavail@` (*dimen*)
`splay@firstavail@set`

The unused space on the first line of an alignment is stored in `\eql@display@firstavail@` for comparison against `\predisplaysize` and determining short skip mode of display equations. It it convenient to set it via `\eql@display@firstavail@set` provided that we are on the first line:

```
\newdimen\eql@display@firstavail@
\def\eql@display@firstavail@set#1{%
  \ifnum\eql@row@=\@ne
    \global\eql@appendexpand\eql@display@container{%
      \eql@display@firstavail@\the#1\relax}%
  \fi
}
```

The counter stores whether the tag one first/last line is raised/lowered as `1`/`2` (or `3` for both). This implies a different vskip corresponding to the mostly empty line: **TODO:** adjust

```
\newdimen\eql@display@aboveextend@
\newdimen\eql@display@belowextend@
```

Shared Registers.

`\ifmeasuring@` (*bool*)

All display environments get typeset twice – once during a “measuring” phase and then again during a “production” phase. We reuse the original `amsmath` definition

`\ifmeasuring@` to determine which case we're in, so we and other packages may take appropriate action. It does not hurt to define this conditional in any case. We should tell hyperref about measuring processes as we're not amsmath and not being catered for:

```
\ifdefined\measuring@true\else
  \expandafter\newif\csname ifmeasuring@\endcsname
\fi
\AddToHook{package/hyperref/after}{%
  \ifdefined\Hy@ifnotmeasuring
    \renewcommand\Hy@ifnotmeasuring[1]{\ifmeasuring@\else#1\fi}%
  \fi
}
```

`\if@display` (*bool*) amsmath defines the conditional `\if@display` to test whether we're in a display equation including the inner math parts of equation blocks. We provide it in case amsmath is absent, and initialise it:

```
\ifdefined\@displaytrue\else
  \expandafter\newif\csname if@display\endcsname
  \everydisplay\expandafter{\the\everydisplay\@displaytrue}
\fi
```

3.3 Hooks

`\eql@hook@...` For what it's worth, we define a couple of entry points where one might hook into the equations typesetting framework. The LaTeX hook framewould would be more versatile, but as the purpose of these hooks is rather unclear at the moment, we make this as efficient as it could get: **TODO:** may add a few more hooks

```
\let\eql@hook@blockbefore\@empty
\let\eql@hook@blockafter\@empty
\let\eql@hook@blockin\@empty
\let\eql@hook@blockout\@empty
\let\eql@hook@linein\@empty
\let\eql@hook@lineout\@empty
\let\eql@hook@colin\@empty
\let\eql@hook@colout\@empty
\let\eql@hook@eqin\@empty
\let\eql@hook@eqout\@empty
\let\eql@hook@innerleft\@empty
\let\eql@hook@innerright\@empty
\let\eql@hook@innerlead\@empty
```

4 Features

4.1 Punctuation

The equations environments supply an automatic punctuation scheme which allows to define a default punctuation at the end of each column, line and equation block.

`\eql@punct@col`
`\eql@punct@line`
`\eql@punct@block`
`\eql@punct@next`
`\eql@punct@top`
`\eql@punct@cases`

These macros store the punctuation character for columns, lines and blocks. An undefined value indicates that the punctuation should be handed down to the next lower level: **TODO:** update

```
\let\eql@punct@col\@empty
```

```
\let\eql@punct@line\@undefined
\let\eql@punct@block\@undefined
\let\eql@punct@next\@undefined
\let\eql@punct@top\@undefined
\let\eql@punct@cases\@empty
```

`\eql@punct@sep` This macro stores the separation to be applied before the punctuation (unless it is empty):

```
\let\eql@punct@class\@empty
\let\eql@punct@sep\@empty
```

`\eql@punct@top@set`
`\eql@punct@top@stop`
`\eql@punct@top@reset`

TODO: describe

```
\def\eql@punct@top@set{\let\eql@punct@top\eql@punct@block}
\def\eql@punct@top@stop{\let\eql@punct@top\relax}
\def\eql@punct@top@reset{\let\eql@punct@top\@undefined}
```

`\eql@punct@set` **TODO:** describe

```
\def\eql@punct@tilde{~}
\def\eql@punct@set#1#2{%
  \def#1{#2}%
  \ifx#1\eql@relax
    \let#1\@undefined
  \fi
  \ifx#1\eql@punct@tilde
    \let#1\@empty
  \fi
}
\def\eql@punct@clear{%
  \let\eql@punct@col\@empty
  \let\eql@punct@line\@empty
  \let\eql@punct@block\@empty
}
\def\eql@punct@next@clear{\let\eql@punct@next\@empty}
```

Set the punction for blocks. Note that the macro `\eqnpunct` sets the punctuation for the following equation block or for the current cell. Starred versions clear the punctuation for the respectively levels:

TODO: describe

```
\def\eql@punct@addopt{%
  \eql@ifstar@tight\eql@punct@addopt@relax\eql@punct@addopt@set}
\def\eql@punct@addopt@set#1{\eqnaddopt{punct={#1}}\ignorespaces}
\def\eql@punct@addopt@relax{\eqnaddopt{punct*}\ignorespaces}
```

TODO: describe

```
\def\eql@punct@setnext{%
  \eql@ifstar@tight\eql@punct@setnext@relax\eql@punct@setnext@set}
\def\eql@punct@setnext@set#1{%
  \eql@punct@set\eql@punct@next{#1}%
  \ifdefined\eql@punct@next\else\let\eql@punct@next\relax\fi
  \ignorespaces}
\def\eql@punct@setnext@relax{\let\eql@punct@next\relax\ignorespaces}
```

`\eqnpunct` **TODO:** describe

```
\let\eqnpunct\eql@punct@addopt
```

`\eql@punct@fill@next` Fill the next punctuation:

```
\def\eql@punct@fill@next#1{%
  \ifdefined\eql@punct@next
    \ifx\eql@punct@next\relax
      \let\eql@punct@next\@undefined
    \fi
  \else
    \ifx\eql@punct@top\relax\else
      \let\eql@punct@next#1%
    \fi
  \fi
}
```

`ql@punct@output@next` Output the next punctuation. If non-empty, prepend some separation:

```
\def\eql@punct@output@next{%
  \ifx\eql@punct@next\@empty\else
    \ifmmode\eql@punct@class\fi
    \eql@punct@sep
    \eql@punct@next
  \fi
  \let\eql@punct@next\@undefined
}
```

`eql@punct@print@next` Print the next punctuation if available:

```
\def\eql@punct@print@next{%
  \ifdefined\eql@punct@next
    \eql@punct@output@next
  \fi
}
```

`eql@punct@apply@next` Print the next punctuation if available. Stop further punctuation within the current group:

```
\def\eql@punct@apply@next{%
  \ifdefined\eql@punct@next
    \eql@punct@output@next
    \eql@punct@top@stop
  \fi
}
```

`\eql@punct@print@col` Print the punctuation for the present column:

```
\def\eql@punct@print@col{%
  \eql@punct@fill@next\eql@punct@col
  \eql@punct@print@next
}
```

`\eql@punct@apply@col` Output the punctuation for the present column. Stop further punctuation within the current group:

```
\def\eql@punct@apply@col{%
  \eql@punct@fill@next\eql@punct@col
  \eql@punct@apply@next
}
```

Output the punctuation for the present line unless disabled:

`eql@punct@print@line`

```
\def\eql@punct@print@line{%
  \eql@punct@fill@next\eql@punct@line
  \eql@punct@print@next
}
```

Output the punctuation for the present line unless disabled. Stop further punctuation within the current group:

`eql@punct@apply@line`

```
\def\eql@punct@apply@line{%
  \eql@punct@fill@next\eql@punct@line
  \eql@punct@apply@next
}
```

`ql@punct@apply@block` Output the punctuation for the present block unless disabled. Stop further punctuation within the current group:

```
\def\eql@punct@apply@block{%
  \eql@punct@fill@next\eql@punct@block
  \eql@punct@apply@next
}
```

`\eql@punct@apply@top` Output the top punctuation unless disabled. Stop further punctuation globally:

```
\def\eql@punct@apply@top{%
  \eql@punct@fill@next\eql@punct@top
  \eql@punct@print@next
  \global\eql@punct@top@stop
}
```

`\eqnpunctapply` Output the top punctuation unless disabled. Stop further punctuation globally:

```
\newcommand{\eqnpunctapply}{\ifmmode\else\unskip\fi\eql@punct@apply@top}
```

4.2 Math Classes at Alignment

The following describes the adjustment of math classes surrounding the alignment marker.

`lass@innerright@sel@` Select between `\eql@class@innerlead` and `\eql@class@innerright` depending on whether the left part of the aligned column is empty:

```
\def\eql@class@innerright@sel@{%
  \ifdim\eql@prevwidth@=\z@
    \eql@class@innerlead
  \else
    \eql@class@innerright
  \fi
}
```

`@class@innerleft@set`
`class@innerright@set`
`@class@innerlead@set`

Set the left, right and leading math classes. Setting the right math class disables the leading math class, so the leading math class must be specified after the right one:

```
\def\eql@class@innerleft@set#1{%
  \def\eql@class@innerleft{#1}%
}
```

```
\def\eql@class@innerright@set#1{%
  \def\eql@class@innerright{#1}%
  \let\eql@class@innerright@sel\eql@class@innerright
}
\def\eql@class@innerlead@set#1{%
  \def\eql@class@innerlead{#1}%
  \let\eql@class@innerright@sel\eql@class@innerright@sel@
}
```

```
\def\eql@class@rel@symb{=}
\def\eql@class@rel@break#1{\eql@class@rel@start{#1}\mathclose{}}
\def\eql@class@rel@start#1{\mathrel{\phantom{#1}}}
\def\eql@class@rel@break@set#1{\def\eql@class@rel@break##1{#1}}
\def\eql@class@rel@start@set#1{\def\eql@class@rel@start##1{#1}}
\def\eql@class@rel@break@skip{\thickmuskip-\medmuskip}
\def\eql@class@rel@amp{&}
\def\eql@class@rel@amprelax{&\relax}
\def\eql@class@rel@relaxamp{\relax&}
\def\eql@class@rel@ordamp{{}&}
\def\eql@class@rel@make#1{%
  \ifdefined\eql@class@rel@after
    \def\eql@tmp{&#1}%
    \ifx\eql@tmp\eql@class@rel@amp
      \def\eql@tmp{&\eql@class@rel@break\eql@class@rel@symb
        \mskip\muexpr\eql@class@rel@break@skip\relax}%
    \else\ifx\eql@tmp\eql@class@rel@amprelax
      \def\eql@tmp{&\eql@class@rel@start\eql@class@rel@symb}%
    \fi\fi
  \else
    \def\eql@tmp{#1&}%
    \ifx\eql@tmp\eql@class@rel@amp
      \def\eql@tmp{&\mathclose{}%
        \mskip\muexpr\eql@class@rel@break@skip\relax}%
    \fi
    \ifx\eql@tmp\eql@class@rel@relaxamp
      \def\eql@tmp{&\mathclose{}\mathopen{}\mathrel{}}%
    \fi
    \ifx\eql@tmp\eql@class@rel@ordamp
      \def\eql@tmp{&\mathclose{}}%
    \fi
  \fi
  \eql@tmp
}
```

`\eql@class@ampeq`
`\eql@class@eqamp`

We define two standard combinations of math classes intended to be used with '`&=`' (ampeq) or '`=&`' (eqamp). The default setting is '`&=`' (ampeq):

```
\def\eql@class@ampeq{%
  \eql@class@innerleft@set{}%
  \eql@class@innerright@set{{}}%
  \let\eql@class@rel@after\eql@true
}
\def\eql@class@eqamp{%
  \eql@class@innerleft@set{}%
  \eql@class@innerright@set{\mathrel{}}%
  \eql@class@innerlead@set{\mathclose{}}%
  \let\eql@class@rel@after\eql@false
}
\eql@class@ampeq
```

4.3 Framed Cells

TODO: describe **TODO:** warn if issued in even cells

```
\let\eql@frame@cmd\@undefined
\newdimen\eql@frame@margin@
\def\eql@frame@set[#1]{%
  \global\eql@append\eql@cell@container{\def\eql@frame@cmd{#1}}}
\protected\def\framecell{\eql@testopt@tight@ampsafe\eql@frame@set\fbox}
\def\eql@frame@measure{%
  \setbox\z@\hbox{\eql@frame@cmd{}}%
  \eql@frame@margin@.5\wd\z@
}
\def\eql@frame@print{%
  \setbox\eql@cellbox@\hbox{%
    \eql@frame@cmd{\unhbox\eql@cellbox@}%
  }%
}
\def\eql@frame@adjust{%
  \setbox\eql@cellbox@\hbox{%
    \eql@frame@cmd{%
      \unhbox\eql@cellbox@
      \unkern
      \unskip
    }%
    \hfil
    \kern\z@
  }%
}
```

4.4 Single-Line Composition

TODO: describe

`\eql@break@line`
`\eql@break@col`

```
\def\eql@break@line{%
  \let\eql@break@sep\eql@break@line@sep
  \let\eql@break@shortsep\eql@break@line@shortsep
  \let\eql@break@print\eql@punct@print@line
  \let\eql@punct@term\eql@false
  \let\eql@class@rel@composed\@undefined
  \eql@ampprotect\eql@break@test\eql@break@process}
\def\eql@break@col{%
  \let\eql@break@sep\eql@break@col@sep
  \let\eql@break@shortsep\eql@break@col@shortsep
  \let\eql@break@print\eql@punct@print@col
  \let\eql@punct@term\eql@false
  \let\eql@class@rel@composed\@undefined
  \eql@ampprotect\eql@break@test\eql@break@process}
\def\eql@break@cr{%
  \let\eql@break@sep\eql@break@line@sep
  \let\eql@break@shortsep\eql@break@line@shortsep
  \let\eql@break@print\eql@punct@print@line
  \let\eql@punct@term\eql@false
  \let\eql@class@rel@composed\@undefined
  \eql@ampprotect\eql@break@cr@test\eql@break@process}
\def\eql@break@amp{%
```

```
  \eql@ampprotecttwo\eql@break@amp@testescape
    \eql@amp@org\eql@break@process}
\def\eql@break@amp@testescape#1#2{%
  \eql@ifnextgobble@tight/{#1}{\eql@break@amp@testnoescape{#2}}}
\def\eql@break@amp@testnoescape#1{%
  \relax
  \let\eql@break@sep\eql@break@col@sep
  \let\eql@break@shortsep\eql@break@col@shortsep
  \let\eql@break@print\eql@punct@print@col
  \let\eql@punct@term\eql@false
  \let\eql@class@rel@composed\@undefined
  \eql@break@amp@test{#1}}
```

TODO: describe

```
\def\eql@break@test@setopt{\let\eql@break@test\eql@break@testopt}
\def\eql@break@test@setall{\let\eql@break@test\eql@break@testall}
\def\eql@break@cr@test@setopt{\let\eql@break@cr@test\eql@break@testopt}
\def\eql@break@cr@test@setall{\let\eql@break@cr@test\eql@break@testall}
\def\eql@break@amp@test@setopt{\let\eql@break@amp@test\eql@break@testopt}
\def\eql@break@amp@test@setall{\let\eql@break@amp@test\eql@break@testall}
```

`\eql@break@testopt`

TODO: describe

```
\def\eql@break@testopt#1{\eql@ifstar@tight
  {\let\eql@break@sep\eql@break@shortsep#1}{\eql@break@testopt@arg{#1}}}
\def\eql@break@testopt@arg#1{\eql@ifnextchar@tight[%
  {\eql@break@testopt@set{#1}}{#1}}
\def\eql@break@testopt@set#1[#2]{\def\eql@break@sep{#2}#1}
```

`\eql@break@testall`
`@break@testall@parse`

TODO: describe

```
\def\eql@break@testall{\eql@parseopt@cr\eql@break@testall@parse}
\def\eql@break@testall@parse{%
  \ifx\eql@parseopt@token[%
    \let\eql@parseopt@next\eql@break@parse@val
  \fi
  \ifx\eql@parseopt@token*%
    \let\eql@parseopt@next\eql@break@parse@star
  \fi
  \ifx\eql@parseopt@token.%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token,%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token~%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token`%
    \let\eql@parseopt@next\eql@parseopt@punctnext
  \fi
  \ifx\eql@parseopt@token!%
    \let\eql@parseopt@next\eql@parseopt@puncterm
  \fi
  \ifx\eql@parseopt@token/%
    \let\eql@parseopt@next\eql@parseopt@punctclear
  \fi
  \ifx\eql@parseopt@token=%
```

```
    \let\eql@parseopt@next\eql@parseopt@relsymb
  \fi
  \ifx\eql@parseopt@token;%
    \let\eql@parseopt@next\eql@parseopt@relcont
  \fi
  \ifx\eql@parseopt@token:%
    \let\eql@parseopt@next\eql@parseopt@relstart
  \fi
  \ifx\eql@parseopt@token|%
    \let\eql@parseopt@next\eql@parseopt@relord
  \fi
  \ifx\eql@parseopt@token?%
    \let\eql@parseopt@next\eql@break@parse@rel
  \fi
}
\def\eql@break@parse@val[#1]{%
  \def\eql@break@sep{#1}\eql@parseopt@peek}
\def\eql@break@parse@star#1{%
  \let\eql@break@sep\eql@break@shortsep\eql@parseopt@peek}
\def\eql@break@parse@rel#1#2{%
  \def\eql@class@rel@composed{#2}\eql@parseopt@end}
```

`\eql@break@process`

```
\def\eql@break@process{%
  \ifdefined\eql@punct@term\eql@punct@apply@top\fi
  \ifdefined\eql@class@rel@composed
    \eql@class@rel@composed
  \else
    \eql@break@print
    \hspace{\glueexpr\eql@break@sep\relax}%
  \fi
}
```

`\eql@break@join`

```
\def\eql@break@join{\eql@srbgroup
  \eql@ifstar@tight
    {\eql@break@join@opt[\eql@break@col@shortsep]}%
    {\eql@testopt@tight\eql@break@join@opt\eql@break@col@sep}}
\def\eql@break@join@opt[#1]#2{\eql@sregroup%
  \hspace{\glueexpr#1\relax}#2\hspace{\glueexpr#1\relax}}
```

`\eqnsep`
`\eqnbreak`
`\eqnjoin`

TODO: expand to lines and columns mode

```
\newcommand{\eqnsep}{\eql@break@col}
\newcommand{\eqnbreak}{\eql@break@line}
\newcommand{\eqnjoin}{\eql@break@join}
```

4.5 Alternative Content Description

TODO: describe **TODO:** would be nice to provide as environments as well **TODO:** implement for PDF tagging

```
\DeclareRobustCommand{\eqnalt}[2][]{}
```

5 Equation Numbering

TODO: describe

5.1 Supporting Definitions

Parameters.

```
\let\eql@tags@autolabel\eql@false
\let\eql@tags@autotag\eql@true
\let\eql@tags@warn\eql@true
```

```
\def\eql@tags@name@generic{[equation]}
```

```
\let\eql@tagpos@doconvert\eql@false
```

```
\def\eql@tagpos@snap{4pt}
```

Registers.

```
\let\eql@numbering@mode\@undefined
```

```
\let\eql@numbering@active\eql@false
\let\eql@numbering@multi\eql@true
```

```
\let\eql@tags@container\@undefined
\def\eql@tags@container@clear{%
  \let\eql@tags@label\@undefined
  \let\eql@tags@name\@undefined
  \let\eql@tags@tag\@undefined
  \let\eql@tags@ref\@undefined
  \let\eql@tags@anchor\@empty
  \eql@tagpos@shift@\z@
  \eql@tagpos@smashup@\z@
  \eql@tagpos@smashdown@\z@
  \let\eql@tagpos@reserve\eql@true
}
```

```
\let\eql@tags@label\@undefined
\let\eql@tags@name\@undefined
\let\eql@tags@tag\@undefined
\let\eql@tags@ref\@undefined
\let\eql@tags@frame@cmd\@firstofone
```

`ags@glabel@` (*counter*)

```
\newcount\eql@tags@glabel@
\eql@tags@glabel@\z@
\def\eql@tags@glabel{equation.eql-\the\eql@tags@glabel@}
\def\eql@tags@glabel@step{\global\advance\eql@tags@glabel@\@ne}
```

```
\let\eql@tagpos@continuous\eql@false
```

```
\newcount\eql@tagpos@row@
\newcount\eql@tagpos@prevrow@
\newdimen\eql@tagpos@shift@
\newdimen\eql@tagpos@smashup@
\newdimen\eql@tagpos@smashdown@
```

```
\newdimen\eql@tagpos@current@
\newdimen\eql@tagpos@plain@
\newdimen\eql@tagpos@raised@
\newdimen\eql@tagpos@target@
\newdimen\eql@tagpos@headroom@
\newdimen\eql@tagpos@footroom@
```

5.2 Schemes

TODO: describe

Table.

```
\def\eql@numbering@tab@sub{sub}
\def\eql@numbering@tab@all{all}
\def\eql@numbering@tab@first{first}
\def\eql@numbering@tab@last{last}
\def\eql@numbering@tab@in{in}
\def\eql@numbering@tab@out{out}
\def\eql@numbering@tab@middle{middle}
\def\eql@numbering@tab@best{best}
\def\eql@numbering@tab@here{here}
\def\eql@numbering@tab@top{top}
\def\eql@numbering@tab@bottom{bottom}
\def\eql@numbering@tab@center{center}
\def\eql@numbering@tab@centerone{centerone}
\def\eql@numbering@tab@median{median}
\def\eql@numbering@tab@baseline{baseline}
```

```
\let\eql@numbering@mode\eql@numbering@tab@all
\let\eql@numbering@mode@multi\eql@numbering@tab@all
\let\eql@numbering@mode@single\eql@numbering@tab@out
```

TODO: describe

```
\let\eql@numbering@tab@subeq\eql@numbering@tab@sub
\let\eql@numbering@tab@subequation\eql@numbering@tab@sub
\let\eql@numbering@tab@subequations\eql@numbering@tab@sub
\let\eql@numbering@tab@mid\eql@numbering@tab@middle
\let\eql@numbering@tab@outside\eql@numbering@tab@out
\let\eql@numbering@tab@inside\eql@numbering@tab@in
\let\eql@numbering@tab@within\eql@numbering@tab@in
\let\eql@numbering@tab@opt\eql@numbering@tab@best
\let\eql@numbering@tab@optimal\eql@numbering@tab@best
\let\eql@numbering@tab@pick\eql@numbering@tab@here
\let\eql@numbering@tab@med\eql@numbering@tab@median
\eql@letcs{eql@numbering@tab@center*}\eql@numbering@tab@baseline
\eql@letcs{eql@numbering@tab@center!}\eql@numbering@tab@centerone
```

TODO: describe

```
\let\eql@numbering@tab@a\eql@numbering@tab@all
\let\eql@numbering@tab@s\eql@numbering@tab@sub
\let\eql@numbering@tab@f\eql@numbering@tab@first
\let\eql@numbering@tab@l\eql@numbering@tab@last
\let\eql@numbering@tab@o\eql@numbering@tab@out
\let\eql@numbering@tab@i\eql@numbering@tab@in
\let\eql@numbering@tab@h\eql@numbering@tab@here
```

```
\let\eql@numbering@tab@t\eql@numbering@tab@top
\let\eql@numbering@tab@b\eql@numbering@tab@bottom
\let\eql@numbering@tab@c\eql@numbering@tab@center
\let\eql@numbering@tab@m\eql@numbering@tab@median
\eql@letcs{eql@numbering@tab@+}\eql@numbering@tab@best
\eql@letcs{eql@numbering@tab@m*}\eql@numbering@tab@middle
\eql@letcs{eql@numbering@tab@c*}\eql@numbering@tab@baseline
\eql@letcs{eql@numbering@tab@c!}\eql@numbering@tab@centerone
```

Implementations. **TODO:** describe

```
\def\eql@numbering@init@all{\let\eql@numbering@multi\eql@true}
```

TODO: describe

```
\def\eql@numbering@init@sub{%
  \let\eql@numbering@multi\eql@true
  \ifdefined\eql@subequations@active
    \let\eql@numbering@mode\eql@numbering@tab@all
  \else
    \let\eql@numbering@subeq@use\eql@true
  \fi
}
```

```
\def\eql@numbering@init@first{\eql@tagpos@row@\@ne}
\def\eql@numbering@init@last{\eql@tagpos@row@\@MM}
\def\eql@numbering@init@here{\eql@tagpos@row@\m@ne}
```

TODO: describe

```
\def\eql@numbering@init@in{%
  \ifdefined\eql@tagsleft
    \eql@numbering@init@last
  \else
    \eql@numbering@init@first
  \fi
}
```

TODO: describe

```
\def\eql@numbering@init@out{%
  \ifdefined\eql@tagsleft
    \eql@numbering@init@first
  \else
    \eql@numbering@init@last
  \fi
}
```

TODO: describe

```
\def\eql@tagpos@eval@middle{%
  \ifnum\eql@tagpos@row@=\z@
    \eql@tagpos@row@\numexpr(\eql@totalrows@
        +\ifdefined\eql@tagsleft\z@\else\@ne\fi)/\tw@\relax
  \fi
}
```

TODO: describe

```
\def\eql@tagpos@eval@best{%
  \ifnum\eql@tagpos@row@=\z@
```

```
    \let\eql@numbering@best@use\eql@true
    \eql@numbering@init@out
  \fi
}
```

TODO: describe

```
\def\eql@numbering@init@continuous{\let\eql@tagpos@continuous\eql@true}
```

TODO: describe

```
\let\eql@numbering@init@top\eql@numbering@init@continuous
\def\eql@tagpos@eval@top{%
  \eql@tagpos@current@\z@
}
```

TODO: describe

```
\let\eql@numbering@init@bottom\eql@numbering@init@continuous
\def\eql@tagpos@eval@bottom{%
  \eql@tagpos@current@\dimexpr\eql@totalheight@
      -\eql@tagheight@block@-\eql@tagdepth@block@\relax
}
```

TODO: describe

```
\let\eql@numbering@init@center\eql@numbering@init@continuous
\def\eql@tagpos@eval@center{%
  \ifnum\eql@totalrows@=\@ne
    \eql@tagpos@row@\@ne
  \fi
  \eql@tagpos@current@\dimexpr(\eql@totalheight@
      -\eql@tagheight@block@-\eql@tagdepth@block@)/\tw@\relax
}
```

TODO: describe

```
\let\eql@numbering@init@centerone\eql@numbering@init@continuous
\def\eql@tagpos@eval@centerone{%
  \eql@tagpos@current@\dimexpr(\eql@totalheight@
      -\eql@tagheight@block@-\eql@tagdepth@block@)/\tw@\relax
}
```

TODO: describe

```
\let\eql@numbering@init@baseline\eql@numbering@init@continuous
\def\eql@tagpos@eval@baseline{%
  \eql@tagpos@current@\dimexpr(\eql@totalheight@
      +\eql@topheight@-\eql@bottomdepth@)/\tw@-\eql@tagheight@block@\relax
}
```

TODO: describe

```
\let\eql@numbering@init@median\eql@numbering@init@continuous
\def\eql@tagpos@eval@median{%
  \ifnum\eql@tagpos@row@=\z@
    \ifodd\eql@totalrows@
      \eql@tagpos@row@\numexpr(\eql@totalrows@+\@ne)/\tw@\relax
    \else
      \eql@tagpos@row@\numexpr(\eql@totalrows@+\tw@)/\tw@\relax
      \eql@dimensions@get\eql@tagpos@row@
      \advance\eql@tagpos@shift@\dimexpr\eql@line@height@
          +(\eql@line@interline@-\eql@tagheight@block@
          +\eql@tagdepth@block@)/\tw@\relax
    \fi
    \ifnum\eql@totalrows@=\@ne
      \eql@tagpos@row@\@ne
    \else
      \eql@tagpos@adjust@eval@convert
      \eql@tagpos@row@\z@
    \fi
  \fi
}
```

Selection.

```
\def\eql@numbering@set#1{%
  \ifcsname eql@numbering@tab@#1\endcsname
    \expandafter\let\expandafter\eql@numbering@mode
      \csname eql@numbering@tab@#1\endcsname
    \ifx\eql@numbering@mode\eql@numbering@tab@all
      \let\eql@numbering@mode@multi\eql@numbering@mode
    \else\ifx\eql@numbering@mode\eql@numbering@tab@sub
      \let\eql@numbering@mode@multi\eql@numbering@mode
    \else
      \let\eql@numbering@mode@single\eql@numbering@mode
    \fi\fi
  \else
    \eql@error{numbering mode '#1' unknown: setting mode to 'all'}%
    \let\eql@numbering@mode\eql@numbering@tab@all
  \fi
}
```

TODO: describe

```
\def\eql@numbering@init{%
  \let\eql@numbering@multi\eql@false
  \let\eql@tagpos@continuous\eql@false
  \let\eql@numbering@subeq@use\eql@false
  \let\eql@numbering@best@use\eql@false
  \eql@tagpos@row@\z@
  \csname eql@numbering@init@\eql@numbering@mode\endcsname
  \ifdefined\eql@numbering@active
    \let\eql@numbering@eqnswinit\@eqnswtrue
  \else
    \let\eql@numbering@eqnswinit\@eqnswfalse
  \fi
  \let\eql@numbering@active\eql@false
}
```

5.3 Interface

Activation. **TODO:** note `\nonumber` already defined, modifications by amsmath

```
\eql@amsmath@after{%
  \let\eql@print@eqnum@default\print@eqnum
  \let\eql@incr@eqnum@default\incr@eqnum
}
```

TODO: describe

```
\protected\def\donumber{%
  \if@eqnsw\else
    \global\@eqnswtrue
    \ifx\print@eqn\@empty
      \global\let\print@eqn\eql@print@eqnum@default
    \fi
    \ifx\incr@eqn\@empty
      \global\let\incr@eqn\eql@incr@eqnum@default
    \fi
  \fi
}
```

TODO: reconsider operation

`\numberhere`

```
\protected\def\eql@numberhere{%
  \ifdefined\eql@numbering@multi
    \global\@eqnswtrue
  \else
    \global\eql@tagpos@row@\eql@row@
  \fi
}
```

TODO: describe

`\numbernext`

```
\protected\def\eql@numbernext{%
  \ifdefined\eql@numbering@multi
    \global\@eqnswfalse
  \else
    \ifdefined\eql@tagpos@continuous\else
      \ifnum\eql@tagpos@row@=\eql@row@
        \global\advance\eql@tagpos@row@\@ne
      \fi
    \fi
  \fi
}
```

Activation Trigger.

```
\def\eql@tags@autoenable{%
  \global\@eqnswtrue
  \ifnum\eql@tagpos@row@=\m@ne
    \numberhere
  \fi
}
```

Labels. **TODO:** describe

`\eql@label@org`

```
\let\eql@label@org\label
```

TODO: describe

```
\def\eql@label@gobble{\eql@ampprotect\eql@testopt@tight\eql@gobbleoptone{}}
```

TODO: describe

```
\protected\def\eql@label{%
  \eql@ampprotect\eql@testopt@tight\eql@tags@add@labelname\eql@testopt@default
}
```

TODO: describe

```
\def\eql@tags@add@labelname[#1]#2{%
  \def\eql@tmp{#1}%
  \ifx\eql@tmp\eql@testopt@default\else
    \eql@tags@add@name{#1}%
  \fi
  \eql@tags@add@label{#2}%
}
```

TODO: describe

```
\def\eql@tags@set@label#1{%
  \ifdefined\eql@tags@warn
    \ifdefined\eql@tags@label
      \eql@warn@label@multiple{#1}%
    \fi
  \fi
  \def\eql@tags@label{#1}%
}
```

TODO: describe

```
\def\eql@tags@set@name#1{%
  \ifdefined\eql@tags@warn
    \ifdefined\eql@tags@name
      \eql@warn@name@multiple
    \fi
  \fi
  \def\eql@tags@name{#1}%
}
```

TODO: describe

```
\def\eql@tags@add@label#1{%
  \ifdefined\eql@tags@autolabel
    \eql@tags@autoenable
  \fi
  \global\eql@appendexpand\eql@tags@container{%
    \noexpand\eql@tags@set@label{#1}}%
}
```

TODO: describe

```
\def\eql@tags@add@name#1{%
  \protected@edef\eql@tmp{\noexpand\eql@tags@set@name{#1}}%
  \global\eql@appendmacro\eql@tags@container\eql@tmp
}
```

TODO: describe

```
\def\eql@tags@addblock@label#1{%
  \eql@appendexpand\eql@tags@container@block{%
    \noexpand\eql@tags@set@label{#1}}%
}
```

TODO: describe

```
\def\eql@tags@addblock@name#1{%
  \protected@edef\eql@tmp{\noexpand\eql@tags@set@name{#1}}%
  \eql@appendmacro\eql@tags@container@block\eql@tmp
}
```

Tags. **TODO:** describe

`\eql@tag@default`

```
\protected\def\eql@tag@default{%
  \eql@warn@here\tag
  \eql@tag@gobble
}
\let\tag\eql@tag@default
```

`\eql@tag@gobble`

```
\def\eql@tag@gobble{%
  \eql@ampprotecttwo\eql@teststaropt@tight\eql@gobbleoptone\eql@gobbleoptone{}}
```

TODO: describe

```
\protected\def\eql@tag{%
  \eql@ampprotecttwo\eql@teststaropt@tight
    {\eql@tags@add@tagform@off\eql@tags@add@tagref}{\eql@tags@add@tagref}%
    \eql@testopt@default
}
```

`\eql@tags@add@tagref`

```
\def\eql@tags@add@tagref[#1]#2{%
  \def\eql@tmp{#1}%
  \ifx\eql@tmp\eql@testopt@default\else
    \eql@tags@add@ref{#1}%
  \fi
  \eql@tags@add@tag{#2}%
}
```

TODO: describe

```
\def\eql@tags@set@tag#1{%
  \ifdefined\eql@tags@warn
    \ifdefined\eql@tags@tag
      \eql@warn@tag@multiple
    \fi
  \fi
  \def\eql@tags@tag{#1}%
}
```

TODO: describe

```
\def\eql@tags@set@ref#1{%
  \ifdefined\eql@tags@warn
    \ifdefined\eql@tags@ref
      \eql@warn@ref@multiple
    \fi
  \fi
  \def\eql@tags@ref{#1}%
}
```

TODO: describe

```
\def\eql@tags@add@tag#1{%
  \ifdefined\eql@tags@autotag
    \eql@tags@autoenable
  \fi
  \protected@edef\eql@tmp{\noexpand\eql@tags@set@tag{#1}}%
  \global\eql@appendmacro\eql@tags@container\eql@tmp
}
```

TODO: describe

```
\def\eql@tags@add@ref#1{%
  \protected@edef\eql@tmp{\noexpand\eql@tags@set@ref{#1}}%
  \global\eql@appendmacro\eql@tags@container\eql@tmp
}
```

`tags@add@tagform@off`

```
\def\eql@tags@add@tagform@off{%
  \global\eql@append\eql@tags@container{\let\eql@tags@tagform\@firstofone}%
}
```

TODO: describe

```
\def\eql@tags@addblock@tag#1{%
  \protected@edef\eql@tmp{\noexpand\eql@tags@set@tag{#1}}%
  \eql@appendmacro\eql@tags@container@block\eql@tmp
}
```

TODO: describe

```
\def\eql@tags@addblock@ref#1{%
  \protected@edef\eql@tmp{\noexpand\eql@tags@set@ref{#1}}%
  \eql@appendmacro\eql@tags@container@block\eql@tmp
}
```

TODO: describe

```
\def\eql@tags@addblock@tagform@off{%
  \eql@append\eql@tags@container@block{\let\eql@tags@tagform\@firstofone}%
}
```

Raise Tags.

