

l3doc 文档类 — 实验性质*

The L^AT_EX3 Project[†] 2023 年 12 月 11 日 发布

张泓知 2023 年 12 月 20 日 【译】

目 录

1	介绍	3
2	其他包的特性	3
2.1	hypdoc 包	3
2.2	docmfp 包	3
2.3	xdoc2 包	4
2.4	gmdoc 包	4
3	问题与待办事项	4
4	文档	5
4.1	配置	5
4.2	类选项	5
4.3	文档和实现的分割	6
4.4	一般文本标记	6
4.5	在文档中描述函数	8
4.6	描述实现中的函数	9
4.7	保持一致性	10
4.8	文档化模板	11

*根据广泛需求，我们现在发布了这个实验性的类文档。但请注意，它绝不是最终版本，并且很有可能会经历修改，甚至是不兼容的修改！因此，如果类发生变化，可能需要进行更新才能继续使用。

[†]<https://www.latex-project.org/latex3/>

5	l3doc 代码实现	11
5.1	变量	12
5.2	Variants and helpers	17
5.3	Messages	25
5.4	Options and configuration	26
5.5	Class and package loading	27
5.6	Configuration and tweaks	28
5.7	Design	29
5.8	Text markup	31
5.9	Implementing text markup	36
5.9.1	Common between <code>macro</code> and <code>function</code>	39
5.9.2	The <code>function</code> environment	45
5.9.3	The <code>macro</code> environment	55
5.9.4	Misc	66
5.9.5	NB and NOTE	67
5.10	Footnote support	68
5.11	Documenting templates	69
5.12	Inheriting doc	70
5.12.1	The <code>macrocode</code> environment	75
5.13	At end document	77
5.14	Indexing	80
5.14.1	Necessary patching	80
5.14.2	Userspace commands	81
5.14.3	Internal index commands	82
5.14.4	Finding sort-key and module	86
5.15	Change history	90
5.16	Default configuration	90
5.17	Internal macros for L ^A T _E X3 sources	91
5.18	Math extras	92
5.19	Makeindex configuration	92
	索引	93

1 介绍

在 doc 从版本 2 变更为版本 3 之前编写了此类的代码和文档，这已经显示出这个类目前的落后程度。所以请认真对待以下警告：

**它的稳定性远不如主要的 expl3 包。
请自行承担风险！**

这是一个专门用于记录 expl3 捆绑包的类，它是组成 L^AT_EX3 编程环境的模块或包的集合。最终它将取代 ltxdoc 类作为 L^AT_EX3 的文档类，但在吸收 hypdoc、xdoc2、docmfp 和 gmdoc 中的优秀思想之前不会这样做。

它被编写为一个“自包含”的 docstrip 文件：执行 `latex l3doc.dtx` 将生成文件 `l3doc.cls` 并排版此文档；执行 `tex l3doc.dtx` 将只生成 `l3doc.cls`。

2 其他包的特性

这个类基于 ltxdoc 类和 doc 宏包，但在它们最初编写之后，一些改进和替代方案出现了，我们希望能够借鉴这些新特性。

这些包或类有 hypdoc、docmfp、gmdoc 和 xdoc。我在下面对它们进行了总结，以便确定我们至少应该为 l3doc 设定什么样的最低特性。

2.1 hypdoc 包

此包为 doc 包提供了超链接支持。我将它包含在此列表中是为了提醒我，文档和方法实现之间的交叉引用并不是很好。（例如，能够自动地从方法实现链接到其文档说明，反之亦然，将会很不错。）

2.2 docmfp 包

- 为 MetaFont 和 MetaPost 代码提供了 `\DescribeRoutine` 和 `routine` 环境（等等）。
- 为更通用的代码提供了 `\DescribeVariable` 和 `variable` 环境（等等）。
- 提供了 `\Describe` 和 `Code` 环境（等等）作为上述两个实例的一般化。
- 对 DocStrip 系统进行了小的调整，以帮助非 L^AT_EX 的使用。

2.3 xdoc2 包

- 双面打印支持。
- `\NewMacroEnvironment`、`\NewDescribeEnvironment`；与 `docmfp` 类似的概念但更全面。
- 大量小改进。

2.4 gmdoc 包

将 `doc` 作为包或类进行了根本性的重新实现。

- 不需要 `\begin{macrocode}` 块！
- 自动插入 `\begin{macro}` 块！
- 还有许多其他细微的改进。

3 问题与待办事项

目前存在的问题：(1) 对可以记录的内容类型不够灵活；(2) `\begin{function}` 环境用于记录内容，与在实现中类似地使用的 `\begin{macro}` 函数之间没有明显的联系。

在用于实现部分时，`macro` 可能应该改名为 `function`。但在这种改名发生之前，它们应该具有相同的语法！

此外，我们需要另一层文档命令来处理“用户宏”与“代码函数”；`expl3` 函数可能需要不同的文档方式（至少在索引方面），与 `ltxcmd` 用户宏不同。

以下是一些待完成事项的列表，没有特定顺序：

- 将 `function/macro` 环境重命名，以更好地描述其用途。
- 普遍化 `function/macro`，用于记录“其他内容”，如环境名称、包选项，甚至键值选项。
- 像 `\part` 一样新增一个用于文件的函数（删除笨拙的“File”作为 `\partname`）。
- 寻找更好的替代方案来取代 `\StopEventually`；我考虑使用两个环境 `documentation` 和 `implementation`，它们可以有条件地排版/忽略其内容。（这已经被实现，但需要进一步考虑。）
- 将宏的文档和实现进行超链接（参考 `svn-multi v2` 的 `DTX` 文件）。现在这部分已经部分完成，但需要改进。