`\raisetag`

```
\def\eql@raisetag@default{%
  \eql@warn@here\raisetag
  \eql@raisetag@gobble
}
```

```
\def\eql@raisetag@gobble{%
  \eql@ampprotecttwo\eql@ifstar@tight\@gobble\@gobble
}
```

TODO: describe

```
\eql@amsmath@let\raisetag\eql@raisetag@default
```

```
\def\eql@raisetag{%
  \eql@ampprotecttwo\eql@ifstar@tight\eql@tags@add@raiseshift\eql@raisetag@test
}
```

```
\def\eql@raisetag@test#1{%
  \def\eql@tmpa{#1}\def\eql@tmpb{!}%
  \ifx\eql@tmpa\eql@tmpb
    \eql@tags@add@forceraise
  \else
    \eql@tags@add@raisesmash{#1}%
  \fi
}
```

```
\def\eql@tags@add@raiseshift#1{%
  \global\eql@appendexpand\eql@tags@container{%
    \advance\eql@tagpos@shift@\the\glueexpr#1\relax\relax}%
}
```

```
\def\eql@tags@add@raisesmash#1{%
  \dimen@\glueexpr#1\relax
  \ifdim\dimen@<\z@
    \global\eql@appendexpand\eql@tags@container{%
      \advance\eql@tagpos@smashdown@-\the\dimen@\relax}%
  \else
    \global\eql@appendexpand\eql@tags@container{%
      \advance\eql@tagpos@smashup@\the\dimen@\relax}%
  \fi
}
```

```
\def\eql@tags@add@forceraise{%
  \global\eql@append\eql@tags@container{\let\eql@tagpos@reserve\eql@false}%
}
```

5.4 Integration

TODO: describe

Support. **TODO:** describe

```
\def\eql@numbering@settools{%
  \let\label\eql@label
  \let\tag\eql@tag
  \let\raisetag\eql@raisetag
  \let\numberhere\eql@numberhere
  \let\numbernext\eql@numbernext
}
```

TODO: not necessary anymore

```
\def\eql@numbering@settools@gobble{%
  \let\label\eql@label@gobble
  \let\tag\eql@tag@gobble
  \let\raisetag\eql@raisetag@gobble
  \let\numberhere\relax
  \let\numbernext\relax
}
```

```
\def\eql@numbering@autoblock{%
  \begingroup
    \let\eql@tags@warn\eql@false
    \eql@tags@container@block
    \ifdefined\eql@tags@autolabel
```

```
      \ifdefined\eql@tags@label
        \global\@eqnswtrue
      \fi
    \fi
    \ifdefined\eql@tags@autotag
      \ifdefined\eql@tags@tag
        \global\@eqnswtrue
      \fi
    \fi
  \endgroup
}
```

```
\def\eql@numbering@warnunused{%
  \ifdefined\eql@tags@label
    \eql@warn@label@unused
  \fi
  \ifdefined\eql@tags@name
    \eql@warn@name@unused
  \fi
  \ifdefined\eql@tags@tag
    \eql@warn@tag@unused
  \fi
  \ifdefined\eql@tags@erf
    \eql@warn@ref@unused
  \fi
}
```

Single Line. **TODO:** describe

```
\def\eql@numbering@single@init{%
  \let\eql@numbering@multi\eql@false
  \eql@numbering@settools
  \eql@numbering@eqnswinit
  \eql@numbering@autoblock
  \global\let\eql@tags@container\eql@tags@container@block
  \let\eql@tags@warn\eql@true
}
```

```
\def\eql@numbering@single@eval{%
  \ifnum\eql@tagpos@row@=\m@ne
    \@eqnswfalse
  \fi
}
```

Multi-Line Measuring Pass. **TODO:** describe

```
\def\eql@numbering@measure@init{%
  \eql@numbering@settools
  \ifdefined\eql@numbering@multi\else
    \eql@numbering@eqnswinit
    \eql@numbering@autoblock
  \fi
  \global\let\eql@tags@container\eql@tags@container@block
  \let\eql@tags@warn\eql@true
}
```

TODO: might select only relevant routines in init **TODO:** describe

```
\def\eql@numbering@measure@line@begin{%
  \ifdefined\eql@numbering@multi
    \global\eql@numbering@eqnswinit
  \fi
}
```

TODO: describe

```
\def\eql@numbering@measure@blocktag{%
  \ifdefined\eql@numbering@multi
    \@eqnswfalse
  \else
    \ifnum\eql@tagpos@row@=\m@ne
      \@eqnswfalse
    \fi
    \ifnum\eql@totalrows@=\z@
      \@eqnswfalse
    \fi
  \fi
}
```

Multi-Line Print Pass. **TODO:** describe

TODO: can we be absolutely sure about all values being preserved global might pick up a value from a higher level block and restore it globally!

```
\def\eql@numbering@print@init{%
  \let\eql@tags@warn\eql@false
  \ifdefined\eql@numbering@multi
    \eql@numbering@settools
    \global\let\eql@tags@container\eql@tags@container@block
  \else
    \let\eql@tags@container@block\eql@tags@container
    \eql@numbering@settools@gobble
  \fi
}
```

TODO: might select only relevant routines in init **TODO:** describe

```
\def\eql@numbering@print@block@begin{%
  \ifdefined\eql@numbering@multi\else
    \ifnum\eql@tagpos@row@>\z@
      \eql@tags@makeblockanchor
      \global\eql@appendexpand\eql@tags@container@block{%
        \def\noexpand\eql@tags@anchor{%
          \unexpanded\expandafter{\eql@tags@anchor}}}%
    \fi
  \fi
  \ifdefined\eql@numbering@subeq@use
    \eql@tags@printsubeqlabel
  \fi
}
```

TODO: describe

```
\def\eql@numbering@print@line@begin{%
  \ifdefined\eql@numbering@multi
    \global\eql@numbering@eqnswinit
  \fi
}
```

TODO: describe

```
\def\eql@numbering@print@line@eval{%
  \ifdefined\eql@numbering@multi
    \if@eqnsw
      \eql@tags@container
    \fi
  \else
    \ifnum\eql@tagpos@row@=\eql@row@
      \@eqnswtrue
      \eql@tags@container@block
    \else
      \@eqnswfalse
    \fi
  \fi
}
```

5.5 Positioning

TODO: describe

```
\def\eql@tagpos@single@eval{%
  \if@eqnsw
    \csname eql@tagpos@eval@\eql@numbering@mode\endcsname
    \ifnum\eql@tagpos@row@>\@ne
      \eql@tagpos@row@\@ne
    \fi
    \ifdefined\eql@tagpos@doconvert
      \let\eql@tagpos@continuous\eql@true
    \fi
    \ifdefined\eql@tagpos@continuous
      \eql@tagpos@single@eval@continuous
    \fi
  \else
    \eql@tagpos@row@\z@
  \fi
  \eql@tagpos@prevrow@\z@
  \eql@tagpos@headroom@\z@
  \eql@tagpos@footroom@\z@
}
```

TODO: describe

```
\def\eql@tagpos@single@eval@continuous{%
  \ifnum\eql@tagpos@row@>\z@
    \eql@tagpos@target@\eql@tagpos@shift@
  \else
    \eql@tagpos@target@\dimexpr\eql@line@height@
        -\eql@tagpos@current@+\eql@tagpos@shift@-\eql@tagheight@block@\relax
  \fi
  \eql@tagpos@row@\@ne
  \ifdim\ifdim\eql@tagpos@target@<\z@-\fi
      \eql@tagpos@target@<\glueexpr\eql@tagpos@snap\relax
    \eql@tagpos@target@\z@
  \fi
}
```

TODO: describe

```
\def\eql@tagpos@adjust@eval{%
  \if@eqnsw
    \csname eql@tagpos@eval@\eql@numbering@mode\endcsname
    \ifnum\eql@tagpos@row@>\eql@totalrows@
      \eql@tagpos@row@\eql@totalrows@
    \fi
    \ifdefined\eql@tagpos@doconvert
      \let\eql@tagpos@continuous\eql@true
    \fi
    \ifdefined\eql@tagpos@continuous
      \ifnum\eql@tagpos@row@>\z@
        \eql@tagpos@adjust@eval@convert
      \fi
      \eql@tagpos@adjust@eval@continuous
    \fi
  \else
    \eql@tagpos@row@\z@
    \eql@tagpos@prevrow@\z@
  \fi
}
```

TODO: describe

```
\def\eql@tagpos@adjust@eval@convert{%
  \eql@tagpos@current@\z@
  \eql@dimensions@for{%
    \ifnum\eql@row@<\eql@tagpos@row@
      \advance\eql@tagpos@current@\dimexpr\eql@line@interline@
          +\eql@line@height@+\eql@line@depth@\relax
    \fi
    \ifnum\eql@row@=\eql@tagpos@row@
      \advance\eql@tagpos@current@\dimexpr\eql@line@interline@
          +\eql@line@height@-\eql@tagheight@block@\relax
    \fi
  }%
}
```

TODO: describe

```
\def\eql@tagpos@adjust@eval@continuous{%
  \dimen@\dimexpr\eql@tagpos@current@-\eql@tagpos@shift@\relax
  \eql@tagpos@row@\eql@totalrows@
  \eql@tagpos@prevrow@\z@
  \eql@tagpos@headroom@\z@
  \eql@tagpos@footroom@\z@
  \eql@dimensions@for{%
    \ifnum\eql@tagpos@row@=\eql@totalrows@
      \eql@tagpos@headroom@\eql@line@interline@
      \eql@tagpos@target@\dimexpr\eql@line@interline@
          +\eql@line@height@-\dimen@-\eql@tagheight@block@\relax
      \ifdim\ifdim\eql@tagpos@target@<\z@-\fi
          \eql@tagpos@target@<\glueexpr\eql@tagpos@snap\relax
        \advance\dimen@\eql@tagpos@target@
        \eql@tagpos@target@\z@
      \fi
      \ifdim\eql@tagpos@target@>%
          \ifdefined\eql@tagsleft-1sp\relax\else\z@\fi
        \eql@tagpos@row@\eql@row@
        \eql@tagpos@prevrow@\numexpr\eql@row@-\@ne\relax
      \fi
      \advance\dimen@-\dimexpr\eql@line@interline@
          +\eql@line@depth@+\eql@line@height@\relax
    \fi
    \ifnum\eql@row@=\numexpr\eql@tagpos@row@+\@ne\relax
      \eql@tagpos@footroom@\eql@line@interline@
    \fi
  }%
}
```

TODO: describe

```
\def\eql@tagpos@print@line@eval{%
  \ifdefined\eql@tagpos@continuous
    \eql@tagpos@print@line@eval@continuous
  \else
    \eql@tagpos@print@line@eval@discrete
  \fi
}
```

TODO: describe

```
\def\eql@tagpos@print@line@eval@continuous{%
  \if@eqnsw
    \ht\eql@tagbox@\dimexpr\ht\eql@tagbox@-\eql@tagpos@smashup@\relax
    \dp\eql@tagbox@\dimexpr\dp\eql@tagbox@-\eql@tagpos@smashdown@\relax
    \eql@tagpos@plain@\eql@tagpos@target@
    \@tempdima\dimexpr\eql@line@height@+\eql@tagpos@headroom@
        -\ht\eql@tagbox@\relax
    \@tempdimb\dimexpr-\eql@line@depth@-\eql@tagpos@footroom@
        +\dp\eql@tagbox@\relax
    \ifnum\eql@row@=\@ne
      \@tempdima.5\maxdimen
    \fi
    \ifnum\eql@row@=\eql@totalrows@
      \@tempdimb-.5\maxdimen
    \fi
    \ifdim\eql@tagpos@plain@>\@tempdima
      \ifdim\eql@tagpos@plain@>\@tempdimb
        \ifdim\@tempdima>\@tempdimb
          \eql@tagpos@plain@\@tempdima
        \else
          \eql@tagpos@plain@\@tempdimb
        \fi
      \fi
    \else
      \ifdim\eql@tagpos@plain@<\@tempdimb
        \ifdim\@tempdima>\@tempdimb
          \eql@tagpos@plain@\@tempdimb
        \else
          \eql@tagpos@plain@\@tempdima
        \fi
      \fi
    \fi
    \ifnum\eql@tagpos@prevrow@>\z@
      \eql@tagpos@raised@\dimexpr\eql@line@height@+\dp\eql@tagbox@\relax
      \ifdim\eql@tagpos@raised@>\eql@tagpos@plain@\else
        \eql@tagpos@raised@\eql@tagpos@plain@
        \let\eql@tagpos@reserve\eql@false
      \fi
    \else
```

```
      \ifdim\eql@tagpos@target@>%
          \ifdefined\eql@tagsleft-1sp\relax\else\z@\fi
        \eql@tagpos@raised@\dimexpr\eql@line@height@+\dp\eql@tagbox@\relax
        \ifdim\eql@tagpos@raised@>\eql@tagpos@plain@\else
          \eql@tagpos@raised@\eql@tagpos@plain@
          \let\eql@tagpos@reserve\eql@false
        \fi
      \else
        \eql@tagpos@raised@\dimexpr-\eql@line@depth@
            -\ht\eql@tagbox@\relax
        \ifdim\eql@tagpos@raised@<\eql@tagpos@plain@\else
          \eql@tagpos@raised@\eql@tagpos@plain@
          \let\eql@tagpos@reserve\eql@false
        \fi
      \fi
    \fi
  \else
    \ifnum\eql@tagpos@prevrow@=\eql@row@
      \eql@tagwidth@\eql@tagwidth@block@
    \else
      \let\eql@tagpos@reserve\eql@false
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@tagpos@print@line@eval@discrete{%
  \if@eqnsw
    \ht\eql@tagbox@\dimexpr\ht\eql@tagbox@-\eql@tagpos@smashup@\relax
    \dp\eql@tagbox@\dimexpr\dp\eql@tagbox@-\eql@tagpos@smashdown@\relax
    \eql@tagpos@plain@\eql@tagpos@shift@
    \eql@tagpos@headroom@\z@
    \eql@tagpos@footroom@\z@
    \ifdim\eql@tagpos@shift@>%
          \ifdefined\eql@tagsleft-1sp\relax\else\z@\fi
      \eql@tagpos@raised@\dimexpr\eql@line@height@+\dp\eql@tagbox@\relax
    \else
      \eql@tagpos@raised@\dimexpr-\eql@line@depth@-\ht\eql@tagbox@\relax
    \fi
  \else
    \let\eql@tagpos@reserve\eql@false
  \fi
}
```

TODO: describe

```
\def\eql@tagpos@print@line@end{%
  \ifdefined\eql@tagpos@continuous
    \ifnum\eql@tagpos@prevrow@=\eql@row@
      \ifdefined\eql@tagpos@reserve
        \global\eql@appendexpand\eql@tags@container@block{%
          \advance\eql@tagpos@headroom@\the\dimexpr\eql@line@height@
              +\eql@line@depth@\relax\relax}%
        \eql@displaybreak@star\@M
      \fi
    \fi
  \fi
}
```

5.6 Component Display

Showkeys Integration. **TODO:** describe

```
\let\eql@SK@loaded\eql@false
\let\eql@SK@label\@gobble
\let\eql@SK@clearlabel\@empty
\let\eql@SK@setlabel\@gobble
\let\eql@SK@printlabel@right\@empty
\let\eql@SK@printlabel@left\@empty
\let\eql@SK@printlabel@line\@empty
\def\eql@label@clean{\eql@label@org}
\AddToHook{package/showkeys/after}{%
  \let\eql@SK@loaded\eql@true
  \def\eql@SK@label#1{\SK@\SK@@label#1}%
  \def\eql@SK@clearlabel{\let\eql@SK@lab\relax}%
  \eql@SK@clearlabel
  \def\eql@SK@@label#1>#2\SK@{%
    \def\eql@SK@lab{\smash{\SK@labelcolor\showkeyslabelformat{#2}}}%
  }%
  \def\eql@SK@setlabel#1{\SK@\eql@SK@@label#1}%
  \def\eql@SK@printlabel@right{%
    \ifx\eql@SK@lab\relax\else
      \rlap{\kern\marginparsep\eql@SK@lab}%
      \eql@SK@clearlabel
    \fi
  }%
  \def\eql@SK@printlabel@left{%
    \ifx\eql@SK@lab\relax\else
      \llap{\eql@SK@lab\kern\marginparsep}%
      \eql@SK@clearlabel
    \fi
  }%
  \def\eql@SK@printlabel@line{%
    \ifx\eql@SK@lab\relax\else
      \dimen@\prevdepth
      \nointerlineskip
      \ifdefined\eql@tagsleft
        \llap{%
          \eql@SK@lab
          \kern\marginparsep
        }%
        \eql@SK@clearlabel
      \else
        \rlap{%
          \dimen@\displaywidth
          \advance\dimen@\marginparsep
          \kern\dimen@
          \eql@SK@lab
        }%
      \fi
      \eql@SK@clearlabel
      \prevdepth\dimen@
    \fi
  }%
  \let\eql@label@org\label
  \def\eql@label@clean{\let\SK@\@gobbletwo\eql@label@org}
}
```

Labels.

`eql@composetag@label` **TODO:** describe

```
\def\eql@composetag@label{%
  \eql@SK@clearlabel
  \ifdefined\eql@tags@label
    \eql@SK@setlabel\eql@tags@label
    \ifdefined\eql@tags@name
      \let\@currentlabelname\eql@tags@name
    \else
      \let\@currentlabelname\eql@tags@name@generic
    \fi
    \expandafter\eql@label@clean\expandafter{\eql@tags@label}%
  \fi
}
```

TODO: describe

```
\def\eql@tags@printsubeqlabel{%
  \eql@tags@container@parent
  \ifdefined\eql@tags@label
    \eql@tags@makeblockanchor
    \eql@SK@setlabel\eql@tags@label
    \begingroup
      \def\@currentcounter{equation}%
      \eql@tags@anchor
      \let\@currentlabelname\eql@tags@name@generic
      \protected@edef\@currentlabel{\p@equation\theparentequation}%
      \expandafter\eql@label@clean\expandafter{\eql@tags@label}%
    \endgroup
    \eql@SK@printlabel@line
  \fi
}
```

Hyperref Anchors. **TODO:** describe

```
\let\eql@Hy@anchor\@gobble
\AddToHook{package/hyperref/after}{%
  \def\eql@Hy@anchor#1{%
    \Hy@raisedlink{\hyper@anchor{#1}}%
  }%
}
```

TODO: describe

```
\def\eql@tags@makeblockanchor{%
  \eql@tags@glabel@step
  \eql@Hy@anchor\eql@tags@glabel
  \edef\eql@tags@anchor{%
    \def\noexpand\thepage{\thepage}%
    \def\noexpand\@currentHref{\eql@tags@glabel}%
  }%
}
```

TODO: describe

`ql@composetag@anchor`

```
\def\eql@composetag@anchor{%
```

```
  \ifdefined\eql@tags@tag
    \def\@currentcounter{equation}%
    \ifdefined\eql@tags@ref
      \let\@currentlabel\eql@tags@ref
    \else
      \protected@edef\@currentlabel{\p@equation\eql@tags@tag}%
    \fi
    \eql@tags@glabel@step
    \edef\@currentHref{\eql@tags@glabel}%
    \eql@Hy@anchor\@currentHref
  \else
    \refstepcounter{equation}%
    \protected@edef\eql@tags@tag{\theequation}%
  \fi
  \eql@tags@anchor
}
```

Tag Layout. **TODO:** describe

```
\def\eql@tags@taglayout@set@direct#1{%
  \def\eql@tags@taglayout##1{#1}%
}
\def\eql@tags@taglayout@set#1{%
  \def\eql@tags@taglayout##1{\hbox{\m@th\normalfont#1}}%
}
```

TODO: describe

```
\def\eql@tags@tagform@set@direct#1{%
  \def\eql@tags@tagform##1{#1}%
}
\def\eql@tags@tagform@set#1#2#3{%
  \def\eql@tags@tagform##1{#1\ignorespaces#2\unskip\@@italiccorr#3}%
}
```

```
\eql@tags@taglayout@set{#1}
\eql@tags@tagform@set({#1})
\def\eql@tags@tagcompose#1{\eql@tags@taglayout{\eql@tags@tagform{#1}}}
```

```
\protected\def\tagform{\eql@tags@tagform}
\protected\def\tagbox{\eql@tags@taglayout}
\protected\def\tagboxed{\eql@tags@tagcompose}
```

`\eqref` amsmath defines the macro `\eqref` to refer to equation labels in a proper format. We provide it for completeness:

```
\protected\def\eql@eqref#1{\textup{\eql@tags@tagcompose{\ref{#1}}}}
```

`\eql@composetag@tag` **TODO:** describe

```
\def\eql@composetag@tag{%
  \eql@tagging@tagbegin
  \eql@tags@frame@cmd{%
    \eql@tags@taglayout{%
      \eql@tags@tagform\eql@tags@tag
      \eql@tagging@tagsave
    }%
  }%
  \eql@tagging@tagend
}
```

5.7 Tag Composition

TODO: describe

```
\def\eql@composetag@measure{%
  \ifdefined\eql@tags@tag\else
    \stepcounter{equation}%
    \let\eql@tags@tag\theequation
  \fi
  \eql@tags@frame@cmd{\eql@tags@taglayout{\eql@tags@tagform\eql@tags@tag}}%
  \ifdefined\eql@numbering@multi
    \global\let\eql@tags@container\eql@tags@container@clear
  \fi
}
```

TODO: describe

```
\def\eql@composetag@print{%
  \eql@composetag@anchor
  \eql@composetag@label
  \ifdefined\eql@tagsleft
    \eql@SK@printlabel@left
    \eql@composetag@tag
  \else
    \eql@composetag@tag
    \eql@SK@printlabel@right
  \fi
  \global\let\eql@tags@container\eql@tags@container@clear
}
```

TODO: describe

TODO: one might still compare width to zero and pretend the tag is absent??

```
\def\eql@tagbox@make#1{%
  \setbox\eql@tagbox@\hbox{\eql@strut@tag\@lign#1}%
  \eql@tagwidth@\wd\eql@tagbox@
  \ifdim\eql@tagwidth@<\eql@tagwidthmin@
    \eql@tagwidth@\eql@tagwidthmin@
  \fi
  \advance\eql@tagwidth@\eql@tagsepmin@
}
```

TODO: describe

```
\def\eql@tagbox@print@adjustheadroom{%
  \dimen@\dimexpr\ht\eql@tagbox@+\eql@tagpos@current@-\eql@line@height@\relax
  \ifdim\dimen@>\z@
    \ifdim\dimen@>\eql@tagpos@headroom@
      \ht\eql@tagbox@\dimexpr\ht\eql@tagbox@-\eql@tagpos@headroom@\relax
    \else
      \ht\eql@tagbox@\dimexpr\eql@line@height@-\eql@tagpos@current@\relax
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@tagbox@print@adjustfootroom{%
  \dimen@\dimexpr\dp\eql@tagbox@-\eql@tagpos@current@-\eql@line@depth@\relax
  \ifdim\dimen@>\z@
    \ifdim\dimen@>\eql@tagpos@footroom@
```

```
      \dp\eql@tagbox@\dimexpr\dp\eql@tagbox@-\eql@tagpos@footroom@\relax
    \else
      \dp\eql@tagbox@\dimexpr\eql@line@depth@+\eql@tagpos@current@\relax
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@tagbox@print@extendabove{%
  \dimen@\dimexpr\ht\eql@tagbox@+\eql@tagpos@current@-\eql@line@height@\relax
  \ifdim\dimen@>\z@
    \global\eql@appendexpand\eql@display@container{%
      \eql@display@aboveextend@\the\dimen@\relax}%
  \fi
}
```

TODO: describe

```
\def\eql@tagbox@print@extendbelow{%
  \dimen@\dimexpr\dp\eql@tagbox@-\eql@tagpos@current@-\eql@line@depth@\relax
  \ifdim\dimen@>\z@
    \global\eql@appendexpand\eql@display@container{%
      \eql@display@belowextend@\the\dimexpr\dimen@\relax}%
  \fi
}
```

TODO: describe

```
\def\eql@tagbox@print@prepare{%
  \ifdefined\eql@tagpos@reserve
    \eql@tagpos@current@\eql@tagpos@plain@
  \else
    \eql@tagpos@current@\eql@tagpos@raised@
  \fi
  \ifdim\eql@tagpos@headroom@>\z@
    \eql@tagbox@print@adjustheadroom
  \fi
  \ifdim\eql@tagpos@footroom@>\z@
    \eql@tagbox@print@adjustfootroom
  \fi
  \ifnum\eql@row@=\@ne
    \eql@tagbox@print@extendabove
  \fi
  \ifnum\eql@row@=\eql@totalrows@
    \eql@tagbox@print@extendbelow
  \fi
}
```

TODO: describe

```
\def\eql@tagbox@print@tagsright{%
  \eql@tagbox@print@prepare
  \kern-\wd\eql@tagbox@
  \raise\eql@tagpos@current@\box\eql@tagbox@
}
```

TODO: describe

```
\def\eql@tagbox@print@tagsleft{%
  \eql@display@firstavail@set\z@
  \eql@tagbox@print@prepare
  \wd\eql@tagbox@\z@
  \raise\eql@tagpos@current@\box\eql@tagbox@
}
```

`ql@tagbox@print@cell`

```
\def\eql@tagbox@print@cell{%
  \eql@tagging@tagaddbox
  \ifdefined\eql@tagsleft
    \ifdefined\eql@tagpos@reserve
      \ifdim\eql@tagwidth@>\dimexpr\eql@line@avail@+\eql@tagfuzz@\relax
        \let\eql@tagpos@reserve\eql@false
      \fi
    \fi
    \if@eqnsw
      \eql@tagbox@print@tagsleft
    \fi
    \kern\displaywidth
  \else
    \kern\displaywidth
    \ifdefined\eql@tagpos@reserve
      \ifdim\eql@tagwidth@>%
          \dimexpr\displaywidth-\eql@line@width@+\eql@tagfuzz@\relax
        \let\eql@tagpos@reserve\eql@false
      \fi
    \fi
    \if@eqnsw
      \eql@tagbox@print@tagsright
    \fi
  \fi
}
```

6 Subequation Numbering

We replicate the amsmath functionality to number a block of equations with a common number and a sub-numbering.

6.1 Definitions

`entequation` (*counter*) We define a counter to store the main equation number while in subequation mode. It makes sense to share this definition with amsmath as `parentequation`, and we need to undefine it when amsmath is loaded at a later stage:

```
\eql@amsmath@undefine\c@parentequation
\eql@amsmath@undefine\theparentequation
\ifdefined\c@parentequation\else
\newcounter{parentequation}
\fi
```

`ubequations@template` We store a template which will installed as `\theequation` in subequations mode: **TODO:** need to protect something?!

```
\def\eql@subequations@template{\theparentequation\alph{equation}}
\def\eql@subequations@template@set#1{\def\eql@tmp##1##2{#1}%
  \edef\eql@subequations@template{%
    \unexpanded\expandafter{\eql@tmp\theparentequation{equation}}}}
```

`@subequations@active` A boolean register which tells whether subequations are in use and thus must not be invoked again:

```
\let\eql@subequations@active\eql@false
```

`ql@subequations@init` Low-level initialise the subequations mode. Store the equation counter in `\eql@subequations@restorecounter` for the case that no equation numbers will be used. Step the equation counter, copy it to `parentequation` and initialise `\theparentequation` (and its hyperref counterpart) with the expanded current value of `\theequation`; fill with tag data instead if a tag has been specified. Reset the equation counter and use the template for `\theequation`:

```
\def\eql@subequations@init{%
  \edef\eql@subequations@restorecounter{%
    \global\c@equation\the\c@equation\relax}%
  \eql@tags@container@block
  \ifdefined\eql@tags@tag
    \eql@tags@glabel@step
    \protected@edef\theHparentequation{\eql@tags@glabel}%
    \protected@edef\theparentequation{\eql@tags@tag}%
  \else
    \advance\c@equation\@ne
    \protected@edef\theparentequation{\theequation}%
    \ifdefined\theHequation
      \protected@edef\theHparentequation{\theHequation}%
    \fi
  \fi
  \global\c@parentequation\c@equation
  \global\c@equation\z@
  \let\theequation\eql@subequations@template
  \def\theHequation{\theHparentequation.\arabic{equation}}%
}
```

`l@subequations@close` Low-level close the subequations mode. If no number has been used, return to the original equation counter, otherwise use the value stored in `parentequation`. Note that we cannot use `\setcounter` here because the calc version would involve actions which are not allowed after `\halign` within a display equation:

```
\def\eql@subequations@close{%
  \ifnum\c@equation=\z@
    \eql@subequations@restorecounter
  \else
    \global\c@equation\c@parentequation
  \fi
}
```

6.2 Environment

`l@subequations@start` Start the subequations environment with optional parameters in `#1`. Enter subequations mode and set an anchor for subsequent `\label`'s. Manually print the showkeys tag: **TODO:** join with other similar anchor routines `\eql@tags@printsubeqlabel`

```
\def\eql@subequations@start{%
  \let\eql@tags@container@block\eql@tags@container@clear
  \eql@nextopt@process{subequations}%
  \eql@subequations@init
  \eql@tags@glabel@step
  \edef\eql@subequations@currentHref{\eql@tags@glabel}%
  \eql@Hy@anchor\eql@subequations@currentHref
  \edef\eql@subequations@thepage{\thepage}%
  \def\@currentcounter{equation}%
  \let\@currentHref\eql@subequations@currentHref
  \protected@edef\@currentlabel{\p@equation\theparentequation}%
  \eql@tags@container@block
  \ifdefined\eql@tags@name
    \let\@currentlabelname\eql@tags@name
  \else
    \let\@currentlabelname\eql@tags@name@generic
  \fi
  \let\eql@subequations@active\eql@true
  \ifdefined\eql@tags@label
    \eql@SK@label\eql@tags@label
  \fi
  \ignorespaces
}
```

`eql@subequations@end` End the subequations environment. Issue the label if one has been specified and an equation number has actually been used. Then close subequations mode: **TODO:** how about tag* ?! also within equations!

```
\def\eql@subequations@end{%
  \ifnum\c@equation>\z@
    \eql@tags@container@block
    \ifdefined\eql@tags@label
      \begingroup
        \def\@currentcounter{equation}%
        \let\thepage\eql@subequations@thepage
        \let\@currentHref\eql@subequations@currentHref
        \protected@edef\@currentlabel{\p@equation\theparentequation}%
        \ifdefined\eql@tags@name
          \let\@currentlabelname\eql@tags@name
        \else
          \let\@currentlabelname\eql@tags@name@generic
        \fi
        \expandafter\eql@label@clean\expandafter{\eql@tags@label}%
      \endgroup
    \fi
  \fi
  \eql@subequations@close
}
```

`subequations` (*env.*) The subequations environment tests for optional parameters and passes on to the start and end routines:

```
\newenvironment{eql@subequations}{%
  \eql@verbose@info\eql@verbose@msg@enterenv
  \eql@subequations@testall\eql@subequations@start%
}{%
  \eql@subequations@end
  \ignorespacesafterend
  \eql@verbose@info\eql@verbose@msg@leaveenv
}
```

TODO: describe

```
\def\eql@subequations@testall{\eql@parseopt@env\eql@subequations@testall@parse}
```

```
\def\eql@subequations@testall@parse{%
  \ifx\eql@parseopt@token[%]
    \let\eql@parseopt@next\eql@parseopt@opt
  \fi
  \ifx\eql@parseopt@token\eql@atxi
    \let\eql@parseopt@next\eql@parseopt@label
  \fi
  \ifx\eql@parseopt@token\eql@atxii
    \let\eql@parseopt@next\eql@parseopt@label
  \fi
  \ifx\eql@parseopt@token\label
    \let\eql@parseopt@next\eql@parseopt@end
  \fi
}
```

6.3 Subequation Scheme

TODO: describe

```
\def\eql@numbering@subeq@init{%
  \let\eql@save@theequation\theequation
  \let\eql@save@theHequation\theHequation
  \eql@subequations@init
  \let\eql@tags@container@parent\eql@tags@container@block
  \let\eql@tags@container@block\eql@tags@container@clear
}
```

TODO: describe

```
\def\eql@numbering@subeq@test{%
  \ifnum\eql@tagrows@<\tw@
    \let\eql@tags@container@block\eql@tags@container@parent
    \let\eql@numbering@subeq@use\eql@false
    \let\theequation\eql@save@theequation
    \let\theHequation\eql@save@theHequation
    \eql@subequations@restorecounter
  \fi
}
```

TODO: describe

```
% \TODO note must not use setcounter here (when calc is loaded)
\def\eql@numbering@subeq@close{%
  \eql@subequations@close
}
```

7 Display Equations Support

TODO: describe

```
\let\eql@display@injectbefore\@undefined
\let\eql@display@injectafter\@undefined
\let\eql@interline@container\@undefined
\def\eql@interline@container@clear{%
  \eql@displaybreak@pen@\@MM
  \eql@vspaceskip@\z@skip
}
```

7.1 Display Breaks

TODO: describe

`erdisplaylinepenalty`

```
\interdisplaylinepenalty\@M
```

`\eql@getdsp@pen` **TODO:** isn't this the opposite order than `\@getpen`?!

```
\def\eql@getdsp@pen#1{%
  \ifcase #1\@M \or 9999 \or 6999 \or 2999 \or \z@\fi
}
```

TODO: allow a displaybreak before equations

```
\protected\def\eql@displaybreak@default{%
  \eql@warning{Invalid use of \string\displaybreak}{}%
  \eql@teststaroropt@loose\@gobble\eql@gobbleopt{}}
\eql@amsmath@after{\let\eql@displaybreak@default\displaybreak}
\eql@amsmath@let\displaybreak\eql@displaybreak@default

\newcount\eql@displaybreak@pen@
\newcount\eql@displaybreak@prepen@
\newcount\eql@displaybreak@postpen@
```

TODO: describe

```
\protected\def\eql@displaybreak{%
  \relax
  \eql@ampprotecttwo\eql@teststaroropt@tight
    \eql@displaybreak@star\eql@displaybreak@level{4}%
}

\def\eql@displaybreak@star#1{%
  \global\eql@appendexpand\eql@interline@container{%
    \eql@displaybreak@pen@\the\numexpr#1\relax\relax}%
}

\def\eql@displaybreak@level[#1]{%
  \ifnum#1<\z@
    \global\eql@append\eql@interline@container{\eql@displaybreak@pen@\@MM}%
  \else
    \global\eql@appendexpand\eql@interline@container{%
      \eql@displaybreak@pen@-\@getpen{#1}\relax}%
  \fi
}
```

TODO: describe

```
\def\eql@displaybreak@pre#1{%
  \ifnum#1<\z@
    \eql@displaybreak@prepen@\@MM
  \else
    \eql@displaybreak@prepen@-\@getpen{#1}\relax
  \fi
}
```

TODO: describe

```
\def\eql@displaybreak@post#1{%
```

```
  \ifnum#1<\z@
    \eql@displaybreak@postpen@\@MM
  \else
    \eql@displaybreak@postpen@-\@getpen{#1}\relax
  \fi
}
```

TODO: describe

```
\def\eql@displaybreak@inter#1{%
  \ifnum#1<\z@
    \interdisplaylinepenalty\@M
  \else
    \interdisplaylinepenalty\eql@getdsp@pen{#1}\relax
  \fi
}
```

7.2 Explicit Vertical Space

TODO: describe

`eql@vspaceskip@` (*skip*)

```
\newskip\eql@vspaceskip@
```

```
\let\eql@vspace@org\vspace
\def\eql@vspace{%
  \ifvmode
    \expandafter\eql@vspace@immediate
  \else
    \expandafter\eql@vspace@line
  \fi
}
```

TODO: `\eql@vspace@addfixedafter` on last line has no effect. should apply outside environment

```
\def\eql@vspace@line{%
  \eql@ifstar@loose\eql@vspace@addfixedbefore\eql@vspace@add
}
\def\eql@vspace@add#1{%
  \global\eql@appendexpand\eql@interline@container{%
    \advance\eql@vspaceskip@\the\glueexpr#1\relax\relax}}
\def\eql@vspace@addfixedbefore#1{%
  \global\eql@appendexpand\eql@interline@container{%
    \noexpand\eql@append\noexpand\eql@display@injectbefore{%
      \skip@\the\glueexpr#1\relax\relax
      \penalty\@M
      \vskip\skip@
      \global\advance\eql@line@interline@\skip@
    }%
  }%
}
\def\eql@vspace@addfixedafter#1{%
  \global\eql@appendexpand\eql@interline@container{%
    \noexpand\eql@append\noexpand\eql@display@injectafter{%
      \dimen@\prevdepth
      \hrule\@height\z@
      \skip@\the\glueexpr#1\relax\relax
      \penalty\@M
      \vskip\skip@
      \global\advance\eql@line@interline@\skip@
      \prevdepth\dimen@
    }%
  }%
}
```

TODO: careful to not expand `\eql@display@container` after measure

```
\def\eql@vspace@immediate{%
  \noalign\bgroup
    \eql@ifstar@loose\eql@vspace@fixed\eql@vspace@discardable
}
\def\eql@vspace@fixed#1{%
    \skip@\glueexpr#1\relax
    \ifnum\eql@row@=\@ne
      \global\eql@appendexpand\eql@display@container{%
        \advance\eql@abovespace@\the\skip@\relax}%
    \else\ifnum\eql@row@>\eql@totalrows@
      \global\eql@appendexpand\eql@display@container{%
        \advance\eql@belowspace@\the\skip@\relax}%
    \else
      \dimen@\prevdepth
      \hrule\@height\z@
      \penalty\@M
      \vskip\skip@
      \global\advance\eql@line@interline@\skip@
      \prevdepth\dimen@
    \fi\fi
  \egroup
}
\def\eql@vspace@discardable#1{%
    \skip@\glueexpr#1\relax
    \ifnum\eql@row@=\@ne
      \global\eql@appendexpand\eql@display@container{%
        \advance\eql@abovespace@\the\skip@\relax}%
    \else\ifnum\eql@row@>\eql@totalrows@
      \global\eql@appendexpand\eql@display@container{%
        \advance\eql@belowspace@\the\skip@\relax}%
    \else
      \vskip\skip@
      \global\advance\eql@line@interline@\skip@
    \fi\fi
  \egroup
}
```

7.3 Default Vertical Spacing

TODO: describe

`\eql@strut` `\eql@strutbox@` Next follows a special internal strut which is supposed to match the height and the depth of a normal `\strut` minus `\normallineskiplimit` according to M. Spivak.

```
\newbox\eql@strutbox@
\def\eql@strut@depth{.3}
\def\eql@strut{\copy\eql@strutbox@}
\let\eql@strut@cell\eql@strut
```

```
\let\eql@strut@tag\eql@strut
\def\eql@strut@make{%
  \setbox\eql@strutbox@\hbox{%
    \@tempdimb\dimexpr
        \eql@strut@depth\normalbaselineskip+.5\normallineskiplimit\relax
    \@tempdima\dimexpr
        \normalbaselineskip-\normallineskiplimit-\@tempdimb\relax
    \vrule\@height\@tempdima\@depth\@tempdimb\@width\z@
  }
}
\AtBeginDocument{\eql@strut@make}
```

TODO: describe **TODO:** uses amsmath `\spread@equation`

```
\def\eql@spread@set{%
  \ifdefined\eql@spread@reset
    \lineskip\normallineskip
    \lineskiplimit\normallineskiplimit
    \baselineskip\normalbaselineskip
  \fi
  \eql@spread@\dimexpr\glueexpr\eql@spread@val\relax
      +\normalbaselineskip-\baselineskip\relax
  \ifdim\eql@spread@>\z@
    \openup\eql@spread@
    \ifdefined\spread@equation
      \let\spread@equation\@empty
    \fi
  \fi
}
```

7.4 Entry and Exit

TODO: describe

```
\let\eql@beamerbasecolor@apply\@empty
\let\eql@beamerbasecolor@avoid\@empty
\AddToHook{package/beamerbasecolor/after}{%
  \def\eql@beamerbasecolor@apply{%
    \donotcolorouterdisplaymaths
    \donotcoloroutermaths
    \beamer@setdisplaymathcolor
  }%
  \def\eql@beamerbasecolor@avoid{\donotcoloroutermaths}%
}
```

`eql@abovespace@` (*skip*)
`eql@belowspace@` (*skip*)

```
\newskip\eql@abovespace@
\newskip\eql@belowspace@
```

`\eql@display@enter`

```
\def\eql@display@enter{%
  \if@noskipsec\leavevmode\par\fi
  \ifvmode
    \eql@prevdepth@\prevdepth
    \nointerlineskip
    \noindent
  \else
    \eql@prevdepth@\maxdimen
  \fi
  \eql@beamerbasecolor@apply
}
```

`\eql@display@adjust`

```
\def\eql@display@adjust{%
  \ifdefined\eql@display@linewidth
    \displaywidth\glueexpr\eql@display@linewidth\relax
    \advance\displaywidth-\displayindent
  \fi
  \ifdefined\eql@display@marginleft
    \advance\displaywidth\displayindent
    \displayindent\glueexpr\eql@display@marginleft\relax
    \advance\displaywidth-\displayindent
  \fi
  \ifdefined\eql@display@marginright
    \advance\displaywidth-\glueexpr\eql@display@marginright\relax
  \fi
  \ifdim\displaywidth<\z@
    \displaywidth\z@
  \fi
}
```

`\eql@display@init`

```
\def\eql@display@init{%
  \let\eql@display@restore\eql@display@restore@active
  \let\displaybreak\eql@displaybreak
  \let\eql@vspace@org\vspace
  \let\vspace\eql@vspace
  \let\eqncontrol\eql@control
  \let\eql@display@injectbefore\@empty
  \let\eql@display@injectafter\@empty
  \let\eqnpunct\eql@punct@setnext
  \eql@spread@set
  \eql@strut@make
  \let\eql@frame@cmd\@undefined
}
```

`\eql@display@print`

```
\def\eql@display@print{%
  \eql@punct@top@set
  \let\eql@display@container\@empty
  \eql@display@firstavail@\z@
  \eql@display@aboveextend@\z@
  \eql@display@belowextend@\z@
  \global\let\eql@interline@container\eql@interline@container@clear
}
```

`@display@halign@init` **TODO:** describe

```
\def\eql@display@halign@init#1{%
  \eql@row@\z@
  \eql@prevgraf@\prevgraf
  \everycr{\noalign{%
```

```
    \global\advance\eql@row@\@ne
    \prevgraf\numexpr\prevgraf+\@ne\relax
    #1%
  }}%
}
```

TODO: how about penalty here? not for entry into display

```
\def\eql@display@halign@start{%
  \prevgraf\numexpr\eql@prevgraf@+\@ne\relax
  \ifdim\eql@prevdepth@=\maxdimen\else
    \prevdepth\eql@prevdepth@
  \fi
  \ifdim\prevdepth=-\@m\p@\else
    \ifdefined\eql@display@height
      \skip@\baselineskip
      \advance\skip@-\glueexpr\eql@display@height\relax
      \advance\skip@-\prevdepth\relax
      \ifdim\skip@<\lineskiplimit
        \skip@\lineskip
      \fi
      \advance\skip@-\eql@spread@\relax
      \vskip\skip@
      \nointerlineskip
    \else
      \vskip-\eql@spread@\relax
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@display@vspace{%
  \advance\abovedisplayskip\eql@abovespace@
  \advance\belowdisplayskip\eql@belowspace@
}
```

TODO: describe

```
\def\eql@display@vspace@native{%
  \advance\abovedisplayskip\eql@abovespace@
  \advance\belowdisplayskip\eql@belowspace@
  \advance\abovedisplayshortskip\eql@abovespace@
  \advance\belowdisplayshortskip\eql@belowspace@
}
```

TODO: describe

```
\def\eql@display@penalty{%
  \ifnum\eql@displaybreak@postpen@=\@MM\else
    \postdisplaypenalty\eql@displaybreak@postpen@
  \fi
  \ifnum\eql@displaybreak@pen@=\@MM\else
    \postdisplaypenalty\eql@displaybreak@pen@
  \fi
  \ifnum\eql@displaybreak@prepen@=\@MM\else
    \predisplaypenalty\eql@displaybreak@prepen@
  \fi
}
```

TODO: describe **TODO:** issue: `\vspace*{0pt}` has some effect if page is broken here

```
\def\eql@display@halign@end{%
  \eql@interline@container
  \eql@display@injectbefore
  \global\eql@prevgraf@\numexpr\prevgraf+\@ne\relax
  \ifdefined\eql@display@depth
    \prevdepth\glueexpr\eql@display@depth\relax
  \fi
}
```

`\eql@display@close` **TODO:** there seems to be an offset of 1em in predisplaysize towards actual content, nice. **TODO:** must not use setlength or setcounter when calc is loaded **TODO:** do we actually need penalty adjustments in case of paragraphs above or below?

```
\def\eql@display@close{%
  \eql@display@container
  \ifdim\eql@display@firstavail@<\z@
    \eql@display@firstavail@\z@
  \fi
  \eql@skip@mode@leave@\z@
  \ifdim\eql@prevdepth@=\maxdimen
    \ifdim\predisplaysize=-\maxdimen
      \eql@skip@mode@above@\eql@skip@mode@cont@above\relax
      \eql@skip@mode@below@\eql@skip@mode@cont@below\relax
    \else
      \eql@skip@mode@above@\z@
      \eql@skip@mode@below@\z@
      \advance\eql@display@firstavail@\displayindent
      \ifdim\eql@display@firstavail@>\predisplaysize
        \ifcase\eql@skip@mode@short\relax
        \or
          \eql@skip@mode@above@\@ne
        \or
          \eql@skip@mode@above@\@ne
          \ifnum\eql@totalrows@=\@ne
            \eql@skip@mode@below@\@ne
          \fi
        \or
          \eql@skip@mode@above@\@ne
          \eql@skip@mode@below@\@ne
        \fi
      \fi
    \fi
  \else
    \ifdim\eql@prevdepth@=-\@m\p@
      \eql@skip@mode@above@\eql@skip@mode@top@above\relax
      \eql@skip@mode@below@\eql@skip@mode@top@below\relax
    \else
      \eql@skip@mode@above@\eql@skip@mode@par@above\relax
      \eql@skip@mode@below@\eql@skip@mode@par@below\relax
    \fi
  \fi
  \ifcase\eql@skip@mode@above@
  \or\or\or
    \predisplaypenalty\z@
  \or
    \predisplaypenalty\z@
  \fi
  \ifcase\eql@skip@mode@below@
  \or\or\or
    \eql@skip@mode@leave@\@ne
  \or
    \eql@skip@mode@leave@\tw@
  \fi
  \ifdefined\eql@skip@force@above
    \eql@skip@mode@above@\eql@skip@force@above\relax
  \fi
  \ifdefined\eql@skip@force@below
    \eql@skip@mode@below@\eql@skip@force@below\relax
  \fi
  \ifdefined\eql@skip@force@leave
    \eql@skip@mode@leave@\eql@skip@force@leave\relax
  \fi
  \ifnum\eql@skip@mode@leave@>\z@
    \postdisplaypenalty\z@
  \fi
  \ifcase\eql@skip@mode@above@
    \abovedisplayskip\glueexpr\eql@skip@long@above\relax
  \or
    \abovedisplayskip\glueexpr\eql@skip@short@above\relax
  \or
    \abovedisplayskip\glueexpr\eql@skip@cont@above\relax
  \or
    \abovedisplayskip\glueexpr\eql@skip@par@above\relax
  \or
    \abovedisplayskip\glueexpr\eql@skip@top@above\relax
  \or
    \abovedisplayskip\z@skip
  \or
    \abovedisplayskip\glueexpr\eql@skip@med@above\relax
  \or
    \abovedisplayskip\glueexpr\eql@skip@custom@above\relax
  \fi
  \ifcase\eql@skip@mode@below@
    \belowdisplayskip\glueexpr\eql@skip@long@below\relax
  \or
    \belowdisplayskip\glueexpr\eql@skip@short@below\relax
  \or
    \belowdisplayskip\glueexpr\eql@skip@cont@below\relax
  \or
    \belowdisplayskip\glueexpr\eql@skip@par@below\relax
  \or
    \belowdisplayskip\glueexpr\eql@skip@top@below\relax
  \or
    \belowdisplayskip\z@skip
  \or
    \belowdisplayskip\glueexpr\eql@skip@med@below\relax
  \or
    \belowdisplayskip\glueexpr\eql@skip@custom@below\relax
  \fi
  \global\eql@skip@mode@leave@\eql@skip@mode@leave@
  \eql@interline@container
  \advance\eql@belowspace@\eql@vspaceskip@
  \eql@display@penalty
  \eql@display@vspace
  \skip@\glueexpr\eql@skip@tag@above\relax
  \ifdim\skip@>\abovedisplayskip
    \skip@\abovedisplayskip
  \fi
  \advance\abovedisplayskip-\eql@display@aboveextend@\relax
  \ifdim\abovedisplayskip<\skip@
    \abovedisplayskip\skip@
  \fi
  \skip@\glueexpr\eql@skip@tag@below\relax
  \ifdim\skip@>\belowdisplayskip
    \skip@\belowdisplayskip
  \fi
  \ifdim\eql@display@belowextend@>\z@
    \advance\belowdisplayskip-\eql@display@belowextend@\relax
    \ifdim\belowdisplayskip<\skip@
      \belowdisplayskip\skip@
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@display@leave{%
  \prevgraf\eql@prevgraf@
  \ifcase\eql@skip@mode@leave@
  \or
    \endgraf
  \or
    \endgraf
    \prevdepth-\@m\p@
  \fi
}
```

TODO: describe

```
\def\eql@display@nest{%
  \let\displaybreak\eql@displaybreak@default
  \let\intertext\eql@intertext@default
  \let\vspace\eql@vspace@org
}
```

TODO: describe **TODO:** box version?! (but also consider nesting)

```
\def\eql@display@restore@active{%
  \let\label\eql@label@org
  \let\tag\eql@tag@default
  \let\raisetag\eql@raisetag@default
  \let\displaybreak\eql@displaybreak@default
  \let\intertext\eql@intertext@default
  \let\vspace\eql@vspace@org
  \ifdefined\eql@amp@mode
    \let\&\eql@amp@org
  \fi
  \let\eqnpunct\eql@punct@addopt
  \let\eql@punct@block\@undefined
  \let\eql@display@restore\@empty
}
```

TODO: describe

```
\let\eql@display@restore\@empty
\eql@append\@arrayparboxrestore{%
  \eql@display@restore
  \ifdefined\eql@ampproof@active
```

```
    \eql@amprevert
  \fi
  \@displayfalse
}
```

7.5 Stack

TODO: describe **TODO:** for each global variable declare global nature at its definition! **TODO:** we must be consistent about global variables vs local variables global variables need to be saved at every level where they may be modified (even if modified only locally)

```
\def\eql@stack@enable{%
  \let\eql@stack@save@equations\eql@stack@save@equations@
  \let\eql@stack@save@box\eql@stack@save@box@
}
```

TODO: describe

```
\let\eql@stack@save@equations\eql@stack@enable
\let\eql@stack@save@box\eql@stack@enable
\let\eql@stack@restore\@empty
```

TODO: describe

```
\def\eql@stack@save@reg#1{\global#1\the#1\relax}
\def\eql@stack@save@let#1#2{\global\let\noexpand#2\noexpand#1}
```

TODO: further global variables: global registers: `\eql@nextopt`, `\eql@tags@glabel@` used locally without possibility of change between setting and retrieving: `\eql@prevgraf@`, `\eql@skip@mode@leave@`, `\eql@shape@lastrow`, `\eql@frame@prevcmd` **TODO:** to be reviewed: `\eql@intertext@after`, `\eql@intertext@opt` **TODO:** describe

```
\def\eql@stack@save@equations@{%
  \let\eql@stack@numbering@eqnswinit\eql@numbering@eqnswinit
  \let\eql@stack@cell@container\eql@cell@container
  \let\eql@stack@tags@container\eql@tags@container
  \let\eql@stack@interline@container\eql@interline@container
  \let\eql@stack@dimensions@tab\eql@dimensions@tab
  \let\eql@stack@block@container\eql@display@container
  \let\eql@stack@punct@top\eql@punct@top
  \edef\eql@stack@restore{%
    \global\if@eqnsw\noexpand\@eqnswtrue\else\noexpand\@eqnswfalse\fi
    \eql@stack@save@let\eql@stack@numbering@eqnswinit\eql@numbering@eqnswinit
    \eql@stack@save@let\eql@stack@cell@container\eql@cell@container
    \eql@stack@save@let\eql@stack@tags@container\eql@tags@container
    \eql@stack@save@let\eql@stack@interline@container\eql@interline@container
    \eql@stack@save@let\eql@stack@dimensions@tab\eql@dimensions@tab
    \eql@stack@save@let\eql@stack@block@container\eql@display@container
    \eql@stack@save@let\eql@stack@punct@top\eql@punct@top
    \eql@stack@save@reg\eql@column@
    \eql@stack@save@reg\eql@totalcolumns@
    \eql@stack@save@reg\eql@line@avail@
    \eql@stack@save@reg\eql@line@pos@
    \eql@stack@save@reg\eql@line@width@
    \eql@stack@save@reg\eql@line@depth@
    \eql@stack@save@reg\eql@line@height@
    \eql@stack@save@reg\eql@line@prevdepth@
    \eql@stack@save@reg\eql@line@interline@
    \eql@stack@save@reg\eql@totalheight@
    \eql@stack@save@reg\eql@tagwidth@max@
    \eql@stack@save@reg\eql@tagpos@row@
    \eql@stack@save@reg\eql@row@
    \eql@stack@save@reg\eql@tagrows@
  }%
}
```

TODO: describe

```
\def\eql@stack@save@box@{%
  \let\eql@stack@cell@container\eql@cell@container
  \edef\eql@stack@restore{%
    \eql@stack@save@let\eql@stack@cell@container\eql@cell@container
    \eql@stack@save@reg\eql@row@
  }%
}
```

8 Multi-Line Support

TODO: describe

8.1 Measure Support

TODO: describe

```
\def\eql@measure@init#1#2{%
  \eql@dimensions@reset
  \let\eql@display@container\@empty
  \eql@numbering@measure@init
  \eql@row@\z@
  \eql@totalheight@\z@
  \eql@totalrows@\@M
  \eql@line@prevdepth@-\@m\p@
  \eql@line@interline@\z@
  \tabskip\z@skip
  \everycr{\noalign{%
    \global\advance\eql@row@\@ne
    #1%
  }}%
  \eql@punct@top@set
  \global\let\eql@interline@container\eql@interline@container@clear
  \eql@measure@savestate
  \eql@multi@cr@let{#2}%
}
```

TODO: describe

```
\def\eql@measure@tag{%
  \eql@tagwidth@\z@
  \ifdefined\eql@numbering@multi
    \if@eqnsw
      \eql@tags@container
      \eql@tagbox@make\eql@composetag@measure
      \ifdefined\eql@tagpos@reserve\else
        \eql@tagwidth@\z@
      \fi
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@measure@endrow{%
  \ifdim\eql@line@prevdepth@=-\@m\p@\else
    \dimen@\dimexpr\baselineskip-\eql@line@height@-\eql@line@prevdepth@\relax
    \ifdim\dimen@<\lineskiplimit
      \dimen@\lineskip
    \fi
    \advance\eql@line@interline@\dimen@
  \fi
  \eql@dimensions@endrow
  \ifdim\eql@tagwidth@>\eql@tagwidth@max@
    \global\eql@tagwidth@max@\eql@tagwidth@
  \fi
  \ifdim\eql@tagwidth@>\z@
    \global\advance\eql@tagrows@\@ne
  \fi
  \global\advance\eql@totalheight@\dimexpr
    \eql@line@interline@+\eql@line@height@+\eql@line@depth@
  \global\eql@line@interline@\z@
  \global\eql@line@prevdepth@\eql@line@depth@
}
```

TODO: describe

```
\def\eql@measure@close{%
  \advance\eql@row@-\tw@
  \eql@totalrows@\eql@row@
  \ifnum\eql@totalrows@>\z@
    \eql@dimensions@get\@ne
    \eql@topheight@\dimexpr\eql@line@height@+\eql@line@interline@\relax
    \eql@dimensions@get\eql@totalrows@
    \eql@bottomdepth@\eql@line@depth@
  \fi
  \eql@numbering@measure@blocktag
  \begingroup
    \eql@tags@container
    \if@eqnsw
      \eql@tagbox@make\eql@composetag@measure
      \ifdefined\eql@tagpos@reserve\else
        \eql@tagwidth@\z@
      \fi
      \eql@dimensions@saveblocktag
    \else
      \eql@dimensions@savenoblocktag
      \eql@numbering@warnunused
    \fi
  \endgroup
  \eql@dimensions@get\z@
  \eql@measure@restorestate
}
```

`measure@restorestate`
`ql@measure@savestate`

```
\let\eql@measure@restorestate\@empty
\def\eql@measure@savestate{%
  \begingroup
```

```
    \def\@elt##1{%
      \global\csname c@##1\endcsname\the\csname c@##1\endcsname}%
    \global\edef\@gtempa{\cl@@ckpt}%
  \endgroup
  \let\eql@measure@restorestate\@gtempa
}
```

8.2 Line Breaks

TODO: describe

\eql@multi@cr

```
\def\eql@multi@cr{%
  \let\eql@punct@term\eql@false
  \let\eql@class@rel@composed\@empty
  \eql@ampprotect\eql@multi@cr@test\eql@multi@cr@process}
```

TODO: describe

```
\def\eql@multi@cr@test@setopt{%
  \let\eql@multi@cr@test\eql@multi@cr@testopt}
\def\eql@multi@cr@test@setall{%
  \let\eql@multi@cr@test\eql@multi@cr@testall}
```

eql@multi@cr@testopt

TODO: describe

```
\def\eql@multi@cr@testopt#1{\eql@teststaropt@tight
  {\eql@displaybreak@star\@M\eql@multi@cr@testopt@set{#1}}%
  {\eql@multi@cr@testopt@set{#1}}{0pt}}
\def\eql@multi@cr@testopt@set#1[#2]{\eql@vspace@add{#2}#1}
```

eql@multi@cr@testall
lti@cr@testall@parse

TODO: describe

```
\def\eql@multi@cr@testall{\eql@parseopt@cr\eql@multi@cr@testall@parse}
\def\eql@multi@cr@testall@parse{%
  \ifx\eql@parseopt@token[%
    \let\eql@parseopt@next\eql@multi@cr@parse@vspace
  \fi
  \ifx\eql@parseopt@token*%
    \let\eql@parseopt@next\eql@multi@cr@parse@star
  \fi
  \ifx\eql@parseopt@token.%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token,%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token~%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token`%
    \let\eql@parseopt@next\eql@parseopt@punctnext
  \fi
  \ifx\eql@parseopt@token!%
    \let\eql@parseopt@next\eql@parseopt@puncterm
  \fi
  \ifx\eql@parseopt@token/%
```

```
    \let\eql@parseopt@next\eql@multi@cr@parse@break
  \fi
  \ifx\eql@parseopt@token=%
    \let\eql@parseopt@next\eql@parseopt@relsymb
  \fi
  \ifx\eql@parseopt@token;%
    \let\eql@parseopt@next\eql@parseopt@relcont
  \fi
  \ifx\eql@parseopt@token:%
    \let\eql@parseopt@next\eql@parseopt@relstart
  \fi
  \ifx\eql@parseopt@token|%
    \let\eql@parseopt@next\eql@parseopt@relord
  \fi
  \ifx\eql@parseopt@token?%
    \let\eql@parseopt@next\eql@multi@cr@parse@rel
  \fi
  \ifx\eql@parseopt@token&%
    \let\eql@parseopt@next\eql@parseopt@end
  \fi
}
\def\eql@multi@cr@parse@vspace[#1]{\eql@vspace@add{#1}\eql@parseopt@peek}
\def\eql@multi@cr@parse@star#1{\eql@displaybreak@star\@M\eql@parseopt@peek}
\def\eql@multi@cr@parse@break{\numbernext\eql@parseopt@punctclear}
\def\eql@multi@cr@parse@rel#1#2{%
  \def\eql@tmp{#2}%
  \ifx\eql@tmp\eql@relax\else
    \ifdefined\eql@punct@next\else
      \eql@punct@next@clear
    \fi
    \ifdefined\eql@multi@cr@relnext\numbernext\fi
  \fi
  \ifdefined\eql@multi@linesmode
    \ifx\eql@tmp\@empty
      \def\eql@class@rel@composed{\eql@shape@cont}%
    \else
      \def\eql@class@rel@composed{\eql@shape@rel#2}%
    \fi
  \else
    \def\eql@class@rel@composed{\eql@class@rel@make{#2}}%
  \fi
  \eql@parseopt@end}
```

eql@multi@cr@process

```
\def\eql@multi@cr@process{%
  \ifdefined\eql@punct@term\eql@punct@apply@top\fi
  \edef\eql@tmp{%
    \unexpanded{%
      \eql@multi@endline
      \cr
      \eql@multi@cr@interline
    }%
    \unexpanded\expandafter{\eql@class@rel@composed}%
  }%
  \eql@tmp
}
```

```
\def\eql@multi@cr@interline{%
```

```
  \noalign{%
    \eql@interline@container
    \eql@display@injectbefore
    \ifnum\eql@displaybreak@pen@=\@MM
      \penalty\interdisplaylinepenalty
    \else
      \penalty\eql@displaybreak@pen@
    \fi
    \vskip\eql@vspaceskip@
    \global\advance\eql@line@interline@\eql@vspaceskip@
    \eql@display@injectafter
    \global\let\eql@interline@container\eql@interline@container@clear
  }%
}
```

`\eql@multi@cr@let`

```
\def\eql@multi@cr@let#1{%
  \let\\\eql@multi@cr
  \let\eql@multi@endline#1%
}
```

8.3 Intertext

TODO: describe

TODO: revert in everymath?

```
\def\eql@intertext@default{\eql@error{Invalid use of \string\intertext}}
\eql@amsmath@let\intertext\eql@intertext@default
```

TODO: why does it fail in measuring? total width?! determine total width otherwise!?

```
\def\eql@intertext@process{%
  \eql@multi@endline
  \cr
  \ifmeasuring@
    \expandafter\@gobble
  \else
    \expandafter\eql@intertext@print
  \fi
}
```

TODO: describe **TODO:** prevdepth **TODO:** does this have to be in a vbox? **TODO:** vskip and penalty opposite order **TODO:** can we handle short? certainly needs two passes

```
\def\eql@intertext@print#1{%
  \noalign{%
    \eql@display@halign@end
    \let\eql@skip@force@below\z@
    \let\eql@skip@force@above\z@
    \eql@setkeys{intertext}\eql@intertext@opt
    \openup-\eql@spread@
    \penalty\postdisplaypenalty
    \ifcase\eql@skip@force@below\relax
      \advance\eql@vspaceskip@\glueexpr\eql@skip@long@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@short@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@cont@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@par@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@top@below\relax
    \or
      \advance\eql@vspaceskip@\z@skip
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@med@below\relax
    \or
      \advance\eql@vspaceskip@\glueexpr\eql@skip@custom@below\relax
    \fi
    \vskip\eql@vspaceskip@
    \global\let\eql@interline@container\eql@interline@container@clear
    \vbox{%
      \@parboxrestore
      \ifdim
        \ifdim\@totalleftmargin=\z@\linewidth\else-\maxdimen\fi=\columnwidth
      \else
        \parshape\@ne
        \@totalleftmargin\linewidth
      \fi
      \noindent
      \prevgraf\eql@prevgraf@
      \ignorespaces
      #1%
      \par
      \global\eql@prevgraf@\prevgraf
    }%
    \penalty\predisplaypenalty
    \ifcase\eql@skip@force@above\relax
      \vskip\glueexpr\eql@skip@long@above\relax
    \or
      \vskip\glueexpr\eql@skip@short@above\relax
    \or
      \vskip\glueexpr\eql@skip@cont@above\relax
    \or
      \vskip\glueexpr\eql@skip@par@above\relax
    \or
      \vskip\glueexpr\eql@skip@top@above\relax
    \or
      \vskip\z@skip
    \or
      \vskip\glueexpr\eql@skip@med@above\relax
    \or
      \vskip\glueexpr\eql@skip@custom@above\relax
    \fi
%    \eql@prevdepth@\maxdimen
    \eql@prevdepth@\z@
    \eql@display@halign@start
  }
}
```

TODO: describe

```
\newenvironment{eql@intertext}{%
  \eql@testopt@tight\eql@intertext@{}%
}{%
  \aftergroup\eql@intertext@after
  \ignorespacesafterend
}
```

TODO: describe

```
\def\eql@intertext@env{intertext}
\def\eql@intertext@[#1]{%
  \global\def\eql@intertext@opt{#1}%
  \ifx\@currenvir\eql@intertext@env
    \def\eql@scan@call{\eql@intertext@inject\eql@scan@end}%
    \expandafter\eql@scan@env
  \else
    \expandafter\eql@intertext@process
  \fi
}
```

TODO: describe

```
\def\eql@intertext@inject{%
  \global\edef\eql@intertext@after{%
    \noexpand\eql@intertext@process{%
      \ifx\eql@scan@body\eql@scan@body@dump
        \eql@scan@body@dump
      \else
        \noexpand\scantokens{\eql@scan@body@dump}%
      \fi
    }%
  }%
}
```

8.4 Line Marks

TODO: describe

```
\def\eql@markline@pos@below{below}
\def\eql@markline@pos@bottom{bottom}
\def\eql@markline@pos@baseline{baseline}
\let\eql@markline@pos\eql@markline@pos@baseline
\let\eql@markline@shift\z@
\def\eql@markline@qed{\ifdefined\qedsymbol\qedsymbol\else QED\fi}
\def\eql@markline@symbol{}
```

TODO: describe

```
\def\eql@markline@select#1{%
  \let\eql@markline@shift\z@
  \eql@setkeys{markline}{#1}%
  \eql@markline@print
}
```

TODO: describe

```
\def\eql@markline@print{%
  \dimen@\dimexpr\eql@markline@shift\relax
  \ifx\eql@markline@pos\eql@markline@pos@below
    \ifdim\dimen@=\z@\else
      \penalty\@M
      \vskip-\dimen@
    \fi
    \nointerlineskip
    \penalty\@M
    \vbox{\hfill\hbox{\eql@markline@symbol}}%
  \else
    \ifx\eql@markline@pos\eql@markline@pos@baseline
      \advance\dimen@\prevdepth
    \fi
    \setbox\z@\hbox{\raise\dimen@\hbox{\eql@markline@symbol}}%
    \dimen@\prevdepth
    \ht\z@\z@
    \dp\z@\z@
    \nointerlineskip
    \penalty\@M
    \vbox{\hfill\box\z@}%
    \prevdepth\dimen@
  \fi
}
```

TODO: describe

```
\def\eql@markline@inject#1{%
  \let\eql@markline@push\eql@false
  \ifx\eql@markline@pos\eql@markline@pos@below\else
    \ifdefined\eql@tagsleft\else
      \ifx\eql@equations@main\eql@multi@main
        \ifdefined\eql@numbering@multi
          \if@eqnsw
            \let\eql@markline@push\eql@true
          \fi
        \else
          \ifnum\eql@row@=\eql@tagpos@row@
            \let\eql@markline@push\eql@true
          \fi
        \fi
      \else
        \if@eqnsw
          \let\eql@markline@push\eql@true
        \fi
      \fi
    \fi
  \fi
  \ifdefined\eql@markline@push
    \global\eql@append\eql@interline@container{%
      \eql@append\eql@display@injectbefore{\eql@markline@select{push,#1}}}%
  \else
    \global\eql@append\eql@interline@container{%
      \eql@append\eql@display@injectbefore{\eql@markline@select{#1}}}%
  \fi
}
```

TODO: describe

```
\def\eql@markline@amsthm@opt[#1]{\eql@markline@inject{qed,#1}}
\def\eql@markline@amsthm@staropt[#1]{\eql@markline@inject{qed,push,#1}}
\def\eql@markline@amsthm@qed{\eql@teststaropt@tight
  \eql@markline@amsthm@staropt\eql@markline@amsthm@opt{}}
\def\eql@markline@amsthm@register#1{\eql@letcs{#1@qed}\eql@markline@amsthm@qed}
\def\eql@markline@amsthm@move#1#2{%
  \AddToHook{package/amsthm/after}{%
    \eql@letcs{#1@qed\expandafter}\csname#2@qed\endcsname}}
```

9 Column Placement

TODO: describe

9.1 Supporting Definitions

`l@shape@pos@` (*dimen*)
`shape@amount@` (*dimen*)

The registers `\eql@shape@pos@` and `\eql@shape@amount@` specify the currently selected horizontal alignment (`0` for left, `1` for center, `2` for right) and the indentation amount, respectively:

```
\newcount\eql@shape@pos@
\newdimen\eql@shape@amount@
\let\eql@shape@lastrow\eql@false
```

`l@marginleft@` (*dimen*)
`rginleft@min@` (*dimen*)
`@marginright@` (*dimen*)
`centeroffset@` (*dimen*)

The registers `\eql@marginleft@` and `\eql@marginright@` store the intended left and right margin for the equation lines: **TODO:** update

```
\newdimen\eql@marginleft@
\newdimen\eql@marginright@
\newdimen\eql@marginleft@min@
\newdimen\eql@centeroffset@
```

9.2 Shape Schemes

The horizontal alignment of each line is specified by a shape scheme.

`\eql@shape@tab@...`

We select the scheme through a `\csname` selector with the following names:

```
\def\eql@shape@tab@default{default}
\def\eql@shape@tab@left{left}
\def\eql@shape@tab@center{center}
\def\eql@shape@tab@right{right}
\def\eql@shape@tab@first{first}
\def\eql@shape@tab@hanging{hanging}
\def\eql@shape@tab@steps{steps}
```

For convenience, we add further alias names for the schemes:

```
\let\eql@shape@tab@def\eql@shape@tab@default
\let\eql@shape@tab@\eql@shape@tab@default
\let\eql@shape@tab@l\eql@shape@tab@left
\let\eql@shape@tab@c\eql@shape@tab@center
\let\eql@shape@tab@r\eql@shape@tab@right
\let\eql@shape@tab@rc\eql@shape@tab@first
\let\eql@shape@tab@indent\eql@shape@tab@first
\let\eql@shape@tab@hang\eql@shape@tab@hanging
\let\eql@shape@tab@lc\eql@shape@tab@hanging
\let\eql@shape@tab@outdent\eql@shape@tab@hanging
\let\eql@shape@tab@lcr\eql@shape@tab@steps
```

`\eql@shape@mode`

The currently selected scheme is stored in `\eql@shape@mode`. It it set to `default`:

```
\let\eql@shape@mode\eql@shape@tab@default
```

`\eql@shape@set`

Set the scheme via the translation table:

```
\def\eql@shape@set#1{%
```

```
  \ifcsname eql@shape@tab@#1\endcsname
    \expandafter\let\expandafter\eql@shape@mode
      \csname eql@shape@tab@#1\endcsname
  \else
    \eql@error{shape '#1' unknown: setting to default}%
    \let\eql@shape@mode\eql@shape@tab@default
  \fi
}
```

`ape@layoutcenter@...`
`shape@layoutleft@...`

Define the uniform shape schemes `left`, `center`, `right` and `default` for the central and left alignment layout. The scheme functions determine the desired alignment and indentation for the current row:

```
\def\eql@shape@layoutcenter@left{\eql@shape@pos@\z@\eql@shape@amount@\z@}
\def\eql@shape@layoutcenter@center{\eql@shape@pos@\@ne\eql@shape@amount@\z@}
\def\eql@shape@layoutcenter@right{\eql@shape@pos@\tw@\eql@shape@amount@\z@}
\let\eql@shape@layoutcenter@default\eql@shape@layoutcenter@center
\def\eql@shape@layoutleft@left{\eql@shape@pos@\z@\eql@shape@amount@\z@}
\def\eql@shape@layoutleft@center{\eql@shape@pos@\@ne\eql@shape@amount@\z@}
\def\eql@shape@layoutleft@right{\eql@shape@pos@\tw@\eql@shape@amount@\z@}
\let\eql@shape@layoutleft@default\eql@shape@layoutleft@left
```

The `first` scheme implements left alignment with indentation for the first line (unless there is only one line):

```
\def\eql@shape@layoutcenter@first{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\z@
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@\eql@indent@
    \fi
  \fi
}
\def\eql@shape@layoutleft@first{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\z@
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@\eql@indent@
    \fi
  \fi
}
```

The `hanging` scheme implements left alignment with hanging indentation for the first line (unless there is only one line). In centeral alignment layout all but the first line are indented while in left aligned layout the first line has negative indentation:

```
\def\eql@shape@layoutcenter@hanging{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\eql@indent@
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@\z@
    \fi
  \fi
}
\def\eql@shape@layoutleft@hanging{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\z@
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@-\eql@indent@
    \fi
  \fi
}
```

The `steps` scheme implements singles out the first and last lines which are shifted left and right, respectively. In centeral alignment layout the shift operates on the alignment whereas in left alignment layout the shift uses indentation:

```
\def\eql@shape@layoutcenter@steps{%
  \eql@shape@amount@\z@
  \eql@shape@pos@\@ne
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@pos@\z@
    \fi
    \ifnum\eql@row@=\eql@totalrows@
      \eql@shape@pos@\tw@
    \fi
  \fi
}
\def\eql@shape@layoutleft@steps{%
  \eql@shape@pos@\z@
  \eql@shape@amount@\z@
  \ifnum\eql@totalrows@>\@ne
    \ifnum\eql@row@=\@ne
      \eql@shape@amount@-\eql@indent@
    \fi
    \ifnum\eql@row@=\eql@totalrows@
      \eql@shape@amount@\eql@indent@
    \fi
  \fi
}
```

`\eql@shape@select`
`\eql@shape@eval`

Select the shape selector function for the currrent scheme `@\eql@shape@mode` and layout and store it in `\eql@shape@eval`:

```
\let\eql@shape@eval\@undefined
\def\eql@shape@select{%
  \expandafter\let\expandafter\eql@shape@eval
    \csname eql@shape%
    @\ifdefined\eql@layoutleft layoutleft\else layoutcenter\fi
    @\eql@shape@mode\endcsname
}
```

`\eql@shape@alignleft`
`eql@shape@alignright`
`ql@shape@aligncenter`

Adjust the alignment of the current equation line. The optional argument specifies the amount of indentation:

```
\protected\def\eql@shape@alignleft{%
  \global\eql@append\eql@cell@container{\eql@shape@pos@\z@}%
  \eql@ampprotect\eql@shape@align@testpar\eql@shape@alignamount@opt}
\protected\def\eql@shape@aligncenter{%
  \global\eql@append\eql@cell@container{\eql@shape@pos@\@ne}%
  \eql@ampprotect\eql@shape@align@testpar\eql@shape@alignamount@opt}
\protected\def\eql@shape@alignright{%
  \global\eql@append\eql@cell@container{\eql@shape@pos@\tw@}%
  \eql@ampprotect\eql@shape@align@testpar\eql@shape@alignamount@opt}
```

```
\def\eql@shape@align@testpar#1{%
  \eql@ifstar@tight{#1[\eql@indent@]}%
  {\eql@ifnextgobble@tight{!}{#1[-\eql@indent@]}%
  {\eql@testopt@tight{#1}\z@}}}
\def\eql@shape@alignamount@opt[#1]{\eql@shape@alignamount@set{#1}}
```

`ql@shape@alignamount`

TODO: describe

```
\protected\def\eql@shape@alignamount{%
  \eql@ampprotecttwo\eql@ifstar@tight
    \eql@shape@alignamount@set\eql@shape@alignamount@add}
\def\eql@shape@alignamount@add#1{%
  \global\eql@appendexpand\eql@cell@container{%
    \advance\eql@shape@amount@\the\glueexpr#1\relax\relax}}
\def\eql@shape@alignamount@set#1{%
  \global\eql@appendexpand\eql@cell@container{%
    \eql@shape@amount@\the\glueexpr#1\relax\relax}}
\def\eql@shape@align@enable{%
  \let\shoveleft\eql@shape@alignleft
  \let\shovecenter\eql@shape@aligncenter
  \let\shoveright\eql@shape@alignright
  \let\shoveby\eql@shape@alignamount
}
```

TODO: describe

```
\protected\def\eql@shape@align@default{%
  \eql@warn@here{\shove...}%
  \eql@ampprotect\eql@shape@align@testpar\eql@gobbleopt}
\protected\def\eql@shape@alignamount@default{%
  \eql@warn@here{\shove...}%
  \eql@ampprotecttwo\eql@ifstar@tight\@gobble\@gobble}
\def\eql@shape@align@disable{%
  \let\shoveleft\eql@shape@align@default
  \let\shovecenter\eql@shape@align@default
  \let\shoveright\eql@shape@align@default
  \let\shoveby\eql@shape@alignamount@default
}
```

9.3 Width Data

`gwidth@block@` (*dimen*)

```
\newdimen\eql@tagwidth@block@
\newdimen\eql@tagheight@block@
\newdimen\eql@tagdepth@block@
```

`\eql@dimensions@tab`

TODO: new

```
\let\eql@dimensions@tab\@empty
```

`eql@dimensions@reset`

```
\def\eql@dimensions@reset{%
  \let\eql@dimensions@tab\@empty
  \eql@tagwidth@max@\z@
  \eql@tagrows@\z@
}
```

\eql@dimensions@add

```
\def\eql@dimensions@add#1{%
  \global\eql@appendexpand\eql@dimensions@tab{#1}%
}
```

ql@dimensions@addreg

```
\def\eql@dimensions@addreg#1{#1\the#1\relax}
```

@dimensions@startrow

```
\def\eql@dimensions@startrow{%
  \eql@dimensions@add{\eql@dimensions@addreg\eql@row@}%
}
```

@dimensions@savecell

```
\def\eql@dimensions@savecell{%
  \eql@dimensions@add{%
    \eql@dimensions@addreg\eql@shape@pos@
    \eql@dimensions@addreg\eql@cellwidth@
    \eql@dimensions@addreg\eql@shape@amount@
    \noexpand\eql@dimensions@cellcall
  }%
}
```

l@dimensions@savesep

```
\def\eql@dimensions@savesep{%
  \eql@dimensions@add{\noexpand\eql@dimensions@sepcall}%
}
```

ql@dimensions@endrow

```
\def\eql@dimensions@endrow{%
  \eql@dimensions@add{,%
    \eql@dimensions@addreg\eql@tagwidth@
    \eql@dimensions@addreg\eql@line@height@
    \eql@dimensions@addreg\eql@line@depth@
    \eql@dimensions@addreg\eql@line@interline@
    ;}%
}
```

ensions@saveblocktag

```
\def\eql@dimensions@saveblocktag{%
  \eql@dimensions@add{\eql@row@0\relax,%
    \eql@tagwidth@block@\the\eql@tagwidth@\relax
    \eql@tagheight@block@\the\ht\eql@tagbox@\relax
    \eql@tagdepth@block@\the\dp\eql@tagbox@\relax
    \eql@dimensions@addreg\eql@tagpos@shift@
    \let\noexpand\eql@tagpos@reserve\ifdefined\eql@tagpos@reserve
      \noexpand\eql@true\else\noexpand\eql@false\fi
  ;}%
  \global\eql@tagwidth@max@\eql@tagwidth@
  \global\eql@tagrows@\@ne
}
```

sions@savenoblocktag

```
\def\eql@dimensions@savenoblocktag{%
  \eql@dimensions@add{\eql@row@0\relax,;}%
}
```

\eql@dimensions@for

```
\def\eql@dimensions@for#1{%
  \def\eql@dimensions@forcall{#1}%
  \expandafter\eql@dimensions@forstep\eql@dimensions@tab
}
```

l@dimensions@forstep

```
\def\eql@dimensions@forstep\eql@row@#1\relax#2,#3;{%
  \eql@row@#1\relax
  \ifnum\eql@row@=\z@\else
    #3%
    \def\eql@dimensions@cells{#2}%
    \eql@dimensions@forcall
    \expandafter\eql@dimensions@forstep
  \fi
}
```

\eql@dimensions@get

```
\def\eql@dimensions@get#1{%
  \eql@row@#1\relax
  \expandafter\eql@dimensions@getdef\expandafter{\the\eql@row@}%
  \expandafter\eql@dimensions@getparse\eql@dimensions@tab\@nil
}
```

ql@dimensions@getdef

```
\def\eql@dimensions@getdef#1{%
  \def\eql@dimensions@getparse
    ##1\eql@row@#1\relax##2,##3;##4\@nil{%
    ##3%
    \def\eql@dimensions@cells{##2}%
  }%
}
```

\eql@colwidth@tab

```
\let\eql@colwidth@tab\@empty
```

\eql@colwidth@get

```
\def\eql@colwidth@get#1{%
  \ifcase\expandafter#1\eql@colwidth@tab\else\z@\fi
}
```

\eql@colwidth@save

```
\def\eql@colwidth@save#1{%
  \edef\eql@colwidth@tab{%
    \noexpand\or\the#1%
    \unexpanded\expandafter{\eql@colwidth@tab}%
  }%
}
```

`\eql@dimensions@calc` Compute the space that is available at the beginning and at the end of the row stored in `\eql@dimensions@cells`. The space available at the beginning is returned in `\eql@line@avail@`. and `\eql@line@availsep@` describes the number of unused intercolumn separations. The total used width is returned in `\eql@line@width@` and `\eql@line@widthsep@` describes the number of used intercolumn separations. The available space at the end of the row is given as the difference to `\eql@totalwidth@`:

```
\def\eql@dimensions@calc{%
  \eql@column@\z@
  \eql@line@pos@\z@
  \eql@line@possep@\z@
  \eql@line@avail@\eql@totalwidth@
  \eql@line@availsep@\eql@intercolumns@
  \eql@line@width@\z@
  \eql@line@widthsep@\z@
  \let\eql@dimensions@cellcall\eql@dimensions@calc@call
  \let\eql@dimensions@sepcall\eql@dimensions@calc@callsep
  \eql@dimensions@cells
}
```

`ensions@calc@callsep` Callback for each intercolumn space.

```
\def\eql@dimensions@calc@callsep{%
  \advance\eql@line@possep@\@ne
}%
```

`dimensions@calc@call` Callback for each column. When a non-blank cell is encountered, the available space on the left will be fixed if it is still undetermined, and the total width is updated to the current position: **TODO:** implement an offset for central alignment (global?!)

```
\def\eql@dimensions@calc@call{%
  \advance\eql@column@\@ne
  \ifnum\eql@totalcolumns@=\@ne
    \dimen@\eql@totalwidth@
  \else
    \dimen@\eql@colwidth@get\eql@column@\relax
  \fi
  \ifdim\eql@cellwidth@>\z@
    \ifdim\eql@line@width@=\z@
      \eql@line@avail@\eql@line@pos@
      \eql@line@availsep@\eql@line@possep@
      \ifcase\eql@shape@pos@
      \or
        \advance\eql@line@avail@\dimexpr
            (\dimen@-\eql@cellwidth@+\eql@centeroffset@)/\tw@\relax
      \or
        \advance\eql@line@avail@\dimexpr\dimen@-\eql@cellwidth@\relax
      \fi
      \advance\eql@line@avail@\eql@shape@amount@
    \fi
    \eql@line@width@\eql@line@pos@
    \eql@line@widthsep@\eql@line@possep@
    \ifcase\eql@shape@pos@
      \advance\eql@line@width@\eql@cellwidth@
    \or
      \advance\eql@line@width@\dimexpr
          (\dimen@+\eql@cellwidth@+\eql@centeroffset@)/\tw@\relax
    \or
      \advance\eql@line@width@\dimen@
    \fi
    \advance\eql@line@width@\eql@shape@amount@
  \fi
  \advance\eql@line@pos@\dimen@
}
```

9.4 Best Line Selection

`@numbering@best@auto` **TODO:** describe

```
\let\eql@numbering@best@auto\eql@false
```

`g@best@row@` (*counter*)
`g@best@space@` (*dimen*)
`ering@best@use` (*bool*)

```
\newcount\eql@numbering@best@row@
\newdimen\eql@numbering@best@space@
\let\eql@numbering@best@use\eql@false
```

`@numbering@best@find` Determine the row with the largest available space on the side of the tags:

```
\def\eql@numbering@best@find{%
  \eql@numbering@best@row@\z@
  \eql@numbering@best@space@\z@
  \eql@dimensions@for{%
    \eql@dimensions@calc
    \ifdefined\eql@tagsleft
      \dimen@\eql@line@avail@
    \else
      \dimen@\dimexpr\eql@totalwidth@-\eql@line@width@\relax
    \fi
    \ifdim\dimen@>\eql@numbering@best@space@
      \eql@numbering@best@row@\eql@row@
      \eql@numbering@best@space@\dimen@
    \fi
  }%
  \ifnum\eql@numbering@best@row@>\z@
    \eql@tagpos@row@\eql@numbering@best@row@
    \let\eql@tagpos@continuous\eql@false
    \eql@tagpos@prevrow@\z@
  \fi
}
```

`@numbering@best@test` **TODO:** describe

```
\def\eql@numbering@best@test#1{%
  \eql@dimensions@get#1%
  \eql@dimensions@calc
  \ifdefined\eql@tagsleft
    \dimen@\dimexpr\eql@line@avail@
        +\eql@marginleft@+\eql@line@availsep@\eql@colsep@\relax
  \else
    \dimen@\dimexpr\displaywidth-\eql@line@width@
        -\eql@marginleft@-\eql@line@widthsep@\eql@colsep@\relax
  \fi
  \ifdim\dimen@<\eql@tagwidth@block@
    \let\eql@numbering@best@use\eql@true
  \fi
}
```

`@numbering@best@eval` **TODO:** describe **TODO:** to test both lines individually may cause undesired effects

```
\def\eql@numbering@best@eval{%
  \ifdefined\eql@numbering@best@auto
    \ifdefined\eql@numbering@best@use\else
      \ifdefined\eql@numbering@multi\else
        \ifnum\eql@tagpos@row@>\z@
          \eql@numbering@best@test\eql@tagpos@row@
        \fi
        \ifnum\eql@tagpos@prevrow@>\z@
          \eql@numbering@best@test\eql@tagpos@prevrow@
        \fi
      \fi
    \fi
  \fi
  \ifdefined\eql@numbering@best@use
    \eql@numbering@best@find
  \fi
}
```

9.5 Tag Margin

TODO: describe **TODO:** if a tag margin is installed for a single line, it will shift the center even if there is no tag or importantly if a tag has been raised.

`djust@calc@tagmargin`

```
\def\eql@adjust@calc@tagmargin{%
  \ifdefined\eql@tagmargin@val
    \eql@tagmargin@\glueexpr\eql@tagmargin@val\relax
  \else
    \eql@tagmargin@\eql@tagwidth@max@
    \ifdim\eql@tagmargin@>\z@
      \advance\eql@tagmargin@-\eql@tagsepmin@
    \fi
  \fi

  \dimen@\eql@tagrows@\p@
  \ifnum\eql@totalrows@=\@ne
    \ifnum\eql@tagrows@=\@ne
      \advance\dimen@1sp\relax
    \fi
  \fi
  \ifdim\dimen@>\eql@totalrows@\eql@tagmargin@ratio@\else
    \eql@tagmargin@\z@
  \fi

  \@tempdima\dimexpr\displaywidth
      -\eql@totalwidth@-\eql@intercolumns@\eql@colsepmin@\relax
  \@tempdimb\dimexpr\@tempdima-\tw@\eql@tagmargin@\relax
  \ifdim\@tempdimb>\z@
    \ifdim\eql@tagmargin@threshold\@tempdima<\@tempdimb
      \eql@tagmargin@\z@
    \fi
  \fi
}
```

9.6 Single Column

ql@adjust@calc@lines

```
\def\eql@adjust@calc@lines{%
  \eql@totalcolumns@\@ne
  \eql@intercolumns@\z@
  \eql@colsep@\z@
  \ifdefined\eql@layoutleft
    \eql@marginleft@\glueexpr\eql@layoutleftmargin\relax
    \eql@marginleft@min@\glueexpr\eql@layoutleftmarginmin\relax
    \ifdim\eql@marginleft@<\eql@marginleft@min@
      \eql@marginleft@\eql@marginleft@min@
    \fi
    \dimen@\glueexpr\eql@layoutleftmarginmax\relax
    \ifdim\eql@marginleft@>\dimen@
      \eql@marginleft@\dimen@
    \fi
    \eql@marginright@\z@
    \eql@centeroffset@\z@
  \else
    \eql@adjust@calc@tagmargin
    \ifdefined\eql@paddingleft@val
      \eql@marginleft@\dimexpr
          (\displaywidth-\eql@totalwidth@-\eql@tagmargin@)/\tw@
          -\glueexpr\eql@paddingleft@val\relax\relax
      \ifdim\eql@marginleft@<\z@
        \eql@marginleft@\z@
      \fi
    \else
      \eql@marginleft@\z@
    \fi
    \ifdefined\eql@paddingright@val
      \eql@marginright@\dimexpr
          (\displaywidth-\eql@totalwidth@-\eql@tagmargin@)/\tw@
          -\glueexpr\eql@paddingright@val\relax\relax
      \ifdim\eql@marginright@<\z@
        \eql@marginright@\z@
      \fi
    \else
      \eql@marginright@\z@
    \fi
    \ifdim\eql@tagmargin@>\z@
      \ifdefined\eql@tagsleft
        \ifdim\eql@marginleft@<\eql@tagsepmin@
          \eql@marginleft@\eql@tagsepmin@
        \fi
        \advance\eql@marginleft@\eql@tagmargin@
        \advance\eql@centeroffset@\eql@tagmargin@
      \else
        \ifdim\eql@marginright@<\eql@tagsepmin@
          \eql@marginright@\eql@tagsepmin@
        \fi
        \advance\eql@marginright@\eql@tagmargin@
        \advance\eql@centeroffset@-\eql@tagmargin@
      \fi
    \fi
    \eql@marginleft@min@\z@
    \eql@centeroffset@\dimexpr\eql@marginright@-\eql@marginleft@
```

```
        \ifdefined\eql@tagsleft+\else-\fi\eql@tagmargin@\relax
  \fi
```

```
  \eql@totalwidth@\dimexpr\displaywidth
      -\eql@marginleft@-\eql@marginright@\relax
}
```

9.7 Multiple Columns

The following code computes the horizontal placement of columns. It distributes the columns evenly according to the layout presets and then determines whether there is enough space to place an equation tag on each line. If not, the intercolumn spacing and the space at the opposite margin can be reduced.

`@adjust@calc@columns` Main method to adjust column placement and spacing:

```
\def\eql@adjust@calc@columns{%
```

If there is just a single alignment stucture, there will be no intercolumn space that might stretch to adjust the columns to the margins. We disable fulllength to avoid a division by zero. Also guard against no columns at all (empty body), just in case:

```
  \ifnum\eql@totalcolumns@<\thr@@
    \eql@totalcolumns@\tw@
    \let\eql@columns@fulllength\eql@false
  \fi
```

Determine the number of intercolumn spaces `\eql@intercolumns@`:

```
  \eql@intercolumns@\numexpr(\eql@totalcolumns@-\tw@)/\tw@\relax
```

Evaluate the minimum intercolumn space which we will need often:

```
  \eql@colsepmin@\glueexpr\eql@colsepmin@val\relax
```

Determine the left or target margin width depending on the layout:

```
  \ifdefined\eql@layoutleft
    \eql@marginleft@\glueexpr\eql@layoutleftmargin\relax
    \eql@marginleft@min@\glueexpr\eql@layoutleftmarginmin\relax
    \ifdim\eql@marginleft@<\eql@marginleft@min@
      \eql@marginleft@\eql@marginleft@min@
    \fi
  \else
```

Get the desired tag margin, increase by minimum tag separation if columns are aligned to the margins. Cancel tag margin if too wide:

```
    \eql@adjust@calc@tagmargin
    \ifdefined\eql@columns@fulllength
      \ifdim\eql@tagmargin@>\z@
        \advance\eql@tagmargin@\eql@tagsepmin@
      \fi
    \fi
    \ifdim\eql@tagmargin@>\dimexpr\displaywidth-\eql@totalwidth@
       -\eql@intercolumns@\eql@colsepmin@\relax
      \eql@tagmargin@\z@
    \fi
    \eql@marginleft@min@\z@
  \fi
```

Compute the intercolumn space `\eql@colsep@`:

```
\ifnum\eql@intercolumns@>\z@
```

Distribute the available horizontal space evenly onto the intercolumn spaces and the margins. Unless the columns are aligned to the margins, there are two margins in central alignment layout but only the right margin in left alignment layout:

```
  \eql@colsep@\dimexpr\displaywidth-\eql@totalwidth@\relax
  \ifdefined\eql@layoutleft
    \advance\eql@colsep@-\eql@marginleft@
  \else
    \advance\eql@colsep@-\eql@tagmargin@
  \fi
  \count@\eql@intercolumns@
  \ifdefined\eql@columns@fulllength\else
    \ifdefined\eql@layoutleft
      \advance\count@\@ne
    \else
      \advance\count@\tw@
    \fi
  \fi
  \divide\eql@colsep@\count@
```

Ensure that the intercolumn separation is within the specified bounds. Disable the upper bound if columns are to be aligned to the margins:

```
  \ifdim\eql@colsep@<\eql@colsepmin@
    \eql@colsep@\eql@colsepmin@
  \else
    \ifdefined\eql@columns@fulllength\else
      \dimen@\glueexpr\eql@colsepmax@val\relax
      \ifdim\eql@colsep@>\dimen@
        \eql@colsep@\dimen@
      \fi
    \fi
  \fi
\else
```

For a single column, set the column separation to the minimum amount:

```
  \eql@colsep@\eql@colsepmin@
\fi
```

Compute the left margin `\eql@marginleft@` depending on the layout:

```
\ifdefined\eql@layoutleft
```

Set the default value:

```
  \ifdim\eql@colsep@=\eql@colsepmin@
```

If in left alignment layout the intercolumn space has been adjusted, compute the available space, determine left margin and make sure it is between the minimum and the default value:

```
    \dimen@\dimexpr\displaywidth-\eql@totalwidth@
        -\eql@intercolumns@\eql@colsep@\relax
    \ifdim\dimen@<\eql@marginleft@
      \ifdim\dimen@<\eql@marginleft@min@
        \eql@marginleft@\eql@marginleft@min@
      \else
```

```
        \eql@marginleft@\dimen@
      \fi
    \fi
  \fi
\else
```

In central alignment mode with column aligned to the margins, set margin to zero:

```
  \ifdefined\eql@columns@fulllength
    \eql@marginleft@\z@
```

In central alignment mode with margins, distribute the available space equally to both margins, or remove the left margin if insufficient:

```
  \else
    \eql@marginleft@\dimexpr(\displaywidth-\eql@totalwidth@
        -\eql@intercolumns@\eql@colsep@-\eql@tagmargin@)/\tw@\relax
    \ifdim\eql@marginleft@<\z@
      \eql@marginleft@\z@
    \fi
  \fi
```

Add tag margin in case of left tags:

```
  \ifdefined\eql@tagsleft
    \advance\eql@marginleft@\eql@tagmargin@
  \fi
\fi
```

Find the best row for tag placement:

```
\eql@numbering@best@eval
```

Next consider all rows with tags and adjust the intercolumn and margin space to make the tags fit into the available space at the corresponding side as far as possible. First, select code depending on tag placement:

```
\ifdefined\eql@tagsleft
  \let\eql@adjust@columns@test\eql@adjust@columns@test@tagsleft
\else
  \let\eql@adjust@columns@test\eql@adjust@columns@test@tagsright
\fi
```

Loop over all rows or select the single row containing the tag. Fetch the width data for the current row. If a tag is present, compute the available space and try to adjust spaces if needed: **TODO:** complete for prevrow, ideally join treatment

```
\ifdefined\eql@numbering@multi
  \eql@dimensions@for{%
    \ifdim\eql@tagwidth@>\z@
      \eql@dimensions@calc
      \eql@adjust@columns@test
    \fi
  }%
\else
  \ifnum\eql@tagpos@row@>\z@
    \ifnum\eql@tagpos@row@>\eql@totalrows@\else
      \eql@dimensions@get\eql@tagpos@row@
      \eql@tagwidth@\eql@tagwidth@block@
      \eql@dimensions@calc
      \eql@adjust@columns@test
    \fi
  \fi
  \ifnum\eql@tagpos@prevrow@>\z@
    \eql@dimensions@get\eql@tagpos@prevrow@
    \eql@tagwidth@\eql@tagwidth@block@
    \eql@dimensions@calc
    \eql@adjust@columns@test
  \fi
\fi
```

From now on `\eql@totalwidth@` will include the left margin and the total intercolumn separation:

```
  \advance\eql@totalwidth@\dimexpr
      \eql@intercolumns@\eql@colsep@+\eql@marginleft@\relax
}
```

Placement for Right Tags.

`lumns@test@tagsright` Test whether the spacing can be adjusted to make the current row fit:

```
\def\eql@adjust@columns@test@tagsright{%
```

The register `\@tempdima` will hold the amount of available space. **TODO:** does this apply equally to left alignment layout?

```
  \@tempdima\dimexpr\displaywidth-\eql@line@width@-\eql@tagwidth@\relax
```

Test whether the space at the end of the row is sufficient to hold the tag with the current settings.

```
  \ifdim\@tempdima<\dimexpr
      \eql@marginleft@+\eql@line@widthsep@\eql@colsep@\relax
```

If not, determine whether the row and tag may at all fit into the available space with minimal intercolumn spaces and minimal left margin (in left alignment layout).

```
    \ifdim\@tempdima<\dimexpr
        \eql@marginleft@min@+\eql@line@widthsep@\eql@colsepmin@\relax\else
```

If so, hand over to `\eql@adjust@columns@modify@tagsright`.

```
      \eql@adjust@columns@modify@tagsright
    \fi
  \fi
}
```

`mns@modify@tagsright` Adjust the intercolumn space and left margin to make the row fit.

```
\def\eql@adjust@columns@modify@tagsright{%
```

If there are any intercolumn spaces that contribute to the available space, determine how much intercolumn separation would be needed while keeping the current left margin fixed (in left alignment layout). In central alignment layout, assume that the left margin will be adjusted to match the intercolumn separation by stepping the number of columns to divide by.

```
  \ifnum\eql@line@widthsep@>\z@
    \dimen@\@tempdima
    \count@\eql@line@widthsep@
    \ifdefined\eql@layoutleft
      \advance\dimen@-\eql@marginleft@
    \else
      \ifdefined\eql@columns@fulllength\else
        \advance\count@\@ne
      \fi
    \fi
    \divide\dimen@\count@
```

If smaller, reduce the intercolumn separation, but make sure to not exceed the minumum allowed value.

```
    \ifdim\dimen@<\eql@colsep@
      \ifdim\dimen@<\eql@colsepmin@
        \eql@colsep@\eql@colsepmin@
      \else
        \eql@colsep@\dimen@
      \fi
    \fi
  \fi
```

Now adjust the left margin as much as needed to fit the contents.

```
  \dimen@\dimexpr\@tempdima-\eql@line@widthsep@\eql@colsep@\relax
  \ifdim\eql@marginleft@>\dimen@
    \eql@marginleft@\dimen@
  \fi
}
```

Placement for Left Tags.

`olumns@test@tagsleft` Test whether the spacing can be adjusted to make the current row fit:

```
\def\eql@adjust@columns@test@tagsleft{%
```

The register `\@tempdima` will hold the deficit amount of space at the beginning of the row without adjustable space, and the register `\count@` will hold the number of intercolumn spaces that would contribute to space adjustments.

```
  \count@\numexpr\eql@intercolumns@-\eql@line@availsep@\relax
  \@tempdima\dimexpr\eql@tagwidth@-\eql@line@avail@\relax
```

Test whether the space at the beginning of the row is sufficient to hold the tag with the current settings.

```
  \ifdim\@tempdima>\dimexpr
      \eql@marginleft@+\eql@line@availsep@\eql@colsep@\relax
```

If not, first verify that the tag will fit the line (or the maxumal left margin in left alignment layout).

```
    \ifdim\eql@tagwidth@<%
        \ifdefined\eql@layoutleft
          \glueexpr\eql@layoutleftmarginmax\relax
        \else
          \displaywidth
        \fi
```

If so, determine whether the row and tag may at all fit into the available space with minimal intercolumn spaces.

```
      \ifdim\@tempdima>\dimexpr
          \displaywidth-\eql@totalwidth@-\count@\eql@colsepmin@\relax\else
```

If so, hand over to `\eql@adjust@columns@modify@tagsleft`.

```
        \eql@adjust@columns@modify@tagsleft
      \fi
    \fi
  \fi
}
```

`umns@modify@tagsleft` Adjust the intercolumn space and left margin to make the row fit.

```
\def\eql@adjust@columns@modify@tagsleft{%
```

If there are any intercolumn spaces that contribute to the available space, determine how much intercolumn separation would be needed while keeping the current right margin fixed. In central alignment layout, assume that the right margin will be adjusted to match the intercolumn separation by stepping the number of columns to divide by.

```
  \ifnum\count@>\z@
    \dimen@\dimexpr\displaywidth-\eql@totalwidth@-\@tempdima\relax
    \ifdefined\eql@columns@fulllength\else
      \advance\count@\@ne
    \fi
    \divide\dimen@\count@
```

If smaller, reduce the intercolumn separation, but make sure to not exceed the minumum allowed value. Also adjust the left margin to keep the right margin fixed.

```
    \ifdim\dimen@<\eql@colsep@
      \ifdim\dimen@<\eql@colsepmin@
        \dimen@\eql@colsepmin@
      \fi
      \advance\dimen@-\eql@colsep@
      \advance\eql@marginleft@-\eql@intercolumns@\dimen@
      \advance\eql@colsep@\dimen@
    \fi
  \fi
```

Now adjust the left margin as much as needed to fit the contents.

```
  \dimen@\dimexpr\@tempdima-\eql@line@availsep@\eql@colsep@\relax
  \ifdim\eql@marginleft@<\dimen@
    \eql@marginleft@\dimen@
  \fi
}
```

10 Single Column Arrangement

The following code adjusts individual lines of equations for the equation and lines mode according to the selected layout and shape.

10.1 Supporting Definitions

`\inf@bad` The `\inf@bad` constant is for testing overfull boxes:

```
\ifdefined\inf@bad\else%
  \newcount\inf@bad
  \inf@bad1000000\relax
\fi
```

`\eql@restore@hfuzz`
`\eql@save@hfuzz`

We need to change the value of `\hfuzz` temporarily. The method `\eql@save@hfuzz` stores the value for recovery through `\eql@restore@hfuzz`:

```
\let\eql@restore@hfuzz\@empty
\def\eql@save@hfuzz{\edef\eql@restore@hfuzz{\hfuzz\the\hfuzz\relax}}
```

`\eql@alignbadness@`
`\eql@tagbadness@`

The registers `\eql@alignbadness@` and `\eql@tagbadness@` store the allowable badness threshold for shrinking equation lines to the intended margin or to fit into the line at all before the tag is raised or lowered:

```
\newcount\eql@alignbadness@
\newcount\eql@tagbadness@
\newcount\eql@arrange@badness@
\eql@alignbadness@\inf@bad
\eql@tagbadness@\inf@bad
```

10.2 Arrangement Methods

`\eql@arrange@try`

Try to fit the current equation line in the available space. Argument `#1` specifies the amount of reserved space. Unpack the box `\eql@cellbox@`, replace the previous kerning with the new reserved space, and save the box back into `\eql@cellbox@`:

```
\def\eql@arrange@try#1{%
  \ifdim#1>\dimexpr\displaywidth-\eql@cellwidth@\relax
    \setbox\eql@cellbox@\hbox to\displaywidth{%
      \unhbox\eql@cellbox@\unkern\kern#1}%
    \eql@arrange@badness@\badness
  \else
    \eql@arrange@badness@\m@ne
  \fi
}
```

`\eql@arrange@print`

We have found the final adjustment of the current line, so we typeset it with initial and final space adjustments `#1` and `#2`, respectively. Restore the original value for `\hfuzz`: **TODO:** adjust

```
\def\eql@arrange@print#1#2{%
  \eql@restore@hfuzz
  \if@eqnsw
    \ifdefined\eql@tagsleft
      \eql@tagbox@print@tagsleft
    \fi
  \fi
  \hbox to\displaywidth{%
    #1%
    \unhbox\eql@cellbox@\unkern
    #2%
    \eql@tagging@mathaddlast
  }%
  \if@eqnsw
    \ifdefined\eql@tagsleft\else
      \eql@tagbox@print@tagsright
    \fi
  \fi
}
```

`ange@print@alignleft`
`ge@print@aligncenter`
`nge@print@alignright`

Fit the current equation line with the selected alignment within a given left and right margins `#1` and `#2`. If we're on the first line, adjust `\eql@display@firstavail@` to the mininum left available space we can guarantee:

```
\def\eql@arrange@print@alignleft#1#2{%
  \eql@display@firstavail@set{\dimexpr#1\relax}%
  \eql@arrange@print{\kern#1}{\kern#2}%
}
```

```
\def\eql@arrange@print@alignright#1#2{%
  \eql@display@firstavail@set{\dimexpr\displaywidth-\eql@cellwidth@-#2\relax}%
  \eql@arrange@print{\kern#1\hfil}{\unskip\kern#2}%
}
```

```
\def\eql@arrange@print@aligncenter#1{%
  \eql@display@firstavail@set{\dimexpr
      (\displaywidth-\eql@cellwidth@+#1)/\tw@\relax}%
  \ifdim#1>\z@
    \eql@arrange@print{\kern#1\hfil}{}%
  \else
    \eql@arrange@print{\hfil}{\kern-#1}%
  \fi
}
```

`\eql@arrange@init` Initialise the horizontal adjustment framework. Turn off overfull box messages temporarily – otherwise there would be unwanted extra ones emitted during our measuring operations. Select the shape scheme:

```
\def\eql@arrange@init{%
  \eql@save@hfuzz
  \hfuzz\maxdimen
  \eql@shape@select
}
```

`l@arrange@print@line` Select the appropriate adjustment method depending on the current alignment position, the selected tag placement if any: **TODO:** adjust

```
\def\eql@arrange@print@line{%
  \eql@tagging@tagaddbox
  \csname eql@arrange%
    @\ifcase\eql@shape@pos@ alignleft\or aligncenter\or alignright\fi
    @init\endcsname
  \csname eql@arrange%
    @\ifcase\eql@shape@pos@ alignleft\or aligncenter\or alignright\fi
    @\ifdefined\eql@tagpos@reserve
      \ifdefined\eql@tagsleft tagsleft\else tagsright\fi\else
      notag\fi\endcsname
}
```

10.3 Central Alignment

TODO: describe

```
\def\eql@arrange@aligncenter@init{%
  \eql@tagging@aligncenter
  \eql@line@offset@\dimexpr\tw@\eql@shape@amount@
      +\eql@marginleft@-\eql@marginright@+\eql@centeroffset@\relax
}
```

TODO: describe

```
\def\eql@arrange@aligncenter@notag{%
 \ifdim\dimexpr\displaywidth-\eql@cellwidth@\relax>%
   \ifdim\eql@line@offset@<\eql@marginleft@min@
    \dimexpr\tw@\eql@marginleft@min@-\eql@line@offset@\relax
   \else
    \eql@line@offset@
   \fi
  \eql@arrange@print@aligncenter\eql@line@offset@
 \else
  \ifdim\eql@line@offset@<\eql@marginleft@min@
   \eql@arrange@print@alignleft\eql@marginleft@min@\z@
  \else
   \eql@arrange@print@alignright\eql@marginleft@min@\z@
  \fi
 \fi
}
```

TODO: describe

```
\def\eql@arrange@aligncenter@tagsright{%
 \ifdim\dimexpr\displaywidth-\eql@cellwidth@\relax>%
   \ifdim\eql@line@offset@<\dimexpr\eql@marginleft@min@-\eql@tagwidth@\relax
    \dimexpr\tw@\eql@marginleft@min@-\eql@line@offset@\relax
   \else
    \dimexpr\tw@\eql@tagwidth@+\eql@line@offset@\relax
   \fi
  \eql@arrange@print@aligncenter\eql@line@offset@
 \else
  \eql@arrange@try{\dimexpr\eql@tagwidth@+\eql@marginleft@min@\relax}%
  \ifnum\eql@arrange@badness@<\eql@tagbadness@
   \ifdim\eql@line@offset@<\dimexpr\eql@marginleft@min@-\eql@tagwidth@\relax
    \eql@arrange@print@alignleft\eql@marginleft@min@\eql@tagwidth@
   \else
    \eql@arrange@print@alignright\eql@marginleft@min@\eql@tagwidth@
   \fi
  \else
   \let\eql@tagpos@reserve\eql@false
   \eql@arrange@aligncenter@notag
  \fi
 \fi
}
```

```
\def\eql@arrange@aligncenter@tagsleft{%
 \ifdim\eql@tagwidth@>\eql@marginleft@min@
  \ifdim\dimexpr\displaywidth-\eql@cellwidth@\relax>%
    \ifdim\eql@line@offset@<\eql@tagwidth@
     \dimexpr\tw@\eql@tagwidth@-\eql@line@offset@\relax
    \else
     \eql@line@offset@
    \fi
   \eql@arrange@print@aligncenter\eql@line@offset@
  \else
   \eql@arrange@try\eql@tagwidth@
   \ifnum\eql@arrange@badness@<\eql@tagbadness@
    \ifdim\eql@line@offset@<\eql@tagwidth@
     \eql@arrange@print@alignleft\eql@tagwidth@\z@
    \else
     \eql@arrange@print@alignright\eql@tagwidth@\z@
```

```
        \fi
      \else
        \let\eql@tagpos@reserve\eql@false
        \eql@arrange@aligncenter@notag
      \fi
    \fi
  \else
    \eql@arrange@aligncenter@notag
  \fi
}
```

10.4 Left Alignment

```
\def\eql@arrange@alignleft@init{%
  \eql@tagging@alignleft
  \eql@line@offset@\dimexpr\eql@marginleft@+\eql@shape@amount@\relax
  \ifdim\eql@line@offset@<\eql@marginleft@min@
    \eql@line@offset@\eql@marginleft@min@
  \fi
}
```

```
\def\eql@arrange@alignleft@notag{%
  \ifdim\eql@line@offset@>\eql@marginleft@min@
    \eql@arrange@try\eql@line@offset@
    \ifnum\eql@arrange@badness@<\eql@alignbadness@
      \eql@arrange@print@alignleft\eql@line@offset@\z@
    \else
      \eql@arrange@print@alignright\eql@marginleft@min@\z@
    \fi
  \else
    \eql@arrange@print@alignleft\eql@marginleft@min@\z@
  \fi
}
```

```
\def\eql@arrange@alignleft@tagsright{%
  \eql@arrange@try{\dimexpr\eql@line@offset@+\eql@tagwidth@\relax}%
  \ifnum\eql@arrange@badness@<\eql@alignbadness@
    \eql@arrange@print@alignleft\eql@line@offset@\eql@tagwidth@
  \else
    \ifdim\eql@line@offset@>\eql@marginleft@min@
      \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@tagwidth@\relax}%
    \fi
    \ifnum\eql@arrange@badness@<\eql@tagbadness@
      \eql@arrange@print@alignright\eql@marginleft@min@\eql@tagwidth@
    \else
      \let\eql@tagpos@reserve\eql@false
      \eql@arrange@alignleft@notag
    \fi
  \fi
}
```

```
\def\eql@arrange@alignleft@tagsleft{%
  \ifdim\eql@tagwidth@>\eql@marginleft@min@
    \ifdim\eql@line@offset@>\eql@tagwidth@
      \eql@arrange@try\eql@line@offset@
      \ifnum\eql@arrange@badness@<\eql@alignbadness@
        \eql@arrange@print@alignleft\eql@line@offset@\z@
      \else
        \eql@arrange@try\eql@tagwidth@
        \ifnum\eql@arrange@badness@<\eql@tagbadness@
```

```
          \eql@arrange@print@alignright\eql@tagwidth@\z@
        \else
          \let\eql@tagpos@reserve\eql@false
          \eql@arrange@print@alignright\eql@marginleft@min@\z@
        \fi
      \fi
    \else
      \eql@arrange@try\eql@tagwidth@
      \ifnum\eql@arrange@badness@<\eql@tagbadness@
        \eql@arrange@print@alignleft\eql@tagwidth@\z@
      \else
        \let\eql@tagpos@reserve\eql@false
        \eql@arrange@alignleft@notag
      \fi
    \fi
  \else
    \eql@arrange@alignleft@notag
  \fi
}
```

10.5 Right Alignment

```
\def\eql@arrange@alignright@init{%
  \eql@tagging@alignright
  \eql@line@offset@\dimexpr\eql@marginright@-\eql@shape@amount@\relax
  \ifdim\eql@line@offset@<\z@
    \eql@line@offset@\z@
  \fi
}
```

TODO: describe

```
\def\eql@arrange@alignright@notag{%
  \ifdim\eql@line@offset@>\z@
    \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@line@offset@\relax}%
    \ifnum\eql@arrange@badness@<\eql@alignbadness@
      \eql@arrange@print@alignright\eql@marginleft@min@\eql@line@offset@
    \else
      \eql@arrange@print@alignleft\eql@marginleft@min@\z@
    \fi
  \else
    \eql@arrange@print@alignright\eql@marginleft@min@\z@
  \fi
}
```

TODO: describe

```
\def\eql@arrange@alignright@tagsright{%
  \ifdim\eql@line@offset@>\eql@tagwidth@
    \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@line@offset@\relax}%
    \ifnum\eql@arrange@badness@<\eql@alignbadness@
      \eql@arrange@print@alignright\eql@marginleft@min@\eql@line@offset@
    \else
      \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@tagwidth@\relax}%
      \ifnum\eql@arrange@badness@<\eql@tagbadness@
        \eql@arrange@print@alignleft\eql@marginleft@min@\eql@tagwidth@
      \else
        \let\eql@tagpos@reserve\eql@false
        \eql@arrange@print@alignleft\eql@marginleft@min@\z@
      \fi
    \fi
  \else
    \eql@arrange@try{\dimexpr\eql@marginleft@min@+\eql@tagwidth@\relax}%
    \ifnum\eql@arrange@badness@<\eql@tagbadness@
      \eql@arrange@print@alignright\eql@marginleft@min@\eql@tagwidth@
    \else
      \let\eql@tagpos@reserve\eql@false
      \eql@arrange@alignright@notag
    \fi
  \fi
}
```

TODO: describe

```
\def\eql@arrange@alignright@tagsleft{%
  \ifdim\eql@tagwidth@>\eql@marginleft@min@
    \eql@arrange@try{\dimexpr\eql@line@offset@+\eql@tagwidth@\relax}%
    \ifnum\eql@arrange@badness@<\eql@alignbadness@
      \eql@arrange@print@alignright\eql@tagwidth@\eql@line@offset@
    \else
      \ifdim\eql@line@offset@>\z@
        \eql@arrange@try\eql@tagwidth@
      \fi
      \ifnum\eql@arrange@badness@<\eql@tagbadness@
        \eql@arrange@print@alignleft\eql@tagwidth@\z@
      \else
        \let\eql@tagpos@reserve\eql@false
        \eql@arrange@alignright@notag
      \fi
    \fi
  \else
    \eql@arrange@alignright@notag
  \fi
}
```

11 Equations Box Environment

TODO: outline sequence of calls

TODO: describe

TODO: fixed width version (works only towards intercolumn stretch)?

TODO: vspace?!

11.1 Line Breaks

TODO: describe

`\eql@box@cr`

```
\def\eql@box@cr{%
  \ifmmode\else\unskip\fi
  \eql@vspaceskip@\z@skip
  \let\eql@punct@term\eql@false
  \let\eql@class@rel@composed\@empty
  \eql@ampprotect\eql@box@cr@test\eql@box@cr@process
}
```

TODO: describe

```
\def\eql@box@cr@test@setopt{\let\eql@box@cr@test\eql@box@cr@testopt}
\def\eql@box@cr@test@setall{\let\eql@box@cr@test\eql@box@cr@testall}
```

`\eql@box@cr@testopt` **TODO:** describe

```
\def\eql@box@cr@testopt#1{\eql@teststaropt@tight
  {\eql@box@cr@testopt@set{#1}}{\eql@box@cr@testopt@set{#1}}{0pt}}
\def\eql@box@cr@testopt@set#1[#2]{\advance\eql@vspaceskip@\glueexpr#2\relax#1}
```

`\eql@box@cr@testall` **TODO:** describe

`box@cr@testall@parse`

```
\def\eql@box@cr@testall{\eql@parseopt@cr\eql@box@cr@testall@parse}
\def\eql@box@cr@testall@parse{%
  \ifx\eql@parseopt@token[%
    \let\eql@parseopt@next\eql@parseopt@vspace
  \fi
  \ifx\eql@parseopt@token*%
    \let\eql@parseopt@next\eql@parseopt@gobble
  \fi
  \ifx\eql@parseopt@token.%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token,%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token~%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token`%
    \let\eql@parseopt@next\eql@parseopt@punctnext
  \fi
  \ifx\eql@parseopt@token!%
    \let\eql@parseopt@next\eql@parseopt@puncterm
  \fi
  \ifx\eql@parseopt@token/%
    \let\eql@parseopt@next\eql@parseopt@punctclear
  \fi
  \ifx\eql@parseopt@token=%
    \let\eql@parseopt@next\eql@parseopt@relsymb
  \fi
  \ifx\eql@parseopt@token;%
    \let\eql@parseopt@next\eql@parseopt@relcont
  \fi
  \ifx\eql@parseopt@token:%
    \let\eql@parseopt@next\eql@parseopt@relstart
  \fi
  \ifx\eql@parseopt@token|%
    \let\eql@parseopt@next\eql@parseopt@relord
  \fi
  \ifx\eql@parseopt@token?%
    \let\eql@parseopt@next\eql@box@cr@parse@rel
  \fi
  \ifx\eql@parseopt@token&%
    \let\eql@parseopt@next\eql@parseopt@end
  \fi
}
\def\eql@box@cr@parse@rel#1#2{%
  \def\eql@tmp{#2}%
  \ifx\eql@tmp\eql@relax\else
```

```
    \ifdefined\eql@punct@next\else
      \eql@punct@next@clear
    \fi
  \fi
  \ifx\eql@box@open\eql@box@lines@open
    \ifx\eql@tmp\@empty
      \def\eql@class@rel@composed{\eql@shape@cont}%
    \else
      \def\eql@class@rel@composed{\eql@shape@rel#2}%
    \fi
  \fi
  \ifx\eql@box@open\eql@box@columns@open
    \def\eql@class@rel@composed{\eql@class@rel@make{#2}}%
  \fi
  \ifx\eql@box@open\eql@box@cases@open
    \def\eql@class@rel@composed{&#2}%
  \fi
  \eql@parseopt@end}
```

`\eql@box@cr@process`

```
\def\eql@box@cr@process{%
  \ifdefined\eql@punct@term\eql@punct@apply@top\fi
  \edef\eql@tmp{%
    \unexpanded{%
      \eql@box@endline
      \eql@box@lastcell
      \cr
    }%
    \noalign{%
      \vskip\the\eql@vspaceskip@\relax
    }%
    \unexpanded\expandafter{\eql@class@rel@composed}%
  }%
  \eql@tmp
}
```

`\eql@box@endline`

```
\def\eql@box@endline{%
  \eql@punct@apply@line
  \eql@hook@lineout
}
```

11.2 Column Breaks

TODO: describe

```
\def\eql@box@amp{%
  \eql@ampprotecttwo\eql@box@amp@testescape\eql@amp@org
    \eql@box@amp@process}
\def\eql@box@amp@testescape#1#2{\eql@ifnextgobble@tight/{#1}{%
  \relax
  \let\eql@punct@term\eql@false
  \let\eql@class@rel@composed\@empty
  \eql@box@amp@test{#2}}}
```

TODO: describe

```
\def\eql@box@amp@test@setopt{%
  \let\eql@box@amp@test\eql@box@amp@testopt}
\def\eql@box@amp@test@setall{%
  \let\eql@box@amp@test\eql@box@amp@testall}
```

\eql@box@amp@testopt **TODO:** describe

```
\let\eql@box@amp@testopt\@empty
```

\eql@box@amp@testall
ox@amp@testall@parse **TODO:** describe

```
\def\eql@box@amp@testall{\eql@parseopt@cr\eql@box@amp@testall@parse}
\def\eql@box@amp@testall@parse{%
  \ifx\eql@parseopt@token.%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token,%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token~%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token`%
    \let\eql@parseopt@next\eql@parseopt@punctnext
  \fi
  \ifx\eql@parseopt@token!%
    \let\eql@parseopt@next\eql@parseopt@puncterm
  \fi
  \ifx\eql@parseopt@token=%
    \let\eql@parseopt@next\eql@parseopt@relsymb
  \fi
  \ifx\eql@parseopt@token;%
    \let\eql@parseopt@next\eql@parseopt@relcont
  \fi
  \ifx\eql@parseopt@token:%
    \let\eql@parseopt@next\eql@parseopt@relstart
  \fi
  \ifx\eql@parseopt@token|%
    \let\eql@parseopt@next\eql@parseopt@relord
  \fi
  \ifx\eql@parseopt@token?%
    \let\eql@parseopt@next\eql@box@amp@parse@rel
  \fi
  \ifx\eql@parseopt@token&%
    \let\eql@parseopt@next\eql@parseopt@end
  \fi
}
\def\eql@box@amp@parse@rel#1#2{%
  \ifx\eql@box@open\eql@box@columns@open
    \def\eql@class@rel@composed{\eql@class@rel@make{#2}}%
  \else
    \def\eql@class@rel@composed{#2}%
  \fi
  \eql@parseopt@end}
```

\eql@box@amp@process

```
\def\eql@box@amp@process{%
  \ifdefined\eql@punct@term\eql@punct@apply@top\fi
  \ifx\eql@box@open\eql@box@columns@open
    \edef\eql@tmp{%
      \ifx\eql@class@rel@composed\@empty
        \ifx\eql@box@lastcell\eql@box@columns@lastcell@odd
          &\noexpand\eql@punct@next@clear\fi&%
      \else
        \ifx\eql@box@lastcell\eql@box@columns@lastcell@even&\fi%
        \unexpanded\expandafter{\eql@class@rel@composed}%
      \fi
    }%
  \else
    \edef\eql@tmp{%
      \ifx\eql@class@rel@composed\@empty
        &%
      \else
        \unexpanded\expandafter{\eql@class@rel@composed}%
      \fi
    }%
  \fi
  \eql@tmp
}
```

11.3 Lines Mode

```
\def\eql@box@lines@lastcell{&\omit\kern-2\eql@colsep@}
```

TODO: templates

```
\def\eql@box@lines@open{%
  \eql@shape@align@enable
  \let\eql@box@lastcell\eql@box@lines@lastcell
  \everycr{\noalign{%
    \eql@verbose@info\eql@verbose@msg@startline@number
    \global\advance\eql@row@\@ne
  }}%
  \tabskip\z@skip
  \halign\bgroup
    &%
      \global\let\eql@cell@container\@empty
      \setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \ifdefined\eql@shape@lastrow
        \eql@totalrows@\eql@row@
      \fi
      \eql@shape@eval
      \eql@cell@container
      \ifdefined\eql@frame@cmd
        \ifcase\eql@shape@pos@
          \eql@frame@measure
          \advance\eql@shape@amount@-\eql@frame@margin@
        \or\or
          \eql@frame@measure
          \advance\eql@shape@amount@+\eql@frame@margin@
        \fi
        \eql@frame@print
      \fi
      \ifcase\eql@shape@pos@
        \kern\eql@shape@amount@
        \box\eql@cellbox@
        \hskip\glueexpr\eql@paddingleft@+\eql@paddingright@
          -\eql@shape@amount@+\@flushglue\relax
        \eql@tagging@alignleft
      \or
        \hskip\glueexpr\eql@paddingleft@+\eql@shape@amount@+\@flushglue\relax
        \box\eql@cellbox@
        \hskip\glueexpr\eql@paddingright@-\eql@shape@amount@+\@flushglue\relax
        \eql@tagging@aligncenter
      \or
        \hskip\glueexpr\eql@paddingleft@+\eql@paddingright@
          +\eql@shape@amount@+\@flushglue\relax
        \box\eql@cellbox@
        \kern-\eql@shape@amount@
        \eql@tagging@alignright
      \fi
      \tabskip\eql@colsep@\relax
    \crcr
    \noalign{%
      \global\let\eql@shape@lastrow\eql@false
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
}
```

```
\def\eql@box@lines@set{\let\eql@box@open\eql@box@lines@open}
```

11.4 Columns Mode

```
\def\eql@box@columns@lastcell@odd{%
  &\omit
  \eql@prevwidth@\wd\eql@cellbox@
  \let\eql@frame@cmd\eql@frame@prevcmd
  \ifdefined\eql@frame@cmd
    \eql@frame@measure
    \advance\eql@prevwidth@\eql@frame@margin@
    \eql@frame@print
  \fi
  \kern-\eql@prevwidth@
  \unhbox\eql@cellbox@
  \hfil
  &\omit\kern-\eql@colsep@
}%
\def\eql@box@columns@lastcell@even{&\omit\kern-\eql@colsep@}
```

```
\def\eql@box@columns@open{%
% \TODO templates
  \eql@shape@align@disable
  \let\eql@box@lastcell\@empty
  \everycr{\noalign{%
    \eql@verbose@info\eql@verbose@msg@startline@new
  }}%
  \tabskip\z@skip
  \halign\bgroup
    &%
      \let\eql@box@lastcell\eql@box@columns@lastcell@odd
      \global\let\eql@cell@container\@empty
      \global\setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@next
          \eql@class@innerleft
          \eql@hook@innerleft
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \eql@cell@container
      \hfil
      \kern\wd\eql@cellbox@
      \ifdefined\eql@frame@cmd
        \eql@frame@measure
        \kern\eql@frame@margin@
      \fi
      \global\let\eql@frame@prevcmd\eql@frame@cmd
      \tabskip\z@skip
    &%
      \eql@prevwidth@\wd\eql@cellbox@
      \let\eql@box@lastcell\eql@box@columns@lastcell@even
      \let\eql@frame@cmd\eql@frame@prevcmd
      \global\let\eql@cell@container\@empty
      \setbox\eql@cellbox@\hbox{%
        \unhbox\eql@cellbox@
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \eql@hook@innerright
          \eql@class@innerright@sel
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \eql@cell@container
      \ifdefined\eql@frame@cmd
        \eql@frame@measure
        \advance\eql@prevwidth@\eql@frame@margin@
        \eql@frame@print
      \fi
      \kern-\eql@prevwidth@
      \unhbox\eql@cellbox@
      \hfil
      \tabskip\eql@colsep@\relax
    \crcr
```

```
    \noalign{%
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
}
```

```
\def\eql@box@columns@set{\let\eql@box@open\eql@box@columns@open}
```

11.5 Cases Mode

TODO: describe

TODO: how to get proper height in tagging (and avoid nulldelimiterspace) **TODO:** add alignment?

```
\def\eql@box@cases@lastcell{&}%
```

```
\let\eql@box@cases@condtext\eql@false
\let\eql@box@cases@condintro\@empty
```

```
\def\eql@box@cases@open{%
  \eql@shape@align@disable
  \let\eql@box@lastcell\@empty
  \everycr{\noalign{%
    \eql@verbose@info\eql@verbose@msg@startline@new
  }}%
  \tabskip\z@skip
  \halign\bgroup
      \let\eql@box@lastcell\eql@box@cases@lastcell
      \global\let\eql@cell@container\@empty
      \global\setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@col
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \eql@cell@container
      \unhbox\eql@cellbox@
      \hfil
      \eql@tagging@alignleft
      \tabskip\eql@colsep@\relax
    &%
      \let\eql@box@lastcell\@empty
      \global\let\eql@cell@container\@empty
      \setbox\eql@cellbox@\hbox{%
        \unhbox\eql@cellbox@
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \ifdefined\eql@box@cases@condtext
            \expandafter\hbox\else\expandafter\@firstofone\fi\bgroup
          \eql@box@cases@condintro
          ##%
          \eql@punct@apply@col
          \egroup
          \eql@hook@colout
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \eql@cell@container
      \unhbox\eql@cellbox@
      \hfil
      \eql@tagging@alignleft
      \tabskip\z@skip
    \crcr
    \noalign{%
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
}
```

```
\def\eql@box@cases@set{%
  \ifdefined\eql@cases@mathstyle\let\eql@mathstyle\eql@cases@mathstyle\fi
  \ifdefined\eql@punct@cases\let\eql@punct@col\eql@punct@cases\fi
  \let\eql@box@open\eql@box@cases@open}
```

11.6 Main

```
\let\eql@box@box\vcenter
\let\eql@box@open\@undefined
\let\eql@box@frame\@firstofone
\def\eql@box@wrap#1#2{\def\eql@box@frame##1{#1##1#2}}
```

```
\def\eql@box@delim#1#2{\def\eql@box@frame##1{\mathopen{}\mathclose{%
  \left#1##1\right#2}}}
\def\eql@box@getdim{\setbox\@ne\null\ht\@ne\ht\z@\dp\@ne\dp\z@}
\def\eql@box@deldim#1{\hbox{$\m@th\nulldelimiterspace\z@\left#1%
  \ifx\eql@box@box\vcenter\vcenter{\box\@ne}\else\box\@ne\fi\right.$}}
\def\eql@box@ldelim#1{\def\eql@box@frame##1{\mathopen{}\mathclose{%
  \eql@box@getdim\eql@box@deldim#1##1}}}
\def\eql@box@rdelim#1{\def\eql@box@frame##1{\mathopen{}\mathclose{%
  \eql@box@getdim##1\eql@box@deldim#1}}}
```

TODO: can we avoid setting `\eql@totalrows@` globally here? **TODO:** this is needed for escaping the box and then set the alignment **TODO:** maybe determine alignment within inner math?! **TODO:** difficulty: last line being known (for steps) only after all cells have been processed. Note: only works for single column anyway! we do not have to cater for more!

```
\def\eql@box@close{%
    \ifvmode\else
      \ifmmode\else\unskip\fi
      \global\let\eql@shape@lastrow\eql@true
      \eql@punct@apply@block
      \ifdefined\eql@box@punct@term
        \eql@punct@apply@top
      \fi
      \eql@box@endline
      \eql@box@lastcell
      \cr
    \fi
    \noalign{%
      \eql@hook@blockafter
      \global\let\eql@shape@lastrow\eql@false
    }%
    \eql@tagging@tablesaveinner
  \egroup
}
```

\eql@box@vcenter

```
\def\eql@box@vcenter#1{%
  \ifmmode
    \vcenter{#1}%
  \else
    $\m@th\vcenter{#1}$%
  \fi
}
```

\eql@box@start

```
\let\eql@box@endmath\eql@false
\def\eql@box@start{%
  \relax
  \ifmmode
    \let\eql@box@endmath\eql@false
  \else
    \let\eql@box@endmath\eql@true
    \expandafter$%$
  \fi
  \eql@beamerbasecolor@avoid
  \eql@stack@save@box
  \let\eql@frame@cmd\@undefined
  \let\eql@layoutleft\eql@false
  \eql@row@\z@
  \eql@totalrows@\@M
  \eql@shape@select
  \setbox\z@\ifx\eql@box@box\vcenter
    \expandafter\vbox
  \else
    \expandafter\eql@box@box
  \fi\bgroup
    \let\eqnpunct\eql@punct@setnext
    \eql@display@nest
    \let\\\eql@box@cr
    \ifdefined\eql@amp@mode
      \let\&\eql@box@amp
    \fi
    \eql@spread@set
    \eql@strut@make
    \eql@box@open
}
```

\eql@box@end

```
\def\eql@box@end{%
    \eql@box@close
  \egroup
  \eql@box@frame{%
    \ifdefined\eql@display@marginleft
      \hskip\glueexpr\eql@display@marginleft\relax
    \fi
    \ifx\eql@box@box\vcenter
      \eql@box@vcenter{\unvbox\z@}%
    \else
      \box\z@
    \fi
    \eql@tagging@tableaddinner
    \ifdefined\eql@display@marginright
      \hskip\glueexpr\eql@display@marginright\relax
    \fi
  }%
  \eql@stack@restore
  \ifdefined\eql@box@endmath
    \expandafter$%$
  \fi
}
```

`\eql@box@main` Combined opening, body and closing for pre-scanned body:

```
\def\eql@box@main{%
  \eql@box@start
  \eql@scan@body
  \eql@box@end
}
```

11.7 Options Processing

TODO: describe

```
\def\eql@box@test@setopt{\let\eql@box@test\eql@box@testopt}
\def\eql@box@test@setall{\let\eql@box@test\eql@box@testall}
```

`\eql@box@testopt` **TODO:** describe

```
\def\eql@box@testopt#1{\eql@testopt@tight{\eql@box@testopt@set{#1}}{}}
\def\eql@box@testopt@set#1[#2]{\eqnaddopt{#2}#1}
```

```
\def\eql@box@testall{\eql@parseopt@env\eql@box@testall@parse}
\def\eql@box@testall@parse{%
  \ifx\eql@parseopt@token[%]
    \let\eql@parseopt@next\eql@parseopt@opt
  \fi
  \ifx\eql@parseopt@token.%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token,%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token~%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token`%
    \let\eql@parseopt@next\eql@parseopt@punctopt
  \fi
  \ifx\eql@parseopt@token!%
    \let\eql@parseopt@next\eql@box@parse@puncterm
  \fi
  \ifx\eql@parseopt@token=%
    \let\eql@parseopt@next\eql@parseopt@lines
  \fi
  \ifx\eql@parseopt@token|%
    \let\eql@parseopt@next\eql@parseopt@columns
  \fi
  \ifx\eql@parseopt@token<%
    \let\eql@parseopt@next\eql@parseopt@ampeq
  \fi
  \ifx\eql@parseopt@token>%
    \let\eql@parseopt@next\eql@parseopt@eqamp
  \fi
}
\def\eql@box@parse@puncterm#1{\eqnaddopt{puncterm}\eql@parseopt@peek}
```

`\eql@box@end@testall` **TODO:** describe

```
\def\eql@box@end@testall{\eql@parseopt@env\eql@box@end@testall@parse}
\def\eql@box@end@testall@parse{%
  \ifx\eql@parseopt@token.%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token,%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token~%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token`%
    \let\eql@parseopt@next\eql@parseopt@punctblock
  \fi
  \ifx\eql@parseopt@token!%
    \let\eql@parseopt@next\eql@box@end@parse@puncterm
  \fi
  \ifx\eql@parseopt@token$%$
    \let\eql@parseopt@next\eql@parseopt@end
  \fi
}
\def\eql@box@end@parse@puncterm#1{%
  \let\eql@box@punct@term\eql@true\eql@parseopt@peek}
```

`\eql@box@processopt` **TODO:** describe

```
\def\eql@box@processopt{%
  \let\eql@box@frame\@firstofone
  \let\eql@display@marginleft\@undefined
  \let\eql@display@marginright\@undefined
  \let\eql@box@punct@term\eql@false
  \let\eql@punct@block\@undefined
  \eql@nextopt@process{box}%
  \let\eql@punct@next\@undefined
  \eql@colsep@\glueexpr\eql@box@colsep\relax
  \ifdefined\eql@paddingleft@val
    \eql@paddingleft@\glueexpr\eql@paddingleft@val\relax
  \else
    \eql@paddingleft@\z@
  \fi
  \ifdefined\eql@paddingright@val
    \eql@paddingright@\glueexpr\eql@paddingright@val\relax
  \else
    \eql@paddingright@\z@
  \fi
  \eql@indent@\glueexpr\eql@indent@val\relax
}
```

11.8 Environment

`equationsbox` (*env.*)

```
\newenvironment{equationsbox}{%
  \eql@verbose@info\eql@verbose@msg@enterenv
  \ifdefined\eql@box@env@modifier
    \eql@box@test@setall
  \else
    \eql@box@nomodifier
  \fi
  \eql@ampprotect\eql@box@test\eql@box@env@start
}{%
  \ifdefined\eql@box@doscan\else
    \expandafter\eql@box@end
  \fi
  \eql@verbose@info\eql@verbose@msg@leaveenv
}
```

`\eql@box@env@start`

```
\def\eql@box@env@start{%
  \eql@box@processopt
  \ifdefined\eql@box@doscan
    \eql@box@call@set
    \expandafter\eql@scan@env
  \else
    \expandafter\eql@box@start
  \fi
}
```

`\eql@box@call`

```
\def\eql@box@call{\eql@box@main\eql@scan@end}
\def\eql@box@call@test{%
  \eql@ampprotect\eql@box@end@testall\eql@box@call}
\def\eql@box@call@set{%
  \ifdefined\eql@box@end@modifier
    \let\eql@scan@call\eql@box@call@test
  \else
    \let\eql@scan@call\eql@box@call
  \fi
}
```

`\eql@box@ang@open`

```
\newenvironment{equationsbox@ang}{}{}
\def\eql@box@ang@open{%
  \expandafter\eqnaddopt\expandafter{\eql@box@ang@opt}%
  \begin{equationsbox@ang}%
  \eql@verbose@info\eql@verbose@msg@enterenv
  \let\>\eql@box@ang@close
  \ifdefined\eql@box@ang@modifier
    \eql@box@test@setall
  \else
    \eql@box@nomodifier
  \fi
  \eql@ampprotect\eql@box@test\eql@box@ang@start
}
```

`\eql@box@ang@start` Process arguments and start handling the box:

```
\def\eql@box@ang@start{%
  \eql@box@processopt
  \ifdefined\eql@box@doscan
    \eql@box@call@set
    \expandafter\eql@scan@ang
  \else
    \expandafter\eql@box@start
  \fi
}
```

`\eql@box@ang@close` **TODO:** describe

```
\def\eql@box@ang@close{%
  \ifdefined\eql@box@doscan
    \let\eql@box@end@modifier\eql@false
  \fi
  \ifdefined\eql@box@end@modifier
    \expandafter\eql@ampprotect\expandafter\eql@box@end@testall
  \fi
  \eql@box@ang@end
}
```

`\eql@box@ang@end` **TODO:** describe

```
\def\eql@box@ang@end{%
  \ifdefined\eql@box@doscan\else
    \expandafter\eql@box@end
  \fi
  \eql@verbose@info\eql@verbose@msg@leaveenv
  \end{equationsbox@ang}%
  \ignorespaces
}
```

12 Single-Line Equation

TODO: describe

12.1 Native Mode

```
\def\eql@single@start@native{%
  \eql@display@init
  \eql@display@print
  \let\raisetag\eql@raisetag@default
  \eql@shape@align@disable
  \eql@hook@eqin
}%
```

TODO: describe

```
\def\eql@single@end@native{%
  \eql@tags@container
  \eql@numbering@single@eval
  \if@eqnsw
    \ifdefined\eql@tagsleft
      \leqno
    \else
      \eqno
    \fi
    \eql@composetag@print
  \fi
  \eql@interline@container
  \advance\eql@belowspace@\eql@vspaceskip@
  \eql@display@container
  \eql@display@penalty
  \eql@display@vspace@native
}%
```

12.2 Print

```
\def\eql@single@start@print{%
  \eql@display@init
  \eql@display@print
  \eql@shape@align@enable

  \eql@totalrows@\@ne
  \eql@row@\@ne
  \eql@arrange@init
  \global\let\eql@cell@container\@empty

  \prevgraf\numexpr\prevgraf+\@ne\relax
  \setbox\eql@cellbox@\hbox\bgroup
    \eql@restore@hfuzz
    \eql@strut@cell
    $\m@th\eql@mathstyle%$
      \eql@hook@eqin
}

\def\eql@single@end@print{%
    \eql@tagging@mathsave
    $%$
    \hfil
    \kern\z@
  \egroup
  \prevgraf\numexpr\prevgraf-\@ne\relax

  \eql@shape@eval
  \eql@cell@container

  \ifdefined\eql@frame@cmd
    \eql@frame@adjust
  \fi

  \eql@cellwidth@\wd\eql@cellbox@
  \eql@line@height@\ht\eql@cellbox@
  \eql@line@depth@\dp\eql@cellbox@
  \eql@totalwidth@\eql@cellwidth@
  \eql@totalheight@\dimexpr\eql@line@height@+\eql@line@depth@\relax
  \eql@topheight@\eql@line@height@
  \eql@bottomdepth@\eql@line@depth@

  \eql@tags@container
  \eql@numbering@single@eval
  \if@eqnsw
    \eql@tagbox@make\eql@composetag@print
    \eql@tagrows@\@ne
    \ifdefined\eql@tagpos@reserve\else
      \eql@tagwidth@\z@
    \fi
    \eql@tagheight@block@\ht\eql@tagbox@
    \eql@tagdepth@block@\dp\eql@tagbox@
  \else
    \eql@numbering@warnunused
    \eql@tagwidth@\z@
    \eql@tagrows@\z@
  \fi
  \eql@tagwidth@max@\eql@tagwidth@
  \eql@tagpos@single@eval
  \eql@tagpos@print@line@eval

  \eql@intercolumns@\z@
  \eql@adjust@calc@lines

  \eql@display@halign@init{}%
  \halign{##\crcr
    \noalign{\eql@display@halign@start}%
    \eql@arrange@print@line
    \cr
    \noalign{\eql@display@halign@end}%
    \eql@tagging@tablesavelines
  }%
  \eql@tagpos@print@line@end
  \eql@display@close
}
```

13 Multi-Line with Single Column

TODO: outline sequence of calls

13.1 Measure

TODO: describe

```
\def\eql@lines@measure@line@begin{%
  \eql@verbose@info\eql@verbose@msg@startline@number
  \eql@numbering@measure@line@begin
  \eql@hook@linein
}
```

TODO: describe

```
\def\eql@lines@measure@line@end{%
  \eql@punct@apply@line
  \eql@hook@lineout
}
```

TODO: describe **TODO:** it would be an option to add the absolute shove amount to the calculation of the maximum width

```
\def\eql@lines@measure@cell{%
  \ifdefined\eql@frame@cmd
    \ifcase\eql@shape@pos@
```

```
      \eql@frame@measure
      \advance\eql@shape@amount@-\eql@frame@margin@
    \or\or
      \eql@frame@measure
      \advance\eql@shape@amount@+\eql@frame@margin@
    \fi
    \eql@frame@print
  \fi
  \eql@cellwidth@\wd\eql@cellbox@
  \eql@line@height@\ht\eql@cellbox@
  \eql@line@depth@\dp\eql@cellbox@
  \eql@dimensions@startrow
  \eql@dimensions@savecell
  \kern\eql@cellwidth@
}
```

\eql@lines@measure

```
\def\eql@lines@measure{%
  \eql@verbose@infoarg\eql@verbose@msg@enter\eql@lines@measure
  \eql@measure@init\eql@lines@measure@line@begin\eql@lines@measure@line@end
  \ifdefined\eql@amp@mode
    \let\&\eql@break@amp
  \fi
  \eql@totalrows@\@M
  \eql@shape@select

  \setbox\z@\vbox{\measuring@true\halign{%
      \global\let\eql@cell@container\@empty
      \setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
        $%
      }%
      \ifdefined\eql@shape@lastrow
        \eql@totalrows@\eql@row@
      \fi
      \eql@shape@eval
      \eql@cell@container
      \eql@lines@measure@cell
      \eql@measure@tag
      \eql@measure@endrow
    \crcr

    \noalign{%
      \global\let\eql@shape@lastrow\eql@false
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
    \eql@scan@body
    \ifvmode\else
      \global\let\eql@shape@lastrow\eql@true
      \eql@punct@apply@block
      \eql@hook@blockout
      \eql@multi@endline
      \cr
    \fi
    \omit
    \cr
    \noalign{%
      \eql@hook@blockafter
      \global\let\eql@shape@lastrow\eql@false
    }%
  }}%
```

```
  \eql@measure@close
```

```
  \setbox\z@\vbox{%
    \unvbox\z@
    \unpenalty
    \global\setbox\@ne\lastbox
  }%
  \eql@totalwidth@\wd\@ne
```

```
  \eql@verbose@infoarg\eql@verbose@msg@leave\eql@lines@measure
}
```

13.2 Column Placement

TODO: describe Find the best row for tag placement:

```
\def\eql@lines@adjust{%
  \eql@tagpos@adjust@eval
  \eql@adjust@calc@lines
  \eql@numbering@best@eval
}
```

13.3 Print

TODO: describe

`nes@print@line@begin`

```
\def\eql@lines@print@line@begin{%
  \eql@verbose@info\eql@verbose@msg@startline@number
  \eql@numbering@print@line@begin
  \eql@hook@linein
}
```

TODO: describe

```
\def\eql@lines@print@line@end{%
  \eql@punct@apply@line
  \eql@hook@lineout
}
```

TODO: describe

```
\def\eql@lines@print@line@adjust{%
  \ifdefined\eql@frame@cmd
    \ifcase\eql@shape@pos@
      \eql@frame@measure
      \advance\eql@shape@amount@-\eql@frame@margin@
    \or\or
      \eql@frame@measure
      \advance\eql@shape@amount@+\eql@frame@margin@
    \fi
    \eql@frame@adjust
  \fi
  \eql@cellwidth@\wd\eql@cellbox@
  \eql@line@height@\ht\eql@cellbox@
  \eql@line@depth@\dp\eql@cellbox@
  \eql@numbering@print@line@eval
  \if@eqnsw
    \eql@tagbox@make\eql@composetag@print
  \fi
  \eql@tagpos@print@line@eval
  \eql@arrange@print@line
  \eql@tagpos@print@line@end
}
```

TODO: describe

```
\def\eql@lines@print{%
  \eql@verbose@infoarg\eql@verbose@msg@enter\eql@lines@print
  \eql@arrange@init
  \eql@display@halign@init\eql@lines@print@line@begin
  \eql@multi@cr@let\eql@lines@print@line@end
  \ifdefined\eql@amp@mode
    \let\&\eql@break@amp
  \fi
  \tabskip\z@skip

  \halign{%
      \global\let\eql@cell@container\@empty
      \setbox\eql@cellbox@\hbox{%
        \eql@restore@hfuzz
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
          \eql@tagging@mathsave
        $%
        \hfil
        \kern\z@
      }%
      \eql@shape@eval
      \eql@cell@container
      \eql@lines@print@line@adjust
    \crcr

    \noalign{%
      \eql@display@halign@start
      \eql@numbering@print@block@begin
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
    \eql@scan@body
    \ifvmode\else
      \relax
      \eql@punct@apply@block
      \eql@hook@blockout
      \eql@multi@endline
      \cr
    \fi
    \noalign{%
      \eql@hook@blockafter
      \eql@display@halign@end
      \eql@verbose@infoarg\eql@verbose@msg@leave\eql@lines@print
    }%
    \eql@tagging@tablesavelines
  }%
}
```

14 Multi-Line with Multiple Columns

TODO: describe **TODO:** outline sequence of calls

14.1 Support

TODO: describe

\eql@columns@add@amp
@columns@completerow

```
\def\eql@columns@add@amp#1{\if m#1&\omit\expandafter\eql@columns@add@amp\fi}
\def\eql@columns@completerow{%
  \count@\numexpr\eql@totalcolumns@+\@ne-\eql@column@\relax
  \edef\eql@tmp{%
    \expandafter\eql@columns@add@amp\romannumeral\number\count@ 000q}%
  \eql@tmp
}
```

```
\def\eql@columns@overfull{%
  \dimen@\eql@line@width@
  \advance\dimen@-\hfuzz
  \ifdim\dimen@>\displaywidth
    \setbox\z@\hbox to\displaywidth{\hbox to\eql@line@width@{\hfil}}%
    \wd\z@\z@
    \ht\z@\eql@line@height@
    \dp\z@\eql@line@depth@
    \box\z@
  \fi
}
```

14.2 Column Breaks

TODO: describe

TODO: describe

```
\let\eql@amp@org\&
```

```
\def\eql@columns@amp{%
  \eql@ampprotecttwo\eql@columns@amp@testescape\eql@amp@org
    \eql@columns@amp@process}
\def\eql@columns@amp@testescape#1#2{\eql@ifnextgobble@tight/{#1}{%
```

```
  \relax
  \let\eql@punct@term\eql@false
  \let\eql@class@rel@composed\@empty
  \eql@columns@amp@test{#2}}}
```

TODO: describe

```
\def\eql@columns@amp@test@setopt{%
  \let\eql@columns@amp@test\eql@columns@amp@testopt}
\def\eql@columns@amp@test@setall{%
  \let\eql@columns@amp@test\eql@columns@amp@testall}
```

@columns@amp@testopt

TODO: describe

```
\let\eql@columns@amp@testopt\@empty
```

@columns@amp@testall
ns@amp@testall@parse

TODO: describe

```
\def\eql@columns@amp@testall{\eql@parseopt@cr\eql@columns@amp@testall@parse}
\def\eql@columns@amp@testall@parse{%
  \ifx\eql@parseopt@token.%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token,%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token~%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token`%
    \let\eql@parseopt@next\eql@parseopt@punctnext
  \fi
  \ifx\eql@parseopt@token!%
    \let\eql@parseopt@next\eql@parseopt@puncterm
  \fi
  \ifx\eql@parseopt@token=%
    \let\eql@parseopt@next\eql@parseopt@relsymb
  \fi
  \ifx\eql@parseopt@token;%
    \let\eql@parseopt@next\eql@parseopt@relcont
  \fi
  \ifx\eql@parseopt@token:%
    \let\eql@parseopt@next\eql@parseopt@relstart
  \fi
  \ifx\eql@parseopt@token|%
    \let\eql@parseopt@next\eql@parseopt@relord
  \fi
  \ifx\eql@parseopt@token?%
    \let\eql@parseopt@next\eql@columns@amp@parse@rel
  \fi
  \ifx\eql@parseopt@token&%
    \let\eql@parseopt@next\eql@parseopt@end
  \fi
}
\def\eql@columns@amp@parse@rel#1#2{%
  \def\eql@class@rel@composed{\eql@class@rel@make{#2}}%
  \eql@parseopt@end}
```

@columns@amp@process

```
\def\eql@columns@amp@process{%
  \ifdefined\eql@punct@term\eql@punct@apply@top\fi
  \edef\eql@tmp{%
    \ifx\eql@class@rel@composed\@empty
      \ifodd\eql@column@&\noexpand\eql@punct@next@clear\fi&%
    \else
      \ifodd\eql@column@\else&\fi%
      \unexpanded\expandafter{\eql@class@rel@composed}%
    \fi
  }%
  \eql@tmp
}
```

14.3 Transpose

TODO: describe

TODO: adjust to \&?!

TODO: describe

```
\let\eql@transpose@active\eql@false
\def\eql@transpose@end{\eql@transpose@end}
\def\eql@transpose@skip{&\eql@punct@next@clear}
\def\eql@transpose@complete{%
  \relax\ifodd\eql@column@\expandafter\eql@transpose@skip\fi&}
```

TODO: describe

```
\def\eql@transpose{%
  \eql@totalcolumns@\z@
  \eql@totalrows@\z@
  \expandafter\eql@transpose@scan@col\the\eql@scan@reg@\&\eql@transpose@end\&%
  \eql@scan@reg@{}%
  \eql@row@\z@
  \eql@transpose@output@row
}
```

TODO: describe

```
\def\eql@transpose@save@col#1{%
  \@namedef{eql@transpose@data@col@\the\eql@totalcolumns@}{%
    \ifcase\eql@row@#1\else\let\eql@tmp\eql@transpose@skip\fi}}
```

TODO: describe

```
\def\eql@transpose@scan@col#1\&{%
  \def\eql@tmpa{#1}%
  \ifx\eql@tmpa\eql@transpose@end\else
    \advance\eql@totalcolumns@\@ne
    \eql@row@\z@
    \let\eql@transpose@data@col\@empty
    \eql@transpose@scan@row#1\\\eql@transpose@end\\%
    \ifnum\eql@row@>\eql@totalrows@
      \eql@totalrows@\eql@row@
    \fi
    \expandafter\eql@transpose@save@col\expandafter{\eql@transpose@data@col}%
    \expandafter\eql@transpose@scan@col
  \fi
}
```

TODO: describe

```
\def\eql@transpose@append@row#1{%
  \advance\eql@row@\@ne
  \eql@append\eql@transpose@data@col{\or\def\eql@tmp{#1}}}
```

TODO: describe

```
\def\eql@transpose@scan@row#1\\{%
  \def\eql@tmpa{#1}%
  \ifx\eql@tmpa\eql@transpose@end\else
    \ifx\eql@transpose@active+
      \eql@transpose@scan@cell#1&\eql@transpose@end&%
    \else
      \eql@transpose@append@row{#1}%
    \fi
    \expandafter\eql@transpose@scan@row
  \fi
}
```

TODO: describe

```
\def\eql@transpose@scan@cell#1&#2&{%
  \def\eql@tmpa{#2}%
  \ifx\eql@tmpa\eql@transpose@end
    \eql@transpose@append@row{#1}%
  \else
    \eql@transpose@append@row{#1&#2}%
    \expandafter\eql@transpose@scan@cell@next
  \fi
}
```

TODO: describe

```
\def\eql@transpose@scan@cell@next#1&{%
  \def\eql@tmpa{#1}%
  \ifx\eql@tmpa\eql@transpose@end\else
    \eql@transpose@append@row{&#1}%
    \expandafter\eql@transpose@scan@cell@next
  \fi
}
```

TODO: describe

```
\def\eql@transpose@output@row{%
  \ifnum\eql@row@<\eql@totalrows@
    \advance\eql@row@\@ne
    \eql@column@\z@
    \eql@transpose@output@col
    \ifnum\eql@row@<\eql@totalrows@
      \eql@scan@addto\\%
    \fi
    \expandafter\eql@transpose@output@row
  \fi
}
```

TODO: describe

```
\def\eql@transpose@output@col{%
  \ifnum\eql@column@<\eql@totalcolumns@
    \advance\eql@column@\@ne
    \csname eql@transpose@data@col@\the\eql@column@\endcsname
    \expandafter\eql@scan@addto\expandafter{\eql@tmp}%
    \ifnum\eql@column@<\eql@totalcolumns@
      \eql@scan@addto{\eql@transpose@complete}%
    \fi
    \expandafter\eql@transpose@output@col
  \fi
}
```

14.4 Measure

TODO: describe **TODO:** this is called also for extra line and concluding cr

`s@measure@line@begin`

```
\def\eql@columns@measure@line@begin{%
  \eql@verbose@info\eql@verbose@msg@startline@number
  \global\eql@column@\z@
  \global\eql@line@height@\z@
  \global\eql@line@depth@\z@
  \eql@numbering@measure@line@begin
  \eql@hook@linein
}
```

```
\def\eql@columns@measure@cell{%
  \eql@cellwidth@\wd\eql@cellbox@
  \ifdefined\eql@frame@cmd
    \eql@frame@measure
    \advance\eql@cellwidth@\eql@frame@margin@
  \fi
  \ifdim\ht\eql@cellbox@>\eql@line@height@
    \global\eql@line@height@\ht\eql@cellbox@
  \fi
  \ifdim\dp\eql@cellbox@>\eql@line@depth@
    \global\eql@line@depth@\dp\eql@cellbox@
  \fi
  \ifnum\eql@column@=\@ne
    \eql@dimensions@startrow
  \fi
  \ifodd\eql@column@
    \eql@shape@pos@\tw@
  \else
    \eql@shape@pos@\z@
  \fi
  \eql@shape@amount@\z@
  \eql@dimensions@savecell
  \ifodd\eql@column@\else
    \eql@dimensions@savesep
  \fi
  \kern\eql@cellwidth@
}
```

`mns@measure@line@end`

```
\def\eql@columns@measure@line@end{%
  \eql@punct@apply@line
  \eql@hook@lineout
  &\omit
  \ifnum\eql@column@>\eql@totalcolumns@
```

```
    \global\eql@totalcolumns@\eql@column@
  \fi
```

TODO: not sure whether saving the last cell value makes sense, but rather not increase `\eql@totalcolumns@` because that will disable the fallback to lines mode. **TODO:** additional column in width table is accounted for in column table

```
  \ifdefined\eql@frame@cmd
    \advance\eql@column@\@ne
    \wd\eql@cellbox@\z@
    \eql@columns@measure@cell
  \fi
  \eql@measure@tag
  \eql@measure@endrow
}
```

`\eql@columns@measure`

```
\def\eql@columns@measure{%
  \eql@verbose@infoarg\eql@verbose@msg@enter\eql@columns@measure
  \eql@totalcolumns@\z@
  \eql@measure@init\eql@columns@measure@line@begin\eql@columns@measure@line@end
  \ifdefined\eql@amp@mode
    \let\&\eql@columns@amp
  \fi

  \setbox\z@\vbox{\measuring@true\halign{%
    &%
      \global\advance\eql@column@\@ne
      \global\let\eql@cell@container\@empty
      \global\setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@next
          \eql@class@innerleft
          \eql@hook@innerleft
        $%
      }%
      \eql@cell@container
      \hfil
      \eql@columns@measure@cell
      \global\let\eql@frame@prevcmd\eql@frame@cmd
    &%
      \eql@prevwidth@\wd\eql@cellbox@
      \let\eql@frame@cmd\eql@frame@prevcmd
      \global\advance\eql@column@\@ne
      \global\let\eql@cell@container\@empty
      \setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \eql@hook@innerright
          \eql@class@innerright@sel
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
    $%
   }%
   \eql@cell@container
   \eql@columns@measure@cell
   \hfil
  \crcr

  \noalign{%
   \eql@hook@blockbefore
  }%
  \eql@hook@blockin
  \eql@scan@body

  \ifvmode\else
   \eql@punct@apply@block
   \eql@hook@blockout
   \eql@multi@endline
   \cr
  \fi
  \noalign{%
   \eql@hook@blockafter
  }%
```

TODO: note we also include the tag column as a backup

```
  \omit
  \eql@column@\@ne
  \eql@columns@completerow
  \cr
 }}%

 \eql@measure@close

 \setbox\z@\vbox{%
  \unvbox\z@
  \unpenalty
  \global\setbox\@ne\lastbox
 }%
 \eql@totalwidth@\wd\@ne
```

TODO: why not recycle box contents altogether?!

```
 \let\eql@colwidth@tab\@empty
 \loop
  \setbox\@ne\hbox{%
   \unhbox\@ne
   \unskip
   \global\setbox\thr@@\lastbox
  }%
 \ifhbox\thr@@
  \eql@colwidth@save{\wd\thr@@}%
 \repeat

 \eql@verbose@infoarg\eql@verbose@msg@leave\eql@columns@measure
}
```

14.5 Columns Placement

TODO: describe Make sure we have complete pairs of right and left adjusted columns, otherwise add a final empty column:

```
\def\eql@columns@adjust{%
  \ifodd\eql@totalcolumns@
    \advance\eql@totalcolumns@\@ne
  \fi
  \eql@tagpos@adjust@eval
  \eql@adjust@calc@columns
}
```

14.6 Print

TODO: describe

mns@print@line@begin

```
\def\eql@columns@print@line@begin{%
  \eql@verbose@info\eql@verbose@msg@startline@number
  \global\eql@column@\z@
  \global\eql@line@pos@\eql@marginleft@
  \global\eql@line@width@\z@
  \global\eql@line@avail@\eql@totalwidth@
  \global\eql@line@height@\z@
  \global\eql@line@depth@\z@
  \eql@numbering@print@line@begin
  \eql@hook@linein
}
```

l@columns@print@cell

```
\def\eql@columns@print@cell{%
  \eql@cellwidth@\wd\eql@cellbox@
  \ifodd\eql@column@
    \ifdefined\eql@frame@cmd
      \eql@frame@measure
      \advance\eql@cellwidth@\eql@frame@margin@
    \fi
    \dimen@\z@
  \else
    \advance\eql@cellwidth@-\eql@prevwidth@
```

draw a frame

```
    \ifdefined\eql@frame@cmd
      \eql@frame@measure
      \advance\eql@cellwidth@\eql@frame@margin@
      \advance\eql@prevwidth@\eql@frame@margin@
      \eql@frame@print
    \fi
```

update height and depth

```
    \ifdim\ht\eql@cellbox@>\eql@line@height@
      \global\eql@line@height@\ht\eql@cellbox@
    \fi
    \ifdim\dp\eql@cellbox@>\eql@line@depth@
      \global\eql@line@depth@\dp\eql@cellbox@
    \fi
```

print box

```
    \kern-\eql@prevwidth@
    \unhbox\eql@cellbox@
    \dimen@-\eql@cellwidth@
  \fi
```

enforce given width: hopefully measure was correct, but need a precise width for tag placement

```
  \advance\dimen@\eql@colwidth@get\eql@column@\relax
  \kern\dimen@
```

update available and used space

```
  \dimen@\eql@colwidth@get\eql@column@\relax
  \ifdim\eql@cellwidth@>\z@
    \ifdim\eql@line@width@=\z@
      \eql@line@avail@\eql@line@pos@
      \ifodd\eql@column@
        \advance\eql@line@avail@\dimen@
        \advance\eql@line@avail@-\eql@cellwidth@
      \fi
      \global\eql@line@avail@\eql@line@avail@
    \fi
    \eql@line@width@\eql@line@pos@
    \ifodd\eql@column@
      \advance\eql@line@width@\dimen@
    \else
      \advance\eql@line@width@\eql@cellwidth@
    \fi
    \global\eql@line@width@\eql@line@width@
  \fi
  \advance\eql@line@pos@\dimen@
  \ifodd\eql@column@\else
    \advance\eql@line@pos@\eql@colsep@
  \fi
  \global\eql@line@pos@\eql@line@pos@
}
```

```
\def\eql@columns@print@trailright{%
   &\omit
    \eql@prevwidth@\wd\eql@cellbox@
    \let\eql@frame@cmd\eql@frame@prevcmd
    \global\advance\eql@column@\@ne
    \eql@columns@print@cell
}
```

`lumns@print@line@end` **TODO:** add an even column with empty stuff if box processing deferred

```
\def\eql@columns@print@line@end{%
  \eql@punct@apply@line
  \eql@hook@lineout
  \ifodd\eql@column@
    \expandafter\eql@columns@print@trailright
  \fi
  \eql@columns@completerow
  \eql@columns@print@tag
}
```

`ql@columns@print@tag`

```
\def\eql@columns@print@tag{%
  \kern-\dimexpr\eql@totalwidth@+\eql@colsep@\relax
```

determine first line available space

```
  \eql@display@firstavail@set\eql@line@avail@
  \eql@columns@overfull
  \eql@numbering@print@line@eval
  \if@eqnsw
    \eql@tagbox@make\eql@composetag@print
  \fi
  \eql@tagpos@print@line@eval
  \eql@tagbox@print@cell
  \eql@tagpos@print@line@end
}
```

`\eql@columns@print`

```
\def\eql@columns@print{%
  \eql@verbose@infoarg\eql@verbose@msg@enter\eql@columns@print
  \eql@shape@align@disable
  \eql@display@halign@init\eql@columns@print@line@begin
  \eql@multi@cr@let\eql@columns@print@line@end
  \ifdefined\eql@amp@mode
    \let\&\eql@columns@amp
  \fi
  \tabskip\eql@marginleft@

  \halign{%
    &%
      \global\advance\eql@column@\@ne
      \global\let\eql@cell@container\@empty
      \global\setbox\eql@cellbox@\hbox{%
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
          \eql@hook@colin
          ##%
          \eql@punct@apply@next
          \eql@class@innerleft
          \eql@hook@innerleft
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \eql@cell@container
      \hfil
      \eql@columns@print@cell
      \global\let\eql@frame@prevcmd\eql@frame@cmd
      \tabskip\z@skip
    &%
      \eql@prevwidth@\wd\eql@cellbox@
      \let\eql@frame@cmd\eql@frame@prevcmd
      \global\advance\eql@column@\@ne
      \global\let\eql@cell@container\@empty
      \setbox\eql@cellbox@\hbox{%
        \unhbox\eql@cellbox@
        \eql@strut@cell
        \@lign
        $\m@th\eql@mathstyle
```

```
          \eql@hook@innerright
          \eql@class@innerright@sel
          ##%
          \eql@punct@apply@col
          \eql@hook@colout
          \eql@tagging@mathsave
        $%
        \eql@tagging@mathaddlast
      }%
      \eql@cell@container
      \eql@columns@print@cell
      \hfil
      \tabskip\eql@colsep@\relax
    \crcr
```

```
    \noalign{%
      \eql@display@halign@start
      \eql@numbering@print@block@begin
      \eql@hook@blockbefore
    }%
    \eql@hook@blockin
    \eql@scan@body
    \ifvmode\else
      \relax
      \eql@punct@apply@block
      \eql@hook@blockout
      \eql@multi@endline
      \cr
    \fi
    \noalign{%
      \eql@hook@blockafter
      \eql@display@halign@end
      \eql@verbose@infoarg\eql@verbose@msg@leave\eql@columns@print
    }%
    \eql@tagging@tablesavealign
  }%
}
```

15 Interface

15.1 Options Processing

TODO: describe

```
\def\eql@equations@test@setopt{\let\eql@equations@test\eql@equations@testopt}
\def\eql@equations@test@setall{\let\eql@equations@test\eql@equations@testall}
```

`ql@equations@testopt` **TODO:** describe

```
\def\eql@equations@testopt#1{%
  \eql@testopt@tight{\eql@equations@testopt@set{#1}}{}}
\def\eql@equations@testopt@set#1[#2]{\eqnaddopt{#2}#1}
```

`ql@equations@testall` The macro sequence started by `\eql@equations@testall` scans for optional arguments to the equation environments and appends them to the argument list using `\eqnaddopt`. All arguments are scanned such that any spaces stop the scanning and such that any alignment markers ‘`&`’ cannot interfere: **TODO:** update

```
\def\eql@equations@testall{\eql@parseopt@env\eql@equations@testall@parse}
\def\eql@equations@testall@parse{%
  \ifx\eql@parseopt@token*%
    \let\eql@parseopt@next\eql@parseopt@nonumber
  \fi
  \ifx\eql@parseopt@token!%
    \let\eql@parseopt@next\eql@parseopt@donumber
  \fi
  \ifx\eql@parseopt@token/%
    \let\eql@parseopt@next\eql@parseopt@transpose
  \fi
  \ifx\eql@parseopt@token[%]
    \let\eql@parseopt@next\eql@parseopt@opt
  \fi
  \ifx\eql@parseopt@token\eql@atxi
    \let\eql@parseopt@next\eql@parseopt@label
  \fi
  \ifx\eql@parseopt@token\eql@atxii
    \let\eql@parseopt@next\eql@parseopt@label
  \fi
  \ifx\eql@parseopt@token.%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token,%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token~%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token`%
    \let\eql@parseopt@next\eql@parseopt@punctopt
  \fi
  \ifx\eql@parseopt@token-%
    \let\eql@parseopt@next\eql@parseopt@single
  \fi
  \ifx\eql@parseopt@token=%
    \let\eql@parseopt@next\eql@parseopt@lines
  \fi
  \ifx\eql@parseopt@token|%
    \let\eql@parseopt@next\eql@parseopt@columns
  \fi
  \ifx\eql@parseopt@token<%
    \let\eql@parseopt@next\eql@parseopt@ampeq
  \fi
  \ifx\eql@parseopt@token>%
    \let\eql@parseopt@next\eql@parseopt@eqamp
  \fi
  \ifx\eql@parseopt@token\label
    \let\eql@parseopt@next\eql@parseopt@end
  \fi
  \ifx\eql@parseopt@token\begin
    \let\eql@parseopt@next\eql@parseopt@end
  \fi
}
```

`quations@end@testall` **TODO:** describe

```
\def\eql@equations@end@testall{%
  \eql@parseopt@env\eql@equations@end@testall@parse}
\def\eql@equations@end@testall@parse{%
  \ifx\eql@parseopt@token.%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token,%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token~%
    \let\eql@parseopt@next\eql@parseopt@punctpass
  \fi
  \ifx\eql@parseopt@token`%
    \let\eql@parseopt@next\eql@parseopt@punctblock
  \fi
}
```

`equations@processopt` The macro `\eql@equations@processopt` processes the options recevied by `\eqnaddopt`. First, clear several non-persistent registers (labels, tags, direct vertical spacing). Then process the arguments. Finally evaluate `\eql@indent@val` and `\eql@tagsepmin@val` and prevent main punctuation from being passed to nested environments:

```
\def\eql@equations@processopt{%
  \let\eql@tags@container@block\eql@tags@container@clear
  \let\eql@tags@frame@cmd\@firstofone
  \let\eql@skip@force@above\@undefined
  \let\eql@skip@force@below\@undefined
  \let\eql@skip@force@leave\@undefined
  \let\eql@display@linewidth\@undefined
  \let\eql@display@marginleft\@undefined
  \let\eql@display@marginright\@undefined
  \eql@abovespace@\z@skip
  \eql@belowspace@\z@skip
  \eql@displaybreak@prepen@\@MM
  \eql@displaybreak@postpen@\@MM
  \eql@nextopt@process{equations}%
  \let\eql@punct@next\@undefined
  \eql@indent@\glueexpr\eql@indent@val\relax
  \eql@tagsepmin@\glueexpr\eql@tagsepmin@val\relax
}
```

15.2 Single-Line Main

In the following, we define the main routine for the single-line equation mode.

`\eql@single@cr@error` Cannot use line breaks, produce an error message:

```
\def\eql@single@cr@error{%
  \eql@error{Cannot use '\string\\' within display equation.
    Please switch to equations environment}%
}
```

`\eql@single@start` Opening code for single-line equation. Capture current vertical mode, trigger PDF tagging, enter display math mode, initialise numbering scheme, backup current state of global registers, set native vs. manual equation tag mode, install error message for using \\. Hand over to mode-specific opening:

```
\def\eql@single@start{%
  \eql@display@enter
  \eql@tagging@start
  \eql@dollardollar@begin
  \eql@display@adjust
  \eql@numbering@init
  \eql@stack@save@equations
  \eql@numbering@single@init
  \ifdefined\eql@single@cr@mode
    \let\\\eql@single@cr@mode
  \fi
  \ifdefined\eql@amp@mode
    \let\&\eql@break@amp
  \fi
  \ifdefined\eql@single@native
    \let\eql@single@start@sel\eql@single@start@native
    \let\eql@single@end@sel\eql@single@end@native
  \else
    \let\eql@single@start@sel\eql@single@start@print
    \let\eql@single@end@sel\eql@single@end@print
  \fi
  \eql@single@start@sel
}
```

`\eql@single@end` Closing code for single-line equation. Apply punctuation for the block, perform mode-specific ending, restore global variables, end display math, indicate end to PDF tagging, return to vertical mode if desired:

```
\def\eql@single@end{%
  \eql@punct@apply@block
  \eql@hook@eqout
  \eql@single@end@sel
  \global\eql@punct@top@reset
  \eql@stack@restore
  \eql@dollardollar@end
  \eql@tagging@end
  \eql@display@leave
}
```

`\eql@single@main` Combined opening, body and closing for pre-scanned body:

```
\def\eql@single@main{%
  \eql@single@start
  \eql@scan@body
  \eql@single@end
}
```

`equations@single@set` Configure equations macros to single-line mode:

```
\def\eql@equations@single@set{%
  \ifdefined\eql@single@doscan
    \let\eql@equations@main\eql@single@main
  \else
    \let\eql@equations@main\@undefined
  \fi
}
```

15.3 Multi-Line Main

`ulti@linesmode` (*bool*) Switch register for lines vs. columns mode:

```
\let\eql@multi@linesmode\eql@false
```

`\eql@multi@main` Main routine for multi-line modes. Capture current vertical mode, trigger PDF tagging, enter display math mode, initialise numbering scheme, backup current state of global registers, initialise macros for use within equations: **TODO:** shove depends on lines vs columns

```
\def\eql@multi@main{%
  \eql@display@enter
  \eql@tagging@start
  \eql@dollardollar@begin
  \eql@display@adjust
  \eql@numbering@init
  \eql@stack@save@equations
  \ifdefined\eql@transpose@active
    \ifdefined\eql@multi@linesmode\else
      \eql@transpose
    \fi
  \fi
  \ifdefined\eql@numbering@subeq@use
    \eql@numbering@subeq@init
  \fi
  \eql@display@init
  \let\intertext\eql@intertext
  \let\endintertext\endeql@intertext
  \eql@shape@align@enable
```

Now measure the given multi-line equations body:

```
  \ifdefined\eql@multi@linesmode
    \eql@lines@measure
  \else
    \ifdefined\eql@ampproof@active
      \eql@ampproof
    \fi
    \eql@columns@measure
  \fi
```

If only a single equation number is used for subequation numbering, revert to normal equation numbering. If only a single column is used in columns mode, may fallback to lines mode. Switching from columns to lines mode, the width can be incorrect, expect only minor discrpancies, but for accurateness, should call `\eql@lines@measure`:

```
  \ifdefined\eql@numbering@subeq@use
    \eql@numbering@subeq@test
  \fi
  \ifdefined\eql@multi@linesmode\else
    \ifdefined\eql@multi@linesfallback
      \ifnum\eql@totalcolumns@=\@ne
        \let\eql@multi@linesmode\eql@true
        \ifx\eql@multi@linesfallback\z@\else
          \eql@lines@measure
        \fi
      \fi
    \fi
  \fi
```

Adjust the multi-line equations body:

```
  \ifdefined\eql@multi@linesmode
    \eql@lines@adjust
  \else
    \eql@columns@adjust
  \fi
```

Now print the multi-line equations body:

```
  \eql@display@print
  \eql@numbering@print@init
  \ifdefined\eql@multi@linesmode
    \eql@lines@print
  \else
    \eql@columns@print
  \fi
  \eql@display@close
```

Close numbering, restore global variables, end display math, indicate end to PDF tagging, return to vertical mode if desired:

```
  \ifdefined\eql@numbering@subeq@use
    \eql@numbering@subeq@close
  \fi
  \global\eql@punct@top@reset
  \eql@stack@restore
  \eql@dollardollar@end
  \eql@tagging@end
  \eql@display@leave
}
```

`quations@columns@set`
`@equations@lines@set`

Configure equations macros to one of the two multi-line modes:

```
\def\eql@equations@columns@set{%
  \let\eql@equations@main\eql@multi@main
  \let\eql@multi@linesmode\eql@false
}
\def\eql@equations@lines@set{%
  \let\eql@equations@main\eql@multi@main
  \let\eql@multi@linesmode\eql@true
}
```

15.4 Equations Environment

We now declare the main environment and its symbolic versions.

Environment.

`equations` (*env.*)

Declare the main equations environment. If already in math mode, fail and cancel the environment body. Otherwise scan for optional arguments and pass on to `\eql@equations@start`:

```
\newenvironment{equations}{%
  \ifmmode
    \expandafter\eql@equations@env@cancel
  \else
    \eql@verbose@info\eql@verbose@msg@enterenv
    \expandafter\eql@equations@env@open
  \fi
}{%
  \ifdefined\eql@equations@main\else
    \expandafter\eql@single@end
  \fi
  \ignorespacesafterend
  \eql@verbose@info\eql@verbose@msg@leaveenv
}
\eql@markline@amsthm@register{equations}
\eql@tagging@register@luamml{equations}
```

`equations@env@cancel`

```
\def\eql@equations@env@cancel{%
  \eql@error@mathmode{\string\begin{\@currenvir}}%
  \let\eql@scan@call\eql@scan@env@cancel
  \eql@scan@env
}
```

`l@equations@env@open`

```
\def\eql@equations@env@open{%
  \ifdefined\eql@equations@env@modifier
    \eql@equations@test@setall
  \else
    \eql@equations@nomodifier
  \fi
  \eql@ampprotect\eql@equations@test\eql@equations@env@start
}
```

`@equations@env@start`

The macro `\eql@equations@env@start` first processes the arguments. Depending on the chosen mode of operation, scan the environment body passing on to `\eql@equations@main` or process a single-line equation via `\eql@single@start`:

```
\def\eql@equations@env@start{%
  \eql@equations@processopt
  \ifdefined\eql@equations@main
    \eql@equations@call@set
    \expandafter\eql@scan@env
  \else
    \expandafter\eql@single@start
  \fi
}
```

`\eql@equations@call`

```
\def\eql@equations@call{\eql@equations@main\eql@scan@end}
\def\eql@equations@call@test{%
  \eql@ampprotect\eql@equations@end@testall\eql@equations@call}
\def\eql@equations@call@set{%
  \ifdefined\eql@equations@end@modifier
    \let\eql@scan@call\eql@equations@call@test
  \else
    \let\eql@scan@call\eql@equations@call
  \fi
}
```

Square Brackets.

`equations@sqr` (*env.*) Define a pseudo-environment `equations@sqr` such that `\@currenvir` may point to it when needed:

```
\newenvironment{equations@sqr}{}{}
\eql@markline@amsthm@register{equations@sqr}
\eql@tagging@register@luamml{equations@sqr}
```

`l@equations@sqr@open` Definition for '`\[`'. Add the default arguments `\eql@equations@sqr@opt`, enter the pseudo-environment, scan for optional arguments, and pass on to `\eql@equations@sqr@start`:

```
\def\eql@equations@sqr@open{%
  \expandafter\eqnaddopt\expandafter{\eql@equations@sqr@opt}%
  \begin{equations@sqr}%
  \eql@verbose@info\eql@verbose@msg@enterenv
  \let\]\eql@equations@sqr@close
  \ifdefined\eql@equations@sqr@modifier
    \eql@equations@test@setall
  \else
    \eql@equations@nomodifier
  \fi
  \eql@ampprotect\eql@equations@test\eql@equations@sqr@start
}
```

`@equations@sqr@start` Process arguments. Depending on mode of operation, scan and process enclosed contents via `\eql@equations@main` or pass on to `\eql@single@start`:

```
\def\eql@equations@sqr@start{%
  \eql@equations@processopt
  \ifdefined\eql@equations@main
    \eql@equations@call@set
    \expandafter\eql@scan@sqr
  \else
    \expandafter\eql@single@start
  \fi
}
```

`@equations@sqr@close` Definition for '`\]`'. Parse modifiers following '`\]`' and hand on to `\eql@equations@sqr@end`:

```
\protected\def\eql@equations@sqr@close{%
  \ifdefined\eql@equations@main
    \let\eql@equations@end@modifier\eql@false
  \fi
  \ifdefined\eql@equations@end@modifier
    \expandafter\eql@ampprotect\expandafter\eql@equations@end@testall
  \fi
  \eql@equations@sqr@end
}
```

`ql@equations@sqr@end` **TODO:** complete End `\[`...`\]` block:

```
\def\eql@equations@sqr@end{%
  \ifdefined\eql@equations@main\else
    \expandafter\eql@single@end
  \fi
  \eql@verbose@info\eql@verbose@msg@leaveenv
  \end{equations@sqr}%
  \ignorespaces
}
```

Angle Brackets.

`equations@ang` (*env.*) Define a pseudo-environment `equations@ang`:

```
\newenvironment{equations@ang}{}{}
\eql@markline@amsthm@register{equations@ang}
\eql@tagging@register@luamml{equations@ang}
```

`l@equations@ang@open`

```
\def\eql@equations@ang@open{%
  \expandafter\eqnaddopt\expandafter{\eql@equations@ang@opt}%
  \begin{equations@ang}%
  \eql@verbose@info\eql@verbose@msg@enterenv
  \let\>\eql@equations@ang@close
  \ifdefined\eql@equations@ang@modifier
    \eql@equations@test@setall
  \else
    \eql@equations@nomodifier
  \fi
  \eql@ampprotect\eql@equations@test\eql@equations@ang@start
}
```

`@equations@ang@start` Process arguments and start handling the equation:

```
\def\eql@equations@ang@start{%
  \eql@equations@processopt
  \ifdefined\eql@equations@main
    \eql@equations@call@set
    \expandafter\eql@scan@ang
  \else
    \expandafter\eql@single@start
  \fi
}
```

`@equations@ang@close` **TODO:** describe

```
\def\eql@equations@ang@close{%
  \ifdefined\eql@equations@main
    \let\eql@equations@end@modifier\eql@false
  \fi
  \ifdefined\eql@equations@end@modifier
    \expandafter\eql@ampprotect\expandafter\eql@equations@end@testall
  \fi
  \eql@equations@ang@end
}
```

`ql@equations@ang@end` **TODO:** describe

```
\def\eql@equations@ang@end{%
  \ifdefined\eql@equations@main\else
    \expandafter\eql@single@end
  \fi
  \eql@verbose@info\eql@verbose@msg@leaveenv
  \end{equations@ang}%
  \ignorespaces
}
```

16 Options

16.1 Selection Tools

`ql@decide@abovebelow` Select between values ‘`above`’ or ‘`below`’ or both: execute the corresponding code provided in the latter two arguments:

```
\def\eql@decide@abovebelow#1#2#3#4#5{%
  \eql@decide@select{#1}{#2}{#3}{%
    {abovebelow,both,tb}{#4#5},%
    {above,top,t}{#4},%
    {below,bottom,b}{#5}}}
```

`eql@decide@situation` Select a particular vertical spacing situation and store it in the macro `#4`:

```
\def\eql@decide@situation#1#2#3#4{%
  \eql@decide@select{#1}{#2}{#3}{%
    {{long}{\def#4{0}}},%
    {{short}{\def#4{1}}},%
    {{cont}{\def#4{2}}},%
    {{par}{\def#4{3}}},%
    {{top}{\def#4{4}}},%
    {{noskip}{\def#4{5}}},%
    {{medskip}{\def#4{6}}}}}
```

`\eql@decide@delim` **TODO:** describe

```
\def\eql@decide@delim#1#2#3{%
  \eql@decide@select{#1}{#2}{#3}{%
    {{,.,\eql@decide@false}{\eql@box@wrap{}{}}},%
    {{\eql@decide@true,r,p,round,parentheses}{\eql@box@delim()}},%
    {{s,b,sqr,square}{\eql@box@delim[]}},%
    {{c,B,curly,braces}{\eql@box@delim\lbrace\rbrace}},%
    {{a,ang,angle}{\eql@box@delim\langle\rangle}},%
    {{v,vert}{\eql@box@delim\vert\vert}},%
    {{d,V,dvert}{\eql@box@delim\Vert\Vert}},%
    {\relax{\eql@box@delim#3}}}}%
```

TODO: describe

```
\def\eql@keyall{equations,box,setup}
```

16.2 Options Declarations

We now declare all key-value pairs for options sorted by their category.

Modes for Equations Box Environment. Declare horizontal and vertical alignment modes for the boxed equations environment. Also declare spacing of columns:

```
\eql@define@key{box}{lines,ln,gathered,gather,ga}[]{%
  \eql@box@lines@set}
\eql@define@key{box}{columns,col,aligned,align,al}[]{%
```

```
  \eql@box@columns@set}
\eql@define@key{box}{cases}[]{%
  \eql@box@cases@set\eql@box@ldelim\lbrace%
  \def\eql@box@colsep{\eql@box@condsep}}
\eql@define@key{box}{matrix}[r]{%
  \eql@box@lines@set\eql@shape@set{center}%
  \let\eql@spread@reset\eql@true\def\eql@spread@val{\z@}%
  \def\eql@box@colsep{\eql@box@matrixsep}%
  \let\eql@mathstyle\@empty
  \eql@punct@clear
  \eql@box@cr@test@setopt
  \eql@box@amp@test@setopt
  \let\eql@box@end@modifier\eql@false
  \eql@decide@delim{#3}{#2}{#1}}
\eql@define@key{box}{top,t}[]{\let\eql@box@box\vtop}
\eql@define@key{box}{center,c}[]{\let\eql@box@box\vcenter}
\eql@define@key{box}{bottom,b}[]{\let\eql@box@box\vbox}
\eql@define@key{box}{intro}{%
  \def\eql@box@cases@condintro{#1}}
\eql@define@key{box}{introtext}{%
  \def\eql@box@cases@condintro{%
    \ifmmode\expandafter\hbox\else\expandafter\@firstofone\fi{#1 }}}
\eql@define@key{box}{textcond}[true]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{\eql@decide@true,text}{\let\eql@box@cases@condtext\eql@true}},%
    {{\eql@decide@false,math}{\let\eql@box@cases@condtext\eql@false}}}}
\eql@define@key{setup}{scanbox}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@box@doscan}
\eql@define@key{box}{scan}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@box@doscan}
\eql@define@key{setup}{boxangopt}[]{%
  \def\eql@box@ang@opt{columns,#1}}
```

Modes for Equations Environment.

```
\let\eql@box@doscan\eqn@false
```

Declare modes and switches for the equations environment:

```
\eql@define@key{equations}{single,1,equation,eq}[]{\eql@equations@single@set}
\eql@define@key{equations}{lines,ln,gathered,gather,ga}[]{%
  \eql@equations@lines@set}
\eql@define@key{equations}{columns,col,aligned,align,al}[]{%
  \eql@equations@columns@set}
\eql@define@key{equations,setup}{transpose}[true]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@transpose@active\eql@false}},%
    {{noamp,plain,restricted}{\let\eql@transpose@active\eql@true}},%
    {{\eql@decide@true,amp,cont}{\let\eql@transpose@active=+}}}}
\eql@define@key{setup}{nativeequation}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@single@native}
\eql@define@key{equations}{native}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@single@native%
  \ifdefined\eql@single@native\let\eql@layoutleft\eql@false\fi}
\eql@define@key{setup}{scanequation}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@single@doscan}
\eql@define@key{equations}{scan}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@single@doscan}
\eql@define@key{setup}{sqropt}[]{%
```

```
  \def\eql@equations@sqr@opt{equation,#1}}
\eql@define@key{setup}{angopt}[]{%
  \def\eql@equations@ang@opt{columns,#1}}
```

TODO: describe

```
\eql@define@key{control}{restoreexterior}[]{\eql@display@restore}
\eql@define@key{control}{restoreexterior*}[]{\@arrayparboxrestore}
```

Modes for Modifier Scanning.

```
\def\eql@equations@nomodifier{%
  \eql@equations@test@setopt
  \let\eql@equations@end@modifier\eql@false
  \eql@multi@cr@test@setopt
  \eql@columns@amp@test@setopt
  \eql@break@cr@test@setopt
  \eql@break@amp@test@setopt
}
\let\eql@equations@env@modifier\eql@false
\let\eql@equations@sqr@modifier\eql@true
\let\eql@equations@ang@modifier\eql@true
\let\eql@equations@end@modifier\eql@true
\eql@equations@test@setall
\eql@multi@cr@test@setall
\eql@columns@amp@test@setall
\eql@break@cr@test@setall
\eql@break@amp@test@setall
\eql@break@test@setopt
\let\eql@multi@cr@relnext\eql@true

\def\eql@box@nomodifier{%
  \eql@box@test@setopt
  \let\eql@box@end@modifier\eql@false
  \eql@box@cr@test@setopt
  \eql@box@amp@test@setopt
}
\let\eql@box@env@modifier\eql@false
\let\eql@box@ang@modifier\eql@true
\let\eql@box@end@modifier\eql@true
\eql@box@test@setall
\eql@box@cr@test@setall
\eql@box@amp@test@setall

\eql@define@key{setup}{modifier}[true]{%
  \eql@decide@if{#3}{#2}{#1}%
    {\let\eql@equations@env@modifier\eql@true
     \let\eql@equations@sqr@modifier\eql@true
     \let\eql@equations@ang@modifier\eql@true
     \let\eql@box@env@modifier\eql@true
     \let\eql@box@ang@modifier\eql@true
     \eql@multi@cr@test@setall\eql@break@cr@test@setall
     \eql@columns@amp@test@setall\eql@break@amp@test@setall
     \eql@box@cr@test@setall\eql@box@amp@test@setall
     \eql@break@test@setall}%
    {\let\eql@equations@env@modifier\eql@false
     \let\eql@equations@sqr@modifier\eql@false
     \let\eql@equations@ang@modifier\eql@false
     \let\eql@box@env@modifier\eql@false
     \let\eql@box@ang@modifier\eql@false
     \eql@multi@cr@test@setopt\eql@break@cr@test@setopt
     \eql@columns@amp@test@setopt\eql@break@amp@test@setopt
     \eql@box@cr@test@setopt\eql@box@amp@test@setopt
     \eql@break@test@setopt}}
\eql@define@key{setup}{modifierenv}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@env@modifier
  \eql@decide@bool{#3}{#2}{#1}\eql@box@env@modifier}
\eql@define@key{setup}{modifiersqrang}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@sqr@modifier
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@ang@modifier
  \eql@decide@bool{#3}{#2}{#1}\eql@box@ang@modifier}
\eql@define@key{setup}{modifierend}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@end@modifier
  \eql@decide@bool{#3}{#2}{#1}\eql@box@end@modifier}
\eql@define@key{setup}{modifiereqncr}[true]{\eql@decide@if{#3}{#2}{#1}%
    {\eql@multi@cr@test@setall\eql@break@cr@test@setall
     \eql@box@cr@test@setall}%
    {\eql@multi@cr@test@setopt\eql@break@cr@test@setopt
     \eql@box@cr@test@setopt}}
\eql@define@key{setup}{modifiereqnamp}[true]{\eql@decide@if{#3}{#2}{#1}%
    {\eql@columns@amp@test@setall\eql@break@amp@test@setall
     \eql@box@amp@test@setall}%
    {\eql@columns@amp@test@setopt\eql@break@amp@test@setopt
     \eql@box@amp@test@setopt}}
\eql@define@key{setup}{modifierbreak}[true]{\eql@decide@if{#3}{#2}{#1}%
    \eql@break@test@setall\eql@break@test@setopt}
\eql@define@key{setup}{modifierwarning}[all]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@parseopt@warn@env\@empty
      \let\eql@parseopt@warn@cr\@empty}},%
    {{env,-}{\let\eql@parseopt@warn@env\eql@warn@parseopt
      \let\eql@parseopt@warn@cr\@empty}},%
    {{all,\eql@decide@true}{\let\eql@parseopt@warn@env\eql@warn@parseopt
      \let\eql@parseopt@warn@cr\eql@warn@parseopt}},%
    {{verbose,+}{\let\eql@parseopt@warn@env\eql@warn@parseopt@verbose
      \let\eql@parseopt@warn@cr\eql@warn@parseopt@verbose}}}}

\eql@define@key{setup}{modifiereqn}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@env@modifier
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@sqr@modifier
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@ang@modifier}
\eql@define@key{setup}{modifiereqnenv}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@env@modifier}
\eql@define@key{setup}{modifiereqnsqr}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@sqr@modifier}
\eql@define@key{setup}{modifiereqnang}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@ang@modifier}
\eql@define@key{setup}{modifiereqnend}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@end@modifier}
\eql@define@key{setup}{modifiereqncr}[true]{\eql@decide@if{#3}{#2}{#1}%
    {\eql@multi@cr@test@setall\eql@break@cr@test@setall}%
    {\eql@multi@cr@test@setopt\eql@break@cr@test@setopt}}
\eql@define@key{setup}{modifiereqnamp}[true]{\eql@decide@if{#3}{#2}{#1}%
    {\eql@columns@amp@test@setall\eql@break@amp@test@setall}%
    {\eql@columns@amp@test@setopt\eql@break@amp@test@setopt}}

\eql@define@key{setup}{modifierbox}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@box@env@modifier
  \eql@decide@bool{#3}{#2}{#1}\eql@box@ang@modifier}
\eql@define@key{setup}{modifierboxenv}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@box@env@modifier}
\eql@define@key{setup}{modifierboxang}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@box@ang@modifier}
\eql@define@key{setup}{modifierboxend}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@box@end@modifier}
\eql@define@key{setup}{modifierboxcr}[true]{%
  \eql@decide@if{#3}{#2}{#1}\eql@box@cr@test@setall\eql@box@cr@test@setopt}
\eql@define@key{setup}{modifierboxamp}[true]{%
  \eql@decide@if{#3}{#2}{#1}\eql@box@amp@test@setall\eql@box@amp@test@setopt}
```

```
\eql@define@key{equations}{modifierend}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@equations@end@modifier}
\eql@define@key{equations}{modifiercr}[true]{\eql@decide@if{#3}{#2}{#1}%
    {\eql@multi@cr@test@setall\eql@break@cr@test@setall}%
    {\eql@multi@cr@test@setopt\eql@break@cr@test@setopt}}
\eql@define@key{equations}{modifieramp}[true]{\eql@decide@if{#3}{#2}{#1}%
    {\eql@columns@amp@test@setall\eql@break@amp@test@setall}%
    {\eql@columns@amp@test@setopt\eql@break@amp@test@setopt}}
\eql@define@key{setup,equations}{crrelnext}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@multi@cr@relnext}
```

```
\eql@define@key{box}{modifier}[true]{\eql@decide@if{#3}{#2}{#1}%
    {\eql@box@cr@test@setall\eql@box@amp@test@setall
     \let\eql@box@end@modifier\eql@true}%
    {\eql@box@cr@test@setopt\eql@box@amp@test@setopt
     \let\eql@box@end@modifier\eql@false}}
\eql@define@key{box}{modifierend}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@box@end@modifier}
\eql@define@key{box}{modifiercr}[true]{%
  \eql@decide@if{#3}{#2}{#1}\eql@box@cr@test@setall\eql@box@cr@test@setopt}
\eql@define@key{box}{modifieramp}[true]{%
  \eql@decide@if{#3}{#2}{#1}\eql@box@amp@test@setall\eql@box@amp@test@setopt}
```

Vertical Spacing. Settings concerning the spacing of lines: **TODO:** set at end of env only!

```
\eql@define@key\eql@keyall{spread}{%
  \let\eql@spread@reset\eql@false\def\eql@spread@val{#1}}
\eql@define@key\eql@keyall{spread*}[0pt]{%
  \let\eql@spread@reset\eql@true\def\eql@spread@val{#1}}
\eql@define@key\eql@keyall{strut}[true]{\eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@strut@cell\relax\let\eql@strut@tag\relax}},%
    {{cell}{\let\eql@strut@cell\eql@strut\let\eql@strut@tag\relax}},%
    {{tag}{\let\eql@strut@cell\relax\let\eql@strut@tag\eql@strut}},%
    {\eql@decide@true
      {\let\eql@strut@cell\eql@strut\let\eql@strut@tag\eql@strut}}}}
\eql@define@key{setup}{strutdepth}{\def\eql@strut@depth{#1}}
```

Settings to specify the apparent height and depth of equations:

```
\eql@define@key\eql@keyall{displayheight}[strut]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@display@height\@undefined}},%
    {{strut}{\def\eql@display@height{\ht\eql@strutbox@}}},%
    {\relax{\def\eql@display@height{#1}}}}}
\eql@define@key\eql@keyall{displaydepth}[strut]{%
  \eql@decide@select{#3}{#2}{#1}{%
```

```
    {\eql@decide@false{\let\eql@display@depth\@undefined}},%
    {{strut}{\def\eql@display@depth{\dp\eql@strutbox@}}},%
    {\relax{\def\eql@display@depth{#1}}}}}
```

Settings concerning page breaks:

```
\eql@define@key{equations}{prebreak}[4]{\eql@decide@select{#3}{#2}{#1}{%
    {{force,4,\eql@decide@true}{\eql@displaybreak@pre4}},%
    {{high,3}{\eql@displaybreak@pre3}},%
    {{med,medium,2}{\eql@displaybreak@pre2}},%
    {{low,1}{\eql@displaybreak@pre1}},%
    {{0,\eql@decide@false}{\eql@displaybreak@pre0}},%
    {{default,inherit,-1}{\eql@displaybreak@pre\m@ne}}}}
\eql@define@key{equations}{postbreak}[4]{\eql@decide@select{#3}{#2}{#1}{%
    {{force,4,\eql@decide@true}{\eql@displaybreak@post4}},%
    {{high,3}{\eql@displaybreak@post3}},%
    {{med,medium,2}{\eql@displaybreak@post2}},%
    {{low,1}{\eql@displaybreak@post1}},%
    {{0,\eql@decide@false}{\eql@displaybreak@post0}},%
    {{default,inherit,-1}{\eql@displaybreak@post\m@ne}}}}
\eql@define@key{equations,setup}{allowbreaks,allowdisplaybreaks}[4]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{full,4}{\eql@displaybreak@inter4}},%
    {{high,3}{\eql@displaybreak@inter3}},%
    {{med,medium,2}{\eql@displaybreak@inter2}},%
    {{low,1}{\eql@displaybreak@inter1}},%
    {{0,\eql@decide@false}{\eql@displaybreak@inter\z@}}}}
\eql@define@key{equations}{prepenalty}{%
  \eql@displaybreak@prepen@\numexpr#1\relax}
\eql@define@key{equations}{postpenalty}{%
  \eql@displaybreak@postpen@\numexpr#1\relax}
\eql@define@key{equations,setup}{interpenalty}{%
  \interdisplaylinepenalty\numexpr#1\relax}
```

TODO: describe

```
\eql@define@key{control}{vspace}[]{\eql@vspace@add{#1}}
\eql@define@key{control}{vspace*}[]{\eql@vspace@addfixedbefore{#1}}
\eql@define@key{control}{vspace!}[]{\eql@vspace@addfixedafter{#1}}
\eql@define@key{control}{break}[4]{\eql@displaybreak@level[{#1}]}
\eql@define@key{control}{penalty}[]{\eql@displaybreak@star{#1}}
```

Override vertical spacing situation: **TODO:** short should just apply to above?! or as far as short would apply...

```
\eql@define@key{equations}{noskip}[both]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@above{5}}%
    {\def\eql@skip@force@below{5}}}
\eql@define@key{equations}{short}[above]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@above{1}}%
    {\def\eql@skip@force@below{1}}}
\eql@define@key{equations}{long}[both]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@above{0}}%
    {\def\eql@skip@force@below{0}}}
\eql@define@key{equations}{medskip}[both]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@above{6}}%
    {\def\eql@skip@force@below{6}}}
\eql@define@key{equations}{par}[par]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{default,\eql@decide@false}{\let\eql@skip@force@leave\@undefined}},%
    {{cont,hmode}{\let\eql@skip@force@leave\z@}},%
    {{par,vmode}{\let\eql@skip@force@leave\@ne
      \ifdefined\eql@skip@force@below\else
        \def\eql@skip@force@below{3}%
      \fi}},%
    {{top}{\let\eql@skip@force@leave\tw@
      \ifdefined\eql@skip@force@below\else
        \def\eql@skip@force@below{4}%
      \fi}}}}
```

Specify vertical spacing explicitly:

```
\eql@define@key{equations}{skip}{%
  \def\eql@skip@force@above{7}%
  \def\eql@skip@custom@above{#1}%
  \let\eql@skip@force@below\eql@skip@force@above
  \let\eql@skip@custom@below\eql@skip@custom@above}
\eql@define@key{equations}{aboveskip}{%
  \def\eql@skip@force@above{7}%
  \def\eql@skip@custom@above{#1}}
\eql@define@key{equations}{belowskip}{%
  \def\eql@skip@force@below{7}%
  \def\eql@skip@custom@below{#1}}
\eql@define@key{equations}{abovespace}{%
  \advance\eql@abovespace@\glueexpr#1\relax}
\eql@define@key{equations}{belowspace}{%
  \advance\eql@belowspace@\glueexpr#1\relax}
```

Vertical spacing for `intertext`:

```
\eql@define@key{intertext}{skip}{%
  \def\eql@skip@force@above{7}%
  \def\eql@skip@custom@above{#1}%
  \let\eql@skip@force@below\eql@skip@force@above
  \let\eql@skip@custom@below\eql@skip@custom@above}
\eql@define@key{intertext}{aboveskip}{%
  \def\eql@skip@force@below{7}%
  \def\eql@skip@custom@below{#1}}
\eql@define@key{intertext}{belowskip}{%
  \def\eql@skip@force@above{7}%
  \def\eql@skip@custom@above{#1}}
\eql@define@key{intertext}{noskip}[both]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@below{5}}%
    {\def\eql@skip@force@above{5}}}
\eql@define@key{intertext}{short}[both]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@below{1}}%
    {\def\eql@skip@force@above{1}}}
\eql@define@key{intertext}{long}[both]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@below{0}}%
    {\def\eql@skip@force@above{0}}}
\eql@define@key{intertext}{medskip}[both]{%
  \eql@decide@abovebelow{#3}{#2}{#1}%
    {\def\eql@skip@force@below{6}}%
    {\def\eql@skip@force@above{6}}}
```

Configure general vertical spacing behaviour for various situations:

```
\eql@define@key{setup}{skip,longskip}{%
  \abovedisplayskip\glueexpr#1\relax
  \belowdisplayskip\abovedisplayskip
  \def\eql@skip@long@above{#1}%
  \let\eql@skip@long@below\eql@skip@long@above}
\eql@define@key{setup}{aboveskip,abovelongskip}{%
  \abovedisplayskip\glueexpr#1\relax
  \def\eql@skip@long@above{#1}}
\eql@define@key{setup}{belowskip,belowlongskip}{%
  \belowdisplayskip\glueexpr#1\relax
  \def\eql@skip@long@below{#1}}
\eql@define@key{setup}{aboveshortskip}{%
  \abovedisplayshortskip\glueexpr#1\relax
  \def\eql@skip@short@above{#1}}
\eql@define@key{setup}{belowshortskip}{%
  \belowdisplayshortskip\glueexpr#1\relax
  \def\eql@skip@short@below{#1}}
\eql@define@key{setup}{tagskip}{%
  \def\eql@skip@tag@above{#1}%
  \let\eql@skip@tag@below\eql@skip@tag@above}
\eql@define@key{setup}{abovetagskip}{%
  \def\eql@skip@tag@above{#1}}
\eql@define@key{setup}{belowtagskip}{%
  \def\eql@skip@tag@below{#1}}
\eql@define@key{setup}{medskip}{%
  \def\eql@skip@med@above{#1}%
  \let\eql@skip@med@below\eql@skip@med@above}
\eql@define@key{setup}{abovemedskip}{%
  \def\eql@skip@med@above{#1}}
\eql@define@key{setup}{belowmedskip}{%
  \def\eql@skip@med@below{#1}}
\eql@define@key{setup}{abovetopskip}{%
  \def\eql@skip@top@above{#1}}
\eql@define@key{setup}{belowtopskip}{%
  \def\eql@skip@top@below{#1}}
\eql@define@key{setup}{aboveparskip}{%
  \def\eql@skip@par@above{#1}}
\eql@define@key{setup}{belowparskip}{%
  \def\eql@skip@par@below{#1}}
\eql@define@key{setup}{abovecontskip}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{hide}{\def\eql@skip@cont@above{\eql@spread@val-\eql@skip@long@below}}},%
    {\relax{\def\eql@skip@cont@above{#1}}}}}
\eql@define@key{setup}{belowcontskip}{%
  \def\eql@skip@cont@below{#1}}
\eql@define@key{setup}{shortmode}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{\eql@decide@false,never}{\def\eql@skip@mode@short{0}}},%
    {{above,neverbelow,belowoff}{\def\eql@skip@mode@short{1}}},%
    {{belowone,belowsingle}{\def\eql@skip@mode@short{2}}},%
    {{belowall,always,on}{\def\eql@skip@mode@short{3}}}}}
\eql@define@key{setup}{abovecontmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@cont@above}
\eql@define@key{setup}{belowcontmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@cont@below}
```

```
\eql@define@key{setup}{aboveparmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@par@above}
\eql@define@key{setup}{belowparmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@par@below}
\eql@define@key{setup}{abovetopmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@top@above}
\eql@define@key{setup}{belowtopmode}{%
  \eql@decide@situation{#3}{#2}{#1}\eql@skip@mode@top@below}
```

Labels and Tag Declaration. Specify label and tag for equations and subequations:

```
\eql@define@key{equations,subequations}{label}{\eql@tags@addblock@label{#1}}
\eql@define@key{equations,subequations}{labelname}{\eql@tags@addblock@name{#1}}
\eql@define@key{equations,subequations}{tag}{\eql@tags@addblock@tag{#1}}
\eql@define@key{equations,subequations}{tag*}{%
  \eql@tags@addblock@tagform@off\eql@tags@addblock@tag{#1}}
\eql@define@key{equations,subequations}{taglabel}{\eql@tags@addblock@ref{#1}}
```

TODO: describe

```
\eql@define@key{control}{label}{\eql@tags@add@label{#1}}
\eql@define@key{control}{labelname}{\eql@tags@add@name{#1}}
\eql@define@key{control}{tag}{\eql@tags@add@tag{#1}}
\eql@define@key{control}{tag*}{\eql@tags@add@tagform@off\eql@tags@add@tag{#1}}
\eql@define@key{control}{taglabel}{\eql@tags@add@ref{#1}}
\eql@define@key{control}{shifttag}{\eql@tags@add@raiseshift{#1}}
\eql@define@key{control}{smashtag}{\eql@tags@add@raisesmash{#1}}
\eql@define@key{control}{pushtag}[]{\eql@tags@add@forceraise}
```

TODO: describe

```
\eql@define@key{setup}{labelname}{\protected@edef\eql@tags@name@generic{#1}}
\eql@define@key{setup}{autolabel}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@tags@autolabel}
\eql@define@key{setup}{autotag}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@tags@autotag}
```

Tag Spacing. Configure horizontal spacing for equation tags:

```
\eql@define@key{equations,setup}{tagmargin}[auto]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{auto,\eql@decide@false}{\let\eql@tagmargin@val\@undefined}},%
    {\relax{\def\eql@tagmargin@val{#1}}}}}
\eql@define@key{equations,setup}{tagmargin*}{%
  \settowidth\dimen@{#1}\edef\eql@tagmargin@val{\the\dimen@}}
\eql@define@key{equations,setup}{tagmarginratio}{%
  \eql@tagmargin@ratio@\dimexpr#1pt\relax}
\eql@define@key{equations,setup}{tagmarginthreshold}{%
  \def\eql@tagmargin@threshold{#1}}
\eql@define@key{equations,setup}{mintagsep}{\def\eql@tagsepmin@val{#1}}
\eql@define@key{equations,setup}{mintagwidth}{%
  \settowidth\dimen@{#1}\edef\eql@tagsepmin@val{\the\dimen@}}
\eql@define@key{equations,setup}{mintagwidth*}{%
  \settowidth\eql@tagwidthmin@{#1}}
\eql@define@key{equations,setup}{tagsnap}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@tagpos@snap\z@}},%
    {\relax{\def\eql@tagpos@snap{#1}}}}}
```

Tag Layout. Configure methods to declare equation tag layout:

```
\eql@define@key{equations,setup}{tagbox,taglayout}{%
  \eql@tags@taglayout@set{#1}}
\eql@define@key{equations,setup}{tagbox*,taglayout*}{%
  \eql@tags@taglayout@set@direct{#1}}
\eql@define@key{equations,setup}{tagform}{%
  \eql@tags@tagform@set#1}
\eql@define@key{equations,setup}{tagform*}{%
  \eql@tags@tagform@set@direct{#1}}
\eql@define@key{equations,setup}{subeqtemplate}{%
  \eql@subequations@template@set{#1}}
```

```
\eql@define@key{control}{tagbox,taglayout}{%
  \global\eql@append\eql@tags@container{\eql@tags@taglayout@set{#1}}}
\eql@define@key{control}{tagbox*,taglayout*}{%
  \global\eql@append\eql@tags@container{\eql@tags@taglayout@set@direct{#1}}}
\eql@define@key{control}{tagform}{%
  \global\eql@append\eql@tags@container{\eql@tags@tagform@set#1}}
\eql@define@key{control}{tagform*}[{####1}]{%
  \global\eql@append\eql@tags@container{\eql@tags@tagform@set@direct{#1}}}
```

Equation Numbering. Configure equation numbering schemes:

```
\eql@define@key{equations,setup}{numberline,number,num,numline,n}[all]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{\eql@decide@false,0,*}{\let\eql@numbering@active\eql@false}},%
    {{\eql@decide@true,!}{\let\eql@numbering@active\eql@true}},%
    {{none,n,-}{\let\eql@numbering@mode\eql@numbering@mode@multi
        \let\eql@numbering@active\eql@false}},%
    {{single,1}{\let\eql@numbering@mode\eql@numbering@mode@single
        \let\eql@numbering@active\eql@true}},%
    {{multi,@}{\let\eql@numbering@mode\eql@numbering@mode@multi
        \let\eql@numbering@active\eql@true}},%
    {\relax{\eql@numbering@set{#1}}}}}
\eql@define@key{equations,setup}{nonumber,nn,*}[]{%
  \let\eql@numbering@active\eql@false}
\eql@define@key{equations,setup}{donumber,dn,!}[]{%
  \let\eql@numbering@active\eql@true}
\eql@define@key{equations,setup}{tagsleft,leqno}[]{%
  \let\eql@tagsleft\eql@true}
\eql@define@key{equations,setup}{tagsright,reqno}[]{%
  \let\eql@tagsleft\eql@false}
\eql@define@key{equations,setup}{tags,eqno}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{right,r}{\let\eql@tagsleft\eql@false}},%
    {{left,l}{\let\eql@tagsleft\eql@true}}}}
\eql@define@key{equations,setup}{evadetag,avoidtag}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@numbering@best@auto}
\eql@define@key{equations,setup}{tagbetween}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@tagpos@doconvert}
```

TODO: describe

```
\eql@define@key{control}{nonumber,nn,*}[]{\global\@eqnswfalse}
\eql@define@key{control}{donumber,dn,!}[]{\global\@eqnswtrue}
\eql@define@key{control}{numberhere}[]{\eql@numberhere}
\eql@define@key{control}{numbernext}[]{\eql@numbernext}
```

Horizontal Layout. Configure horizontal alignment mode and margin for left alignment:

```
\eql@define@key{equations,setup}{layout}{\eql@decide@select{#3}{#2}{#1}{%
  {{center,c}{\let\eql@layoutleft\eql@false}},%
  {{left,l}{\let\eql@layoutleft\eql@true}}}}
\eql@define@key{equations,setup}{center}[]{%
  \let\eql@layoutleft\eql@false}
\eql@define@key{equations,setup}{flushleft,left}[]{%
  \let\eql@layoutleft\eql@true}
\eql@define@key{equations,setup}{leftmargin}{\def\eql@layoutleftmargin{#1}}
\eql@define@key{equations,setup}{leftmargin*}{%
  \settowidth\dimen@{#1}\edef\eql@layoutleftmargin{\the\dimen@}}
\eql@define@key{equations,setup}{minleftmargin}{%
  \def\eql@layoutleftmarginmin{#1}}
\eql@define@key{equations,setup}{maxleftmargin}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\def\eql@layoutleftmarginmax{.5\maxdimen}}},%
    {\relax{\def\eql@layoutleftmarginmax{#1}}}}}
```

```
\eql@define@key{equations,box}{margin}{%
  \def\eql@display@marginleft{#1}\def\eql@display@marginright{#1}}
\eql@define@key{equations,box}{marginleft}{\def\eql@display@marginleft{#1}}
\eql@define@key{equations,box}{marginright}{\def\eql@display@marginright{#1}}
\eql@define@key{equations}{linewidth,width}{\def\eql@display@linewidth{#1}}
```

Horizontal Spacing and Columns. Configure column spacing and compression threshold:

```
\eql@define@key{equations,setup}{alignshrink}{\eql@decide@select{#3}{#2}{#1}{%
    {{max,full,4}{\eql@alignbadness@\inf@bad}},%
    {{high,3}{\eql@alignbadness@54\relax}},%
    {{med,medium,2}{\eql@alignbadness@18\relax}},%
    {{low,1}{\eql@alignbadness@6\relax}},%
    {{0,\eql@decide@false}{\eql@alignbadness@\z@}}}}
\eql@define@key{equations,setup}{tagshrink}{\eql@decide@select{#3}{#2}{#1}{%
    {{max,full,4}{\eql@tagbadness@\inf@bad}},%
    {{high,3}{\eql@tagbadness@54\relax}},%
    {{med,medium,2}{\eql@tagbadness@18\relax}},%
    {{low,1}{\eql@tagbadness@6\relax}},%
    {{0,\eql@decide@false}{\eql@tagbadness@\z@}}}}
\eql@define@key{equations,setup}{alignbadness}{%
  \eql@alignbadness@\numexpr#1\relax}
\eql@define@key{equations,setup}{tagbadness}{%
  \eql@tagbadness@\numexpr#1\relax}
\eql@define@key{equations,setup}{mincolsep}{\eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@colsepmin@val{0pt}}},%
    {\relax{\def\eql@colsepmin@val{#1}}}}}
\eql@define@key{equations,setup}{maxcolsep}{\eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\def\eql@colsepmax@val{.5\maxdimen}}},%
    {0{\def\eql@colsepmax@val{0pt}}},%
    {\relax{\def\eql@colsepmax@val{#1}}}}}
\eql@define@key{equations,setup}{fulllength}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@columns@fulllength}
```

TODO: is boxcolsep vs breakcolsep okay??!

```
\eql@define@key{equations,setup}{linesep}{\eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@break@line@sep{0pt}}},%
    {\relax{\def\eql@break@line@sep{#1}}}}
\eql@define@key{equations,setup}{linesep*}{\eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@break@line@shortsep{0pt}}},%
    {\relax{\def\eql@break@line@shortsep{#1}}}}}
\eql@define@key{box,setup}{colsep}{\eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@box@colsep{0pt}}},%
    {{short,*}{\def\eql@box@colsep{\eql@box@shortsep}}},%
    {\relax{\def\eql@box@colsep{#1}}}}%
  \let\eql@break@col@sep\eql@box@colsep}
\eql@define@key{equations}{colsep}{\eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@break@col@sep{0pt}}},%
    {\relax{\def\eql@break@col@sep{#1}}}}%
  \let\eql@colsepmin@val\eql@box@colsep
  \let\eql@colsepmax@val\eql@box@colsep
  \let\eql@box@colsep\eql@break@col@sep}
\eql@define@key{equations,setup}{colsep*}{\eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@break@col@shortsep{0pt}}},%
    {\relax{\def\eql@break@col@shortsep{#1}}}}}
\eql@define@key{box,setup}{colsep*}{\eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@box@shortsep{0pt}}},%
    {\relax{\def\eql@box@shortsep{#1}}}}}
\eql@define@key{box,setup}{condsep}{\eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@box@condsep{0pt}}},%
    {{short,*}{\def\eql@box@condsep{\eql@box@shortsep}}},%
    {\relax{\def\eql@box@condsep{#1}}}}}
\eql@define@key{box,setup}{matrixsep}{\eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@box@matrixsep{0pt}}},%
    {{short,*}{\def\eql@box@matrixsep{\eql@box@shortsep}}},%
    {\relax{\def\eql@box@matrixsep{#1}}}}}
```

Horizontal Shape. Configure horizontal alignment schemes:

```
\eql@define@key\eql@keyall{shape}[default]{\eql@shape@set{#1}}
\eql@define@key\eql@keyall{padding,pad}[indent]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{max}{\let\eql@paddingleft@val\@undefined}},%
    {{indent}{\def\eql@paddingleft@val{\eql@indent@val}}},%
    {{0,\eql@decide@false}{\def\eql@paddingleft@val{0pt}}},%
    {\relax{\def\eql@paddingleft@val{#1}}}}%
  \let\eql@paddingright@val\eql@paddingleft@val}
\eql@define@key\eql@keyall{padleft}[indent]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{max}{\let\eql@paddingleft@val\@undefined}},%
    {{indent}{\def\eql@paddingleft@val{\eql@indent@val}}},%
    {{0,\eql@decide@false}{\def\eql@paddingleft@val{0pt}}},%
    {\relax{\def\eql@paddingleft@val{#1}}}}}
\eql@define@key\eql@keyall{padright}[indent]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{max}{\let\eql@paddingright@val\@undefined}},%
    {{indent}{\def\eql@paddingright@val{\eql@indent@val}}},%
    {{0,\eql@decide@false}{\def\eql@paddingright@val{0pt}}},%
    {\relax{\def\eql@paddingright@val{#1}}}}}
\eql@define@key\eql@keyall{indent}[2em]{%
  \def\eql@indent@val{#1}}
```

TODO: describe

```
\def\eql@shape@rel{\eqncontrol{align=left}}
\def\eql@shape@cont{\eqncontrol{align=left,shift=*}}
```

```
\eql@define@key\eql@keyall{shaperel}[]{\def\eql@shape@rel{#1}}
\eql@define@key\eql@keyall{shapecont}[]{\def\eql@shape@cont{#1}}
```

TODO: describe

```
\eql@define@key{control}{align}[]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{l,left}{\global\eql@append\eql@cell@container{\eql@shape@pos@\z@}}},%
    {{c,center}{\global\eql@append\eql@cell@container{\eql@shape@pos@\@ne}}},%
    {{r,right}{\global\eql@append\eql@cell@container{\eql@shape@pos@\tw@}}}}}
\eql@define@key{control}{shift,shiftto}[]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{*,indent}{\eql@shape@alignamount@set{\eql@indent@}}},%
    {{!,outdent}{\eql@shape@alignamount@set{-\eql@indent@}}},%
    {\relax{\eql@shape@alignamount@set{#1}}}}}
\eql@define@key{control}{shift*,shiftby}[]{\eql@shape@alignamount@add{#1}}
```

Math Classes at Alignment. Configure math classes at alignment marker:

```
\eql@define@key\eql@keyall{classout}{\eql@class@innerleft@set{#1}}
\eql@define@key\eql@keyall{classin}{\eql@class@innerright@set{#1}}
\eql@define@key\eql@keyall{classlead,classin*}{\eql@class@innerlead@set{#1}}
\eql@define@key\eql@keyall{rel}{\def\eql@class@rel@symb{#1}}
\eql@define@key\eql@keyall{classbreak}{\eql@class@rel@break@set{#1}}
\eql@define@key\eql@keyall{classstart}{\eql@class@rel@start@set{#1}}
\eql@define@key\eql@keyall{classbreakskip}[adjust]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{0,\eql@decide@false}{\def\eql@class@rel@break@skip{0mu}}},%
    {{adjust,*}{\def\eql@class@rel@break@skip{\thickmuskip-\medmuskip}}},%
    {\relax{\def\eql@class@rel@break@skip{#1}}}}}
\eql@define@key{control}{rel}[\eql@class@rel@symb]{\eql@class@rel@make{#1}}
\eql@define@key{control}{rel;}[]{\eql@class@rel@make{}}
\eql@define@key{control}{rel*}[]{\eql@class@rel@make{}}
\eql@define@key\eql@keyall{ampeq}[]{\eql@class@ampeq}
\eql@define@key\eql@keyall{eqamp}[]{\eql@class@eqamp}
\eql@define@key\eql@keyall{class}{\eql@decide@select{#3}{#2}{#1}{%
  {{ampeq,amprel,eqafter,beforerel}\eql@class@ampeq},%
  {{eqamp,relamp,eqbefore,afterrel}\eql@class@eqamp}}}
```

Math Styles. Configure math classes at alignment marker:

```
\eql@define@key\eql@keyall{style}[display]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{text}{\let\eql@mathstyle\@empty}},%
    {{display}{\let\eql@mathstyle\displaystyle}}}}
\eql@define@key{setup}{casesstyle}[display]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@cases@mathstyle\eql@false}},%
    {{text}{\let\eql@cases@mathstyle\@empty}},%
    {{display}{\let\eql@cases@mathstyle\displaystyle}}}}
```

Punctuation. Configure punctuation defaults: **TODO:** describe

```
\def\eql@punct@all#1#2#3#4#5\eql@punct@end{%
  \def\eql@tmp{#4}\def\eql@tmpa{1}%
  \ifx\eql@tmp\eql@tmpa
    \ifnum#5=1111\relax
      \eql@punct@set\eql@punct@col{#1}%
      \eql@punct@set\eql@punct@line{#2}%
      \eql@punct@set\eql@punct@block{#3}%
    \else\ifnum#5=111\relax
      \eql@punct@set\eql@punct@line{#1}%
      \eql@punct@set\eql@punct@block{#2}%
    \else\ifnum#5=11\relax
      \eql@punct@set\eql@punct@block{#1}%
    \else
      \eql@punct@clear
    \fi\fi\fi
  \else
    \eql@error{Too many arguments to punctall}%
  \fi
}
```

TODO: describe

```
\eql@define@key\eql@keyall{punctsep}[\,]{\def\eql@punct@sep{#1}}
\eql@define@key\eql@keyall{punctclass}[\mathclose{}]{\def\eql@punct@class{#1}}
\eql@define@key\eql@keyall{punct}[.]{\eql@punct@set\eql@punct@block{#1}}
\eql@define@key\eql@keyall{punct*}[]{\eql@punct@set\eql@punct@block\relax}
\eql@define@key\eql@keyall{punctline}[,]{\eql@punct@set\eql@punct@line{#1}}
\eql@define@key\eql@keyall{punctline*}[]{\eql@punct@set\eql@punct@line\relax}
\eql@define@key\eql@keyall{punctcol}[,]{\eql@punct@set\eql@punct@col{#1}}
\eql@define@key\eql@keyall{punctcol*}[]{\eql@punct@set\eql@punct@col\relax}
\eql@define@key{box}{punctcases}[,]{\eql@punct@set\eql@punct@col{#1}}
\eql@define@key{box}{punctcases*}[]{\eql@punct@set\eql@punct@col\relax}
\eql@define@key{setup}{punctcases}[,]{\eql@punct@set\eql@punct@cases{#1}}
\eql@define@key{setup}{punctcases*}[]{\eql@punct@set\eql@punct@cases\relax}
\eql@define@key\eql@keyall{punctall}[,,.]{\eql@punct@all#111111\eql@punct@end}
\eql@define@key{box}{puncttERM}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@box@punct@term}
```

```
\eql@define@key{control}{punctsep}[\,]{\def\eql@punct@sep{#1}}
\eql@define@key{control}{setpunct}[.]{\eql@punct@set\eql@punct@next{#1}}
\eql@define@key{control}{setpunct}[.]{\eql@punct@set\eql@punct@next{#1}}
\eql@define@key{control}{setpunct*}[]{\let\eql@punct@next\relax}
\eql@define@key{control}{punct,punctapply}[\relax]{%
  \eql@punct@set\eql@punct@next{#1}\eql@punct@apply@top}
\eql@define@key{control}{punctline}[]{\eql@punct@print@line}
\eql@define@key{control}{punctcol}[]{\eql@punct@print@col}
```

Frames. **TODO:** describe

```
\eql@define@key{box}{frame}[\fbox]{%
  \def\eql@box@frame{#1}%
  \ifx\eql@box@frame\@empty\let\eql@box@frame\@firstofone\fi}
\eql@define@key{box}{wrap}{\eql@box@wrap#1}
\eql@define@key{box}{delim}[r]{\eql@decide@delim{#3}{#2}{#1}}
\eql@define@key{box}{ldelim}{\eql@box@ldelim#1}
\eql@define@key{box}{rdelim}{\eql@box@rdelim#1}
\eql@define@key{box}{lbrace}[]{\eql@box@ldelim\lbrace}
\eql@define@key{box}{rbrace}[]{\eql@box@rdelim\rbrace}
\eql@define@key{box}{lrbrace,lrbraces}[]{\eql@box@delim\lbrace\rbrace}
\eql@define@key{box}{braces}[lr]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{\eql@decide@false}{\eql@box@wrap{}{}}},%
    {{l,left}{\eql@box@ldelim\lbrace}},%
    {{r,right}{\eql@box@rdelim\rbrace}},%
    {{\eql@decide@true,lr,both}{\eql@box@delim\lbrace\rbrace}}}
```

TODO: describe

```
\eql@define@key{control}{framecell}[\fbox]{%
  \global\eql@append\eql@cell@container{\def\eql@frame@cmd{#1}}}
\eql@define@key{control}{frametag}[\fbox]{%
  \global\eql@append\eql@tags@container{\def\eql@tags@frame@cmd{#1}}}
```

Alternative Content Description. Alternative content description for accessibility or documentation purposes: **TODO:** implement in PDF tagging

```
\eql@define@key{equations,box}{alt}{}
```

Injections.

```
\eql@define@key{control}{inject}{%
  \global\eql@append\eql@interline@container{%
    \eql@append\eql@display@injectbefore{#1}}}
\eql@define@key{control}{inject*}{%
  \global\eql@append\eql@interline@container{%
    \eql@append\eql@display@injectafter{#1}}}
\eql@define@key{control}{markline}[]{\eql@markline@inject{#1}}
\eql@define@key{control}{markline*}[]{\eql@markline@inject{push,#1}}
\eql@define@key{control}{qed}[]{\eql@markline@inject{qed,#1}}
\eql@define@key{control}{qed*}[]{\eql@markline@inject{qed,push,#1}}
```

TODO: describe

```
\eql@define@key{markline}{pos}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{below,push}{\let\eql@markline@pos\eql@markline@pos@below}},%
    {{baseline}{\let\eql@markline@pos\eql@markline@pos@baseline}},%
    {{bottom}{\let\eql@markline@pos\eql@markline@pos@bottom}}}}
\eql@define@key{markline}{below,push}[]{%
  \let\eql@markline@pos\eql@markline@pos@below}
\eql@define@key{markline}{baseline}[]{%
  \let\eql@markline@pos\eql@markline@pos@baseline}
\eql@define@key{markline}{bottom}[]{%
  \let\eql@markline@pos\eql@markline@pos@bottom}
\eql@define@key{markline}{shift}{\def\eql@markline@shift{#1}}
\eql@define@key{markline}{symbol}{\def\eql@markline@symbol{#1}}
\eql@define@key{markline}{qed}[]{\let\eql@markline@symbol\eql@markline@qed}
\eql@define@key{setup}{marksymbol}{\def\eql@markline@symbol{#1}}
\eql@define@key{setup}{qedsymbol}{\def\eql@markline@qed{#1}}
\eql@define@key{setup}{markpos}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{below}{\let\eql@markline@pos\eql@markline@pos@below}},%
    {{baseline}{\let\eql@markline@pos\eql@markline@pos@baseline}},%
    {{bottom}{\let\eql@markline@pos\eql@markline@pos@bottom}}}}
```

Global Switches. Set global switches:

```
\let\eql@multi@linesfallback\eql@false
\let\eql@scan@par\eql@false
\let\eql@single@cr@mode\eql@false
\let\eql@amp@mode\eql@true
\let\eql@ampproof@active\eql@false
```

```
\let\eql@parseopt@warn@env\eql@warn@parseopt
\let\eql@parseopt@warn@cr\@empty
```

```
\eql@define@key{equations,setup}{linesfallback}[true]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@multi@linesfallback\eql@false}},%
    {{reuse,lean}{\let\eql@multi@linesfallback\z@}},%
    {{measure,full,\eql@decide@true}{\let\eql@multi@linesfallback\eql@true}}}}
\eql@define@key{setup}{ampproof}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@ampproof@active}
\eql@define@key{equations,setup}{equationcr}[true]{%
  \eql@decide@select{#3}{#2}{#1}{%
    {\eql@decide@false{\let\eql@single@cr@mode\eql@false}},%
    {{\eql@decide@true,break}{\let\eql@single@cr@mode\eql@break@cr}},%
    {{error,verbose}{\let\eql@single@cr@mode\eql@single@cr@error}}}}
\eql@define@key\eql@keyall{amp}[true]{%
  \eql@decide@bool{#3}{#2}{#1}{\eql@amp@mode}}
\eql@define@key\eql@keyall{rescan}[true]{%
  \eql@decide@if{#3}{#2}{#1}%
    {\let\eql@scan@body\eql@scan@body@rescan}%
    {\let\eql@scan@body\eql@scan@body@dump}}
\eql@define@key\eql@keyall{scanpar}[true]{%
  \eql@decide@bool{#3}{#2}{#1}\eql@scan@par}
\eql@define@key{setup}{defaults}{%
  \eql@decide@select{#3}{#2}{#1}{%
    {{classic}{\eql@defaults@classic}},%
    {{eqnlines}{\eql@defaults@eqnlines}}}}
\eql@define@key\eql@keyall{verbose}[true]{%
  \eql@decide@if{#3}{#2}{#1}\eql@verbose@on\eql@verbose@off}
```

Package Options. Declare choices available at loading of package only: **TODO:** adjust

```
\let\eql@provide@opt@env\tw@
\let\eql@provide@opt@amsmathpatch\eql@false
\let\eql@provide@opt@backup\eql@false
\let\eql@provide@opt@ang\eql@true
\let\eql@provide@opt@eqref\eql@true
\let\eql@provide@opt@matrix\eql@true
```

```
\eql@define@key{setup}{amsmathends,amsmathpatch}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@amsmathpatch}
\eql@define@key{setup}{backup}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@backup}
\eql@define@key{setup}{env}[equation]{%
  \eql@error@packageoption{#2}%
  \eql@decide@select{#3}{#2}{#1}{%
    {{none,\eql@decide@false}{\let\eql@provide@opt@env\z@}},%
    {{equation,latex}{\let\eql@provide@opt@env\@ne}},%
    {{amsmath,all,\eql@decide@true}{\let\eql@provide@opt@env\tw@}}}}
\eql@define@key{setup}{ang}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@ang}
\eql@define@key{setup}{eqref}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@eqref}
\eql@define@key{setup}{matrix}[true]{%
  \eql@error@packageoption{#2}%
  \eql@decide@bool{#3}{#2}{#1}\eql@provide@opt@matrix}
```

Shortcut Options. **TODO:** describe

```
\def\eql@parseopt@nonumber#1{\eqnaddopt{nonumber}\eql@parseopt@peek}
\def\eql@parseopt@donumber#1{\eqnaddopt{donumber}\eql@parseopt@peek}
\def\eql@parseopt@single#1{\eqnaddopt{single}\eql@parseopt@peek}
\def\eql@parseopt@lines#1{\eqnaddopt{lines}\eql@parseopt@peek}
\def\eql@parseopt@eqamp#1{\eqnaddopt{eqamp}\eql@parseopt@peek}
\def\eql@parseopt@ampeq#1{\eqnaddopt{ampeq}\eql@parseopt@peek}
\def\eql@parseopt@columns#1{\eqnaddopt{columns}\eql@parseopt@peek}
\def\eql@parseopt@transpose#1{\eqnaddopt{columns,transpose}\eql@parseopt@peek}
\def\eql@parseopt@opt[#1]{\eqnaddopt{#1}\eql@parseopt@peek}
\def\eql@parseopt@label#1#2{\eqnaddopt{label={#2}}\eql@parseopt@peek}
\def\eql@parseopt@punctpass{\eql@parseopt@peek`}
\def\eql@parseopt@punctclear#1{\eql@parseopt@peek`~}
\def\eql@parseopt@punctopt#1#2{\eqnaddopt{punctall={#2}}\eql@parseopt@peek}
\def\eql@parseopt@punctnext#1#2{%
  \eql@punct@set\eql@punct@next{#2}\eql@parseopt@peek}
\def\eql@parseopt@punctblock#1#2{%
  \eql@punct@set\eql@punct@block{#2}\eql@parseopt@peek}
\def\eql@parseopt@puncterm#1{\let\eql@punct@term\eql@true\eql@parseopt@peek}
\def\eql@parseopt@relsymb#1{\eql@parseopt@peek?\eql@class@rel@symb}
\def\eql@parseopt@relcont#1{\eql@parseopt@peek?{}}
\def\eql@parseopt@relstart#1{\eql@parseopt@peek?\relax}
\def\eql@parseopt@relord#1{\eql@parseopt@peek?{{}}}
\def\eql@parseopt@vspace[#1]{%
  \advance\eql@vspaceskip@\glueexpr#1\relax\eql@parseopt@peek}
```

16.3 Parameter Presets

The package offers two parameter presets which lead to somewhat different layout. Instead of setting the internal parameters directly, we expose them as public settings so that they are easier to read and such that individual settings can be used to compose own layouts.

`eql@defaults@classic` The preset `classic` aims to reproduce the TeX, LaTeX and amsmath layout closely. These presets mostly use fixed dimensions:

```
\def\eql@defaults@classic{%
  \eqnlineset{numberline=all}%
  \eqnlineset{condsep={1em}}%
  \eqnlineset{matrixsep={2\arraycolsep}}%
  \eqnlineset{mintagsep={.5\fontdimen6\textfont2}}%
  \eqnlineset{maxcolsep=off}%
  \eqnlineset{spread={\jot}}%
  \eqnlineset{tagmargin}%
  \eqnlineset{tagmarginratio=1}%
  \eqnlineset{tagmarginthreshold=0.5}%
  \eqnlineset{leftmargin={\leftmargini}}%
  \eqnlineset{padding=max}%
  \eqnlineset{evadetag=off}%
  \eqnlineset{displayheight=off}%
  \eqnlineset{displaydepth=off}%
  \eqnlineset{shortmode=belowsingle}%
  \eqnlineset{abovecontmode=short}%
  \eqnlineset{belowcontmode=short}%
  \eqnlineset{aboveparmode=long}%
  \eqnlinesset{belowparmode=long}%
  \eqnlinesset{abovetopmode=long}%
  \eqnlinesset{belowtopmode=long}%
  \eqnlinesset{abovelongskip={\abovedisplayskip}}%
  \eqnlinesset{belowlongskip={\belowdisplayskip}}%
  \eqnlinesset{aboveshortskip={\abovedisplayshortskip}}%
  \eqnlinesset{belowshortskip={\belowdisplayshortskip}}%
  \eqnlinesset{abovemedskip={.5\abovedisplayskip}}%
  \eqnlinesset{belowmedskip={.5\belowdisplayskip}}%
  \eqnlinesset{abovecontskip=0pt}%
  \eqnlinesset{belowcontskip=0pt}%
  \eqnlinesset{aboveparskip=0pt}%
  \eqnlinesset{belowparskip=0pt}%
  \eqnlinesset{abovetopskip=0pt}%
  \eqnlinesset{belowtopskip=0pt}%
  \eqnlinesset{abovetagskip=0pt}%
  \eqnlinesset{belowtagskip=0pt}%
  \eqnlinesset{allowbreaks=0}%
  \eqnlinesset{equationcr=off}%
  \eqnlinesset{amp=off}%
  \eqnlinesset{modifier=off}%
  \eqnlinesset{linesfallback=false}%
  \eqnlinesset{casesstyle=text}%
  \eqnlinesset{sqropt=nonumber}%
  \eqnlinesset{angopt=nonumber}%
}
```

values based on 10pt vs 12pt

`ql@defaults@eqnlines`

The (default) preset `eqnlines` implements a layout that scales with the font size by using the units `em` and `\normalbaselineskip` for horizontal and vertical spacing, respectively. It aims to approximately reproduce the `classic` spacing for a 12 pt computer modern font such that 10 pt fonts will lead to slightly reduced spacing. Apart from that, the `eqnlines` setting makes some deliberate layout choices that deviate significantly from `classic` (maximum column separation, no shortening below equations):

```
\def\eql@defaults@eqnlines{%
  \eqnlinesset{numberline=all}%
  \eqnlinesset{condsep=short}%
  \eqnlinesset{matrixsep=short}%
  \eqnlinesset{mintagsep=.5em}%
  \eqnlinesset{maxcolsep=2em}%
  \eqnlinesset{spread={0.2\normalbaselineskip}}%
  \eqnlinesset{tagmargin}%
  \eqnlinesset{tagmarginratio=.334}%
  \eqnlinesset{tagmarginthreshold=0.5}%
  \eqnlinesset{leftmargin={\leftmargini}}%
  \eqnlinesset{padding=0pt}%
  \eqnlinesset{evadetag}%
  \eqnlinesset{displayheight=strut}%
  \eqnlinesset{displaydepth=strut}%
  \eqnlinesset{shortmode=above}%
  \eqnlinesset{abovecontmode=noskip}%
  \eqnlinesset{belowcontmode=long}%
  \eqnlinesset{aboveparmode=long}%
  \eqnlinesset{belowparmode=long}%
  \eqnlinesset{abovetopmode=noskip}%
  \eqnlinesset{belowtopmode=long}%
  \eqnlinesset{longskip={0.75\normalbaselineskip
```

```
    plus 0.25\normalbaselineskip minus 0.4\normalbaselineskip}}%
  \eqnlinesset{aboveshortskip={0.0\normalbaselineskip
    plus 0.25\normalbaselineskip}}%
  \eqnlinesset{belowshortskip={0.0\normalbaselineskip
    plus 0.25\normalbaselineskip}}%
  \eqnlinesset{medskip={0.4\normalbaselineskip
    plus 0.2\normalbaselineskip minus 0.2\normalbaselineskip}}%
  \eqnlinesset{abovecontskip=0pt}%
  \eqnlinesset{belowcontskip=0pt}%
  \eqnlinesset{aboveparskip=0pt}%
  \eqnlinesset{belowparskip=0pt}%
  \eqnlinesset{abovetopskip=0pt}%
  \eqnlinesset{belowtopskip=0pt}%
  \eqnlinesset{abovetagskip={0.25\normalbaselineskip
    minus 0.25\normalbaselineskip}}%
  \eqnlinesset{belowtagskip={0.25\normalbaselineskip
    minus 0.25\normalbaselineskip}}%
  \eqnlinesset{allowbreaks=3}%
  \eqnlinesset{equationcr=break}%
  \eqnlinesset{amp=on}%
  \eqnlinesset{modifier=on,modifierenv=off}%
  \eqnlinesset{linesfallback=true}%
  \eqnlinesset{casesstyle=false}%
  \eqnlinesset{sqropt=}%
  \eqnlinesset{angopt=}%
}
```

16.4 Component Selection

The following routines provide several additional math environments beyond `equations`. They also backup and overwrite the original routines of LaTeX and `amsmath` carefully.

Tools.

`\eql@provide@movecmd`
`\eql@provide@moveenv`
`@provide@undefinecmd`
`@provide@undefineenv`

We introduce a couple of tools to rename and undefine commands and environments:

```
\def\eql@provide@movecmd#1#2{%
  \eql@letcs{#1\expandafter}\csname#2\endcsname
}
\def\eql@provide@moveenv#1#2{%
  \eql@provide@movecmd{#1}{#2}%
  \ifcsname end#2\endcsname
    \eql@provide@movecmd{end#1}{end#2}%
  \fi
}
\def\eql@provide@undefinecmd#1{%
  \eql@letcs{#1}\@undefined
}
\def\eql@provide@undefineenv#1{%
  \eql@provide@undefinecmd{#1}%
  \eql@provide@undefinecmd{end#1}%
}
```

Fix Endings for amsmath Environments. The `amsmath` derived environments forward their ending routines directly to the ending routines for the main environments `gather`, `multline`, `align`, `aligned`. This causes a problem when the main environments are replaced but the derived ones are still used. We fix the potential problem by copying the ending routines of the main environments to the ending routines of the derived environments.

`\eql@amsmath@endfix` Check whether the original forwarding of an ending routine is still in place (other packages or future updates to amsmath might change the behaviour). If so, copy the ending routine into place:

```
\def\eql@amsmath@endfix#1#2{%
  \long\edef\eql@tmpa{\expandafter\noexpand\csname end#2\endcsname}%
  \expandafter\ifx\csname end#1\endcsname\eql@tmpa
    \eql@provide@movecmd{end#1}{end#2}%
  \fi
}
```

`ql@amsmath@fixmatrix`
`amsmath@fixmatrixend` **TODO:** describe

```
\def\eql@amsmath@fixmatrix#1{%
  \expandafter\let\expandafter\eql@tmp\csname#1\endcsname
  \begingroup
    \let\matrix@check\@gobble
    \def\env@matrix{\noexpand\env@matrix}%
    \def\env@cases{\noexpand\env@cases}%
    \global\edef\eql@tmp{\eql@tmp}%
  \endgroup
  \eql@letcs{#1}\eql@tmp
}
\def\eql@amsmath@fixmatrixend#1{%
  \expandafter\let\expandafter\eql@tmp\csname end#1\endcsname
  \begingroup
    \expandafter\def\expandafter\endmatrix\expandafter{%
      \expandafter\unexpanded\expandafter{\endmatrix}}%
    \global\long\edef\eql@tmp{\eql@tmp}%
  \endgroup
  \eql@letcs{end#1}\eql@tmp
}
```

`\eql@amsmath@fixends` Perform the replacement for all amsmath environments whenever amsmath is loaded:

```
\def\eql@amsmath@fixends{%
  \eql@amsmath@after{%
    \eql@amsmath@endfix{flalign}{align}%
    \eql@amsmath@endfix{alignat}{align}%
    \eql@amsmath@endfix{xalignat}{align}%
    \eql@amsmath@endfix{xxalignat}{align}%
    \eql@amsmath@endfix{gather*}{gather}%
    \eql@amsmath@endfix{multline*}{multline}%
    \eql@amsmath@endfix{align*}{align}%
    \eql@amsmath@endfix{flalign*}{align}%
    \eql@amsmath@endfix{alignat*}{align}%
    \eql@amsmath@endfix{xalignat*}{align}%
    \eql@amsmath@endfix{gathered}{aligned}%
    \eql@amsmath@endfix{alignedat}{aligned}%
  }
}
```

`@amsmath@fixmatrices` Perform the replacement for all amsmath environments whenever amsmath is loaded:

```
\def\eql@amsmath@fixmatrices{%
  \eql@amsmath@after{%
    \eql@amsmath@fixmatrix{cases}%
    \eql@amsmath@fixmatrix{matrix}%
    \eql@amsmath@fixmatrix{pmatrix}%
    \eql@amsmath@fixmatrixend{pmatrix}%
    \eql@amsmath@fixmatrixend{bmatrix}%
    \eql@amsmath@fixmatrixend{Bmatrix}%
    \eql@amsmath@fixmatrixend{vmatrix}%
    \eql@amsmath@fixmatrixend{Vmatrix}%
  }
}
```

Backup amsmath Environments. We can backup all `amsmath` environments *env* to `ams`*env* so that they can be used in parallel if needed.

`rovide@backup@amsenv` Copy an `amsmath` environment *env* to `ams`*env* whenever `amsmath` is loaded: **TODO:** describe

```
\def\eql@provide@backup@amsenv#1{%
  \eql@amsmath@after{%
    \eql@provide@moveenv{ams#1}{#1}%
    \eql@tagging@register@luamml{ams#1}%
    \eql@markline@amsthm@move{ams#1}{#1}%
  }%
}
```

`rovide@backup@amsbox` **TODO:** describe

```
\def\eql@provide@backup@amsbox#1{%
  \eql@amsmath@after{%
    \eql@provide@moveenv{ams#1}{#1}%
  }%
}
```

`provide@backup@eqref` Copy an `eqref` to `amseqref` whenever `amsmath` is loaded:

```
\def\eql@provide@backup@eqref{%
  \eql@amsmath@after{%
    \eql@provide@movecmd{amseqref}{eqref}%
  }%
}
```

`ide@backup@multlined` The environment `multlined` is supplied by `mathtools`. We copy it to `amsmultlined` anyway, but whenever `mathtools` is loaded:

```
\def\eql@provide@backup@multlined{%
  \AddToHook{package/mathtools/after}{%
    \eql@provide@moveenv{amsmultlined}{multlined}}%
}
```

`vide@backup@equation` The LaTeX environment `equation` is overwritten by several packages to implement their adjustments. Here we cater for adjustments through `amsmath`, `hyperref` and the PDF tagging mechanism. Copy `equation` and `equation*` whenever `amsmath` is loaded. Whenever `hyperref` is loaded, and `amsmath` is not yet present, backup the original LaTeX and `hyperref` versions of `equation`. If neither `hyperref` nor `amsmath` are present, just backup the original LaTeX `equation`. The PDF tagging mechanism registers `equation` upon `\begin{document}`. We thus need to register all copies of `equation` on our own, so that they can be used with their new names:

```
\def\eql@provide@backup@equation{%
  \eql@amsmath@after{%
    \eql@provide@moveenv{amsequation}{equation}%
    \eql@provide@moveenv{amsequation*}{equation*}%
    \eql@tagging@register@env{amsequation}%
    \eql@tagging@register@env{amsequation*}%
    \eql@tagging@register@luamml{amsequation}%
    \eql@tagging@register@luamml{amsequation*}%
    \eql@markline@amsthm@move{amsequation}{equation}%
    \eql@markline@amsthm@move{amsequation*}{equation*}%
  }%
  \AddToHook{package/hyperref/after}{%
    \@ifpackageloaded{amsmath}{}{%
      \eql@provide@moveenv{hyperrefequation}{equation}%
      \eql@tagging@register@env{hyperrefequation}%
      \eql@tagging@register@luamml{hyperrefequation}%
      \eql@markline@amsthm@move{hyperequation}{equation}%
    }%
  }%
  \@ifpackageloaded{amsmath}{}{%
    \@ifpackageloaded{hyperref}{%
      \let\latexequation\H@equation
      \let\endlatexequation\H@endequation
    }{\eql@provide@moveenv{latexequation}{equation}}%
    \eql@tagging@register@env{latexequation}%
    \eql@tagging@register@luamml{latexequation}%
    \eql@markline@amsthm@move{latexequation}{equation}%
  }%
}
```

`e@backup@displaymath` **TODO:** describe

```
\def\eql@provide@backup@displaymath{%
  \eql@provide@moveenv{latexdisplaymath}{displaymath}%
  \eql@markline@amsthm@move{latexdisplaymath}{displaymath}%
}
```

`@backup@subequations` The amsmath `subequations` environment is adjusted by hyperref through an environment hook, but this hook gets applied only later at `\begin{document}`. Hence, we need to supply the hook routine to the new routine ourselves:

```
\def\eql@provide@backup@subequations{%
  \eql@amsmath@after{%
    \eql@provide@moveenv{amssubequations}{subequations}%
  }%
  \AddToHook{package/hyperref/after}{%
    \AddToHook{cmd/amssubequations/before}%
    {%
      \stepcounter{equation}%
      \protected@edef\theHparentequation{\theHequation}%
      \addtocounter{equation}{-1}%
    }%
    \AddToHook{cmd/amssubequations/after}%
    {%
      \def\theHequation{\theHparentequation\alph{equation}}%
      \ignorespaces
    }%
  }%
}
```

`\eql@provide@backup` Backup all `amsmath` environments:

```
\def\eql@provide@backup{%
  \eql@provide@backup@eqref
  \eql@provide@backup@equation
  \eql@provide@backup@displaymath
  \eql@provide@backup@amsenv{gather}%
  \eql@provide@backup@amsenv{multline}%
  \eql@provide@backup@amsenv{align}%
  \eql@provide@backup@amsenv{flalign}%
  \eql@provide@backup@amsenv{alignat}%
  \eql@provide@backup@amsenv{xalignat}%
  \eql@provide@backup@amsenv{xxalignat}%
  \eql@provide@backup@amsenv{gather*}%
  \eql@provide@backup@amsenv{multline*}%
  \eql@provide@backup@amsenv{align*}%
  \eql@provide@backup@amsenv{flalign*}%
  \eql@provide@backup@amsenv{alignat*}%
  \eql@provide@backup@amsenv{xalignat*}%
  \eql@provide@backup@amsbox{gathered}%
  \eql@provide@backup@multlined
  \eql@provide@backup@amsbox{aligned}%
  \eql@provide@backup@amsbox{alignedat}%
  \eql@provide@backup@amsbox{cases}%
  \eql@provide@backup@amsbox{matrix}%
  \eql@provide@backup@amsbox{pmatrix}%
  \eql@provide@backup@amsbox{bmatrix}%
  \eql@provide@backup@amsbox{Bmatrix}%
  \eql@provide@backup@amsbox{vmatrix}%
  \eql@provide@backup@amsbox{Vmatrix}%
  \eql@provide@backup@subequations
}
```

Replacement amsmath Environments. **TODO:** describe

```
\def\eql@alignat@gobblecol#1{%
  \eql@ifnextchar@tight\bgroup{\@firstoftwo{#1}}{#1}}
```

`eql@gathered` (*env.*)
`eql@multlined` (*env.*)
`eql@aligned` (*env.*)

Define replacement versions for boxed environments `gathered`, `multlined` and `aligned` which forward to `equationsbox` with specific presets:

```
\newenvironment{eql@gathered}%
  {\eqnaddopt{lines}\equationsbox}{\endequationsbox}
\newenvironment{eql@multlined}%
  {\eqnaddopt{lines,padding,shape=steps}\equationsbox}{\endequationsbox}
\newenvironment{eql@aligned}%
  {\eqnaddopt{columns}\equationsbox}{\endequationsbox}
\newenvironment{eql@alignedat}%
  {\eqnaddopt{columns,colsep=off}\eql@alignat@gobblecol\equationsbox}%
  {\endequationsbox}
\newenvironment{eql@cases}%
  {\eqnaddopt{cases}\equationsbox}{\endequationsbox}
\newenvironment{eql@matrix}%
  {\eqnlinesset{modifierboxenv=false}\eqnaddopt{matrix=.}\equationsbox}%
  {\endequationsbox}
\newenvironment{eql@pmatrix}%
  {\eqnlinesset{modifierboxenv=false}\eqnaddopt{matrix=r}\equationsbox}%
  {\endequationsbox}
\newenvironment{eql@bmatrix}%
  {\eqnlinesset{modifierboxenv=false}\eqnaddopt{matrix=s}\equationsbox}%
  {\endequationsbox}
\newenvironment{eql@Bmatrix}%
  {\eqnlinesset{modifierboxenv=false}\eqnaddopt{matrix=c}\equationsbox}%
  {\endequationsbox}
\newenvironment{eql@vmatrix}%
  {\eqnlinesset{modifierboxenv=false}\eqnaddopt{matrix=v}\equationsbox}%
  {\endequationsbox}
\newenvironment{eql@Vmatrix}%
  {\eqnlinesset{modifierboxenv=false}\eqnaddopt{matrix=d}\equationsbox}%
  {\endequationsbox}
```

`eql@equation` (*env.*)
`eql@gather` (*env.*)
`eql@multline` (*env.*)
`eql@align` (*env.*)

Define replacement versions for display environments `equation`, `gather`, `multline`, `aligned` and derivatives which forward to `equations` with specific presets: **TODO:** amsmath at variants would need predefined columns for full operation

```
\newenvironment{eql@equation}%
  {\eqnaddopt{equation,donumber}\equations}{\endequations}
\newenvironment{eql@equation*}%
  {\eqnaddopt{equation,nonumber}\equations}{\endequations}
\newenvironment{eql@displaymath}%
  {\eqnaddopt{equation,nonumber}\equations}{\endequations}
\newenvironment{eql@gather}%
  {\eqnaddopt{lines,donumber}\equations}{\endequations}
\newenvironment{eql@gather*}%
  {\eqnaddopt{lines,nonumber}\equations}{\endequations}
\newenvironment{eql@multline}%
  {\eqnaddopt{lines,padding=max,shape=steps,numberline=out,donumber}%
   \equations}{\endequations}
\newenvironment{eql@multline*}%
  {\eqnaddopt{lines,padding=max,shape=steps,numberline=out,nonumber}%
   \equations}{\endequations}
\newenvironment{eql@align}%
  {\eqnaddopt{columns,donumber}\equations}{\endequations}
\newenvironment{eql@align*}%
  {\eqnaddopt{columns,nonumber}\equations}{\endequations}
\newenvironment{eql@flalign}%
  {\eqnaddopt{columns,fulllength,donumber}\equations}{\endequations}
\newenvironment{eql@flalign*}%
  {\eqnaddopt{columns,fulllength,nonumber}\equations}{\endequations}
\newenvironment{eql@alignat}%
  {\eqnaddopt{columns,colsep=off,donumber}%
   \eql@alignat@gobblecol\equations}{\endequations}
\newenvironment{eql@xalignat}%
  {\eqnaddopt{columns,donumber}%
   \eql@alignat@gobblecol\equations}{\endequations}
\newenvironment{eql@xxalignat}%
  {\eqnaddopt{columns,fulllength,donumber}%
   \eql@alignat@gobblecol\equations}{\endequations}
\newenvironment{eql@alignat*}%
  {\eqnaddopt{columns,colsep=off,nonumber}%
   \eql@alignat@gobblecol\equations}{\endequations}
\newenvironment{eql@xalignat*}%
  {\eqnaddopt{columns,nonumber}%
  \eql@alignat@gobblecol\equations}{\endequations}
```

Install Additional Environments. The additional environments need to be installed at their intended names which can be adjusted by the user.

`eql@provide@onlyonce` Process arguments for providing a specific environment. `#1` describes the environment using the amsmath name. `#2` specifies the desired target name. If `#2` is empty or equals `#1`, overwrite the amsmath environment in place making sure that the replacement is robust against loading amsmath before or after. If `#2` equals '`*`', just overwrite the amsmath environment in place immediately (e.g. within a block in the document body):

```
\def\eql@provide@onlyonce#1#2{%
  \def\eql@tmp{#2}\def\eql@tmpa{#1}%
  \ifx\eql@tmp\eql@tmpa
    \let\eql@tmp\@empty
  \fi
  \ifx\eql@tmp\@empty
    \let\eql@tmp\@undefined
    \ifx\@nodocument\relax
      \def\eql@tmp{#1}%
    \fi
    \ifcsname eql@provided@#1\endcsname
      \def\eql@tmp{#1}%
    \fi
    \eql@letcs{eql@provided@#1}\eql@true
  \else
    \def\eql@tmpa{*}%
    \ifx\eql@tmp\eql@tmpa
      \def\eql@tmp{#1}%
    \fi
  \fi
}
```

`\eql@provide@eqref` Provide `\eqref` as the macro `#1`. We have to check whether `#1` is empty or equals `\eqref` or takes the value '`*`'. If not, we should strip the backslash for further processing. Copy the macro into place, and copy again when amsmath or mathtools are loaded. Remove definition before amsmath is loaded in the future to avoid a potential error:

```
\def\eql@provide@eqref#1{%
  \def\eql@tmp{#1}\def\eql@tmpa{\eqref}%
  \ifx\eql@tmp\eql@tmpa
    \let\eql@tmp\@empty
  \fi
  \ifx\eql@tmp\@empty
    \eql@provide@onlyonce{eqref}{}%
  \else
    \def\eql@tmpa{*}%
    \ifx\eql@tmp\eql@tmpa
      \def\eql@tmp{eqref}%
    \else
      \edef\eql@tmp{\expandafter\@gobble\string#1}%
    \fi
  \fi
  \ifdefined\eql@tmp
    \expandafter\eql@provide@movecmd\expandafter{\eql@tmp}{eql@eqref}%
  \else
    \eql@amsmath@after{%
```

```
      \eql@provide@movecmd{eqref}{eql@eqref}%
    }%
    \AddToHook{package/mathtools/after}{%
      \eql@provide@movecmd{eqref}{eql@eqref}%
    }%
    \eql@provide@movecmd{eqref}{eql@eqref}%
    \eql@amsmath@undefine\eqref
  \fi
}
```

`\eql@provide@amsenv` Provide one of the amsmath environments. Copy into place, and copy again when amsmath is loaded. Remove definition before amsmath is loaded in the future to avoid an error:

```
\def\eql@provide@amsenv#1#2{%
  \eql@provide@onlyonce{#1}{#2}%
  \ifdefined\eql@tmp
    \eql@provide@moveenv{\eql@tmp}{eql@#1}%
    \eql@tagging@register@luamml{\eql@tmp}%
    \eql@markline@amsthm@register{\eql@tmp}%
  \else
    \eql@amsmath@after{%
      \eql@provide@moveenv{#1}{eql@#1}%
      \eql@markline@amsthm@register{#1}%
    }%
    \AddToHook{package/mathtools/after}{%
      \eql@provide@moveenv{#1}{eql@#1}%
      \eql@markline@amsthm@register{#1}%
    }%
    \eql@provide@moveenv{#1}{eql@#1}%
    \eql@markline@amsthm@register{#1}%
    \eql@amsmath@before{\eql@provide@undefineenv{#1}}%
  \fi
}
```

`\eql@provide@amsbox` Provide one of the amsmath subequation structures. Copy into place, and copy again when amsmath is loaded. Remove definition before amsmath is loaded in the future to avoid an error:

```
\def\eql@provide@amsbox#1#2{%
  \eql@provide@onlyonce{#1}{#2}%
  \ifdefined\eql@tmp
    \eql@provide@moveenv{\eql@tmp}{eql@#1}%
  \else
    \eql@amsmath@after{%
      \eql@provide@moveenv{#1}{eql@#1}}%
    \AddToHook{package/mathtools/after}{%
      \eql@provide@moveenv{#1}{eql@#1}}%
    \eql@provide@moveenv{#1}{eql@#1}%
    \eql@amsmath@before{\eql@provide@undefineenv{#1}}%
  \fi
}
```

`ql@provide@multlined` Provide mathtools environment `multlined`. Copy into place, and copy again when mathtools is loaded. Remove definition before mathtools is loaded in the future to avoid an error:

```
\def\eql@provide@multlined#1{%
  \eql@provide@onlyonce{multlined}{#1}%
  \ifdefined\eql@tmp
    \eql@provide@moveenv{\eql@tmp}{eql@multlined}%
  \else
    \AddToHook{package/mathtools/after}{%
      \eql@provide@moveenv{multlined}{eql@multlined}}%
    \eql@provide@moveenv{multlined}{eql@multlined}%
    \@ifpackageloaded{mathtools}{}{\AddToHook{package/mathtools/before}{%
      \eql@provide@undefineenv{multlined}}}%
  \fi
}
```

`\eql@provide@matrix`

Provide the `cases` and `matrix` environments. Copy into place, and copy again when amsmath is loaded:

```
\def\eql@provide@matrix#1#2#3{%
  \eql@provide@onlyonce{#1}{#3}%
  \ifdefined\eql@tmp
    \eql@provide@moveenv{\eql@tmp}{eql@#1}%
    \eql@tagging@register@luamml{\eql@tmp}%
  \else
    \eql@amsmath@after{%
      \eql@provide@moveenv{#1}{eql@#1}%
    }%
    \eql@provide@moveenv{#1}{eql@#1}%
    \ifdefined#2\eql@amsmath@before{\eql@provide@undefineenv{#1}}\fi%
  \fi
}
```

`eql@provide@equation`

Provide the environment `equation`. Copy into place, and copy again when amsmath or hyperref are loaded. When PDF tagging is active, the environment is modified at `\begin{document}` in an undesirable fashion, so copy the definition again:

```
\def\eql@provide@equation#1{%
  \eql@provide@onlyonce{equation}{#1}%
  \ifdefined\eql@tmp
    \eql@provide@moveenv{\eql@tmp}{eql@equation}%
    \eql@tagging@register@luamml{\eql@tmp}%
    \eql@markline@amsthm@register{\eql@tmp}%
  \else
    \eql@amsmath@after{%
      \eql@provide@moveenv{equation}{eql@equation}%
      \eql@markline@amsthm@register{equation}%
    }%
    \AddToHook{package/hyperref/after}{%
      \@ifpackageloaded{amsmath}{}{%
        \eql@provide@moveenv{equation}{eql@equation}%
        \eql@markline@amsthm@register{equation}%
      }%
    }%
    \eql@provide@moveenv{equation}{eql@equation}%
    \eql@markline@amsthm@register{equation}%
    \ifdefined\eql@tagging@on
      \AddToHook{begindocument/end}{%
        \eql@provide@moveenv{equation}{eql@equation}%
        \eql@markline@amsthm@register{equation}%
      }%
    \fi
  \fi
}
```

`provide@equationstar`

Provide the environment `equation*`. Copy into place, and copy again when amsmath or hyperref are loaded. Remove definition of `equation*` before amsmath is loaded in the future to avoid an error. When PDF tagging is active, the environment is modified at `\begin{document}` in an undesirable fashion, so copy the definition again:

```
\def\eql@provide@equationstar#1{%
  \eql@provide@onlyonce{equation*}{#1}%
  \ifdefined\eql@tmp
    \eql@provide@moveenv{\eql@tmp}{eql@equation*}%
    \eql@tagging@register@luamml{\eql@tmp}%
    \eql@markline@amsthm@register{\eql@tmp}%
  \else
    \eql@amsmath@after{%
      \eql@provide@moveenv{equation*}{eql@equation*}%
      \eql@markline@amsthm@register{equation*}%
    }%
    \eql@provide@moveenv{equation*}{eql@equation*}%
    \eql@markline@amsthm@register{equation*}%
    \eql@amsmath@before{\eql@provide@undefineenv{equation*}}%
    \ifdefined\eql@tagging@on
      \AddToHook{begindocument/end}{%
        \eql@provide@moveenv{equation*}{eql@equation*}%
        \eql@markline@amsthm@register{equation*}%
      }%
    \fi
  \fi
}
```

`@provide@displaymath`

TODO: describe

```
\def\eql@provide@displaymath#1{%
  \eql@provide@onlyonce{displaymath}{#1}%
  \ifdefined\eql@tmp
    \eql@provide@moveenv{\eql@tmp}{eql@displaymath}%
    \eql@markline@amsthm@register{\eql@tmp}%
    \eql@tagging@register@luamml{\eql@tmp}%
  \else
    \eql@provide@moveenv{displaymath}{eql@displaymath}%
    \eql@markline@amsthm@register{displaymath}%
    \ifdefined\eql@tagging@on
      \AddToHook{begindocument/end}{%
        \eql@provide@moveenv{displaymath}{eql@displaymath}}%
    \fi
  \fi
}
```

`provide@subequations`

Provide the amsmath environment `subequations`. Copy into place, and copy again when amsmath is loaded. hyperref adds a hook to the command which messes up the parsing of optional arguments (even if the hook is emptied). The hook placement happens at `\begin{document}`, so we copy the environment again afterwards. We also remove the hook (after adding an empty hook to avoid errors). Remove definition before amsmath is loaded in the future to avoid an error:

```
\def\eql@provide@subequations#1{%
  \eql@provide@onlyonce{subequations}{#1}%
  \ifdefined\eql@tmp
    \eql@provide@moveenv{\eql@tmp}{eql@subequations}%
  \else
```

```
    \eql@amsmath@after{%
      \eql@provide@moveenv{subequations}{eql@subequations}%
    }%
    \AddToHook{package/hyperref/after}{%
      \AddToHook{cmd/subequations/before}[hyperref]{}%
      \AddToHook{cmd/subequations/after}[hyperref]{}%
      \RemoveFromHook{cmd/subequations/before}[hyperref]%
      \RemoveFromHook{cmd/subequations/after}[hyperref]%
      \AddToHook{begindocument/end}{%
        \eql@provide@moveenv{subequations}{eql@subequations}}%
    }%
    \eql@provide@moveenv{subequations}{eql@subequations}%
    \eql@amsmath@before{\eql@provide@undefineenv{subequations}}%
  \fi
}
```

`\eql@provide@sqr` Provide the symbolic environment `\[`...`\]`. Copy into place, and copy again when amsmath is loaded. If PDF tagging is active, some undesired modifications happen at `\begin{document}`, so copy again afterwards:

```
\def\eql@provide@sqr{%
  \let\[\eql@sqr@open
  \let\]\eql@sqr@close
  \eql@amsmath@after{%
    \let\[\eql@sqr@open
    \let\]\eql@sqr@close
  }%
  \ifdefined\eql@tagging@on
    \AddToHook{begindocument/end}{%
      \let\[\eql@sqr@open
      \let\]\eql@sqr@close
    }%
  \fi
}
```

`\eql@provide@ang` Provide the symbolic environment `\<`...`\>`. This is easy because none of the other packages uses this structure:

```
\def\eql@provide@ang{%
  \let\<\eql@ang@open
  \let\>\eql@ang@close
}
```

TODO: describe

```
\def\eql@provide@tagform{%
  \def\tagform@##1{\maketag@@@{\eql@tags@tagform{##1}}}}
\def\eql@provide@maketag{%
  \def\maketag@@@##1{\eql@tags@taglayout{##1}}}
```

Interface.

`provide` (*key*) We provide the additional environments via key-value pairs, where the value specifies the intended name:

```
\eql@define@key{provide}{equation}[]{\eql@provide@equation{#1}}
\eql@define@key{provide}{equation*}[]{\eql@provide@equationstar{#1}}
\eql@define@key{provide}{displaymath}[]{\eql@provide@displaymath{#1}}
\eql@define@key{provide}{gather}[]{\eql@provide@amsenv{gather}{#1}}
```

```
\eql@define@key{provide}{multline}[]{\eql@provide@amsenv{multline}{#1}}
\eql@define@key{provide}{align}[]{\eql@provide@amsenv{align}{#1}}
\eql@define@key{provide}{flalign}[]{\eql@provide@amsenv{flalign}{#1}}
\eql@define@key{provide}{alignat}[]{\eql@provide@amsenv{alignat}{#1}}
\eql@define@key{provide}{xalignat}[]{\eql@provide@amsenv{xalignat}{#1}}
\eql@define@key{provide}{xxalignat}[]{\eql@provide@amsenv{xxalignat}{#1}}
\eql@define@key{provide}{gather*}[]{\eql@provide@amsenv{gather*}{#1}}
\eql@define@key{provide}{multline*}[]{\eql@provide@amsenv{multline*}{#1}}
\eql@define@key{provide}{align*}[]{\eql@provide@amsenv{align*}{#1}}
\eql@define@key{provide}{flalign*}[]{\eql@provide@amsenv{flalign*}{#1}}
\eql@define@key{provide}{alignat*}[]{\eql@provide@amsenv{alignat*}{#1}}
\eql@define@key{provide}{xalignat*}[]{\eql@provide@amsenv{xalignat*}{#1}}
\eql@define@key{provide}{gathered}[]{\eql@provide@amsbox{gathered}{#1}}
\eql@define@key{provide}{multlined}[]{\eql@provide@multlined{#1}}
\eql@define@key{provide}{aligned}[]{\eql@provide@amsbox{aligned}{#1}}
\eql@define@key{provide}{alignedat}[]{\eql@provide@amsbox{alignedat}{#1}}
\eql@define@key{provide}{cases}[]{\eql@provide@matrix{cases}\eql@false{#1}}
\eql@define@key{provide}{matrix}[]{\eql@provide@matrix{matrix}\eql@false{#1}}
\eql@define@key{provide}{pmatrix}[]{\eql@provide@matrix{pmatrix}\eql@false{#1}}
\eql@define@key{provide}{bmatrix}[]{\eql@provide@matrix{bmatrix}\eql@true{#1}}
\eql@define@key{provide}{Bmatrix}[]{\eql@provide@matrix{Bmatrix}\eql@true{#1}}
\eql@define@key{provide}{vmatrix}[]{\eql@provide@matrix{vmatrix}\eql@true{#1}}
\eql@define@key{provide}{Vmatrix}[]{\eql@provide@matrix{Vmatrix}\eql@true{#1}}
\eql@define@key{provide}{subequations}[]{\eql@provide@subequations{#1}}
\eql@define@key{provide}{sqr}[]{\eql@provide@sqr}
\eql@define@key{provide}{ang}[]{\eql@provide@ang}
\eql@define@key{provide}{eqref}[]{\eql@provide@eqref{#1}}
\eql@define@key{provide}{tagform}[]{\eql@provide@tagform}
\eql@define@key{provide}{maketag}[]{\eql@provide@maketag}
```

`\eqnlinesprovide` Provide an additional environment or macro via key-value interface:

```
\newcommand{\eqnlinesprovide}[1]{%
  \eql@setkeys{provide}{#1}%
  \ignorespaces
}
```

16.5 Global and Package Options

Handle global and package options:

Disable error message for exclusive package options:

```
\let\eql@error@packageoption\@gobble
```

Declare math layout options `leqno` and `fleqn` for common LaTeX classes:

```
\DeclareOption{leqno}{\eqnlinesset{tagsleft}}
\DeclareOption{fleqn}{\eqnlinesset{left}}
```

Pass undeclared options on to keyval processing:

```
\DeclareOption*{\expandafter\eqnlinesset\expandafter{\CurrentOption}}
```

Set defaults for package:

```
\eql@defaults@eqnlines
\eql@equations@columns@set
\eql@box@columns@set
```

Make sure that the amsmath conditionals `\iftagsleft@` and `\if@fleqn` are declared without spelling out their name which may upset the TeX conditional parsing mechanism:

```
\ifdefined\tagsleft@true\else
  \expandafter\newif\csname iftagsleft@\endcsname
\fi
\ifdefined\@fleqntrue\else
  \expandafter\newif\csname if@fleqn\endcsname
\fi
```

Import amsmath switches `leqno` as `tagsleft` and `fleqn` as `left`:

```
\eql@amsmath@after{%
  \ifnum\eql@provide@opt@env=\tw@
    \iftagsleft@
      \eqnlinesset{tags=left}%
    \else
      \eqnlinesset{tags=right}%
    \fi
    \if@fleqn
      \eqnlinesset{layout=left}%
    \else
      \eqnlinesset{layout=center}%
    \fi
  \fi
}
```

Process package options:

```
\ProcessOptions
```

`@error@packageoption` Enable error message for exclusive package options:

```
\def\eql@error@packageoption#1{%
  \eql@error{may only use '#1' as a package option}%
}
```

Make the ending statements for amsmath environments independent if desired, so that they may be overwritten individually:

```
\ifnum\eql@provide@opt@env=\tw@
\ifdefined\eql@provide@opt@matrix
  \let\eql@provide@opt@amsmathpatch\eql@false
\fi\fi
\ifdefined\eql@provide@opt@backup
  \let\eql@provide@opt@amsmathpatch\eql@true
\fi
\ifdefined\eql@provide@opt@amsmathpatch
  \eql@amsmath@fixends
  \eql@amsmath@fixmatrices
\fi
```

Backup all amsmath environments that may be overwritten to `ams...`. This will happen before any replacements:

```
\ifdefined\eql@provide@opt@backup\eql@provide@backup\fi
```

Provide native LaTeX environment `equation` and symbolic shortcut `\[`...`\]` if desired:

```
\ifnum\eql@provide@opt@env>\z@
  \eqnlinesprovide{equation,equation*,sqr,displaymath}
\fi
```

Provide amsmath equation environments if desired:

```
\ifnum\eql@provide@opt@env=\tw@
  \eqnlinesprovide{%
    multline,gather,align,flalign,alignat,xalignat,xxalignat,%
    multline*,gather*,align*,flalign*,alignat*,xalignat*,%
    multlined,gathered,aligned,alignedat,%
    subequations}
\fi
```

Provide symbolic shortcut `\<`...`\>` if desired:

```
\ifdefined\eql@provide@opt@ang\eqnlinesprovide{ang}\fi
```

Provide equation reference `\eqref` if desired:

```
\ifdefined\eql@provide@opt@eqref\eqnlinesprovide{eqref}\fi
```

Provide `cases` and `matrix` environments if desired:

```
\ifdefined\eql@provide@opt@matrix
  \eqnlinesprovide{cases,matrix,pmatrix,bmatrix,Bmatrix,vmatrix,Vmatrix}
\fi
```