4 文档

4.1 配置

在处理类选项之前，l3doc 如果存在配置文件 l3doc.cfg，将加载它，允许你在不必更改文档源文件的情况下定制类的行为。

例如，要在信纸大小的纸张上生成文档而不是默认的 A4 大小，创建 l3doc.cfg，并包含以下内容：

```
\PassOptionsToClass{letterpaper}{l3doc}
```

默认情况下，l3doc 选择 T1 字体编码并加载 Latin Modern 字体。要阻止这一行为，可以使用类选项 cm-default。

4.2 类选项

该类识别了许多选项，其中一些是通常有用的，另一些则专门针对内核团队使用。

- | | |
|-------------------|--|
| full | 当设置 full 选项时（标准设置），源文件的文档和实现部分都会排版。另一方面，如果设置了 onlydoc 选项，则只会排版文档部分。 |
| onlydoc | |
| lm-default | 选择标准字体设置是在 T1 编码下的 Latin Modern（标准设置），还是保持字体设置不变。 |
| kernel | 确定 l3doc 是否将 __kernel_ 命令和 \cgl__kernel_ 变量视为代码中可接受的内容。一般来说，不允许来自当前模块外部的内部内容。然而，为了引导 expl3 内核，需要一些跨模块的功能。为了避免否则会出现的错误消息，可以使用类选项 kernel。 |
| check | 给定 check 选项时，类将记录在 <name>.cmds 文件中定义和记录的所有命令。这将显示哪些命令既被记录又被定义，哪些仅被记录，以及哪些仅被定义。（这里，“定义”指的是在源文件的实现部分使用 macro 或 variable 环境列出的命令。） |
| checktest | 给定 checktest 选项时，类将检查源文件实现部分中的每个函数条目是否使用了 \UnitTest 进行了标记。 |
| show-notes | 这些互补选项确定是否打印使用 \NB 和 \NOTE 命令提供的信息。 |
| hide-notes | 命令 \cmd 和 \cs 允许在大多数下划线后进行连字符的处理。默认情况下，会使用连字符标记连字符位置，但可以使用 cs-break-nohyphen 类选项进行更改。若要完全禁用控制序列的连字符处理，使用 cs-break = false。 |
| cs-break | |
| cs-break-nohyphen | |

4.3 文档和实现的分割

doc 使用 `\OnlyDocumentation/\AlsoImplementation` 宏来指导 `\StopEventually{}` 的使用, 该命令用于在单个 `.dtx` 文件中分隔文档和实现部分。

这并不十分灵活, 因为它假定我们总是要打印文档部分。对于 `expl3` 源文件, 我希望能够以两种模式输入 `.dtx` 文件: 只显示文档部分和只显示实现部分。例如:

```
\DisableImplementation
\DocInput{l3basics,l3prg,...}
\EnableImplementation
\DisableDocumentation
\DocInputAgain
```

`expl3` 包的整个文档, 包括实现部分在最后。这不是完美的, 但是这是一个开始。

在文档部分使用 `\begin{documentation}...\end{documentation}`, 在实现部分使用 `\begin{implementation}...\end{implementation}`。

`\EnableDocumentation/\EnableImplementation` 使其在 `.dtx` 文件 `\DocInput` 时能够排版; 使用 `\DisableDocumentation/\DisableImplementation` 可以省略这些环境的内容。

注意, `\DocInput` 现在接受逗号分隔的参数, 并且 `\DocInputAgain` 可以重新输入以这种方式先前输入的所有 `.dtx` 文件。

4.4 一般文本标记

本节中的许多命令来自于 `ltxdoc`, 做了一些改进。

`\cmd` `\cmd` [*options*] *control sequence*

`\cs` `\cs` [*options*] {*csname*}

这些命令用于排版控制序列。`\cmd\foo` 生成 “\foo”，而 `\cs{foo}` 也生成相同的效果。通常情况下，`\cs` 更健壮，因为它不依赖于类别码是否“正确”，因此更推荐使用。

这些命令知道 `@@ l3docstrip` 语法，并正确替换文档中的这些实例。这仅在 `%<@@=<module>` 声明之后发生。

此外，命令可以用在 `\cs` 的参数中。例如，`\cs{\meta{name}:\meta{signature}}` 生成 `\<name>:\<signature>`。

`<选项>` 是一个键值列表，可以包含以下键：

- `index=<name>`：将 `<csname>` 索引，就好像写了 `\cs{<name>}` 一样。
- `no-index`：不索引 `<csname>`。
- `module=<module>`：在 `<module>` 的命令列表中索引 `<csname>`；特别的，`<module>` 可以是 `TeX`，表示 “`TeX` 和 `LaTeX 2ε`” 命令，或者为空，表示放在主索引中。默认情况下，`<module>` 从命令名称中自动推断。
- `replace` 是一个布尔键（默认为 `true`），表示是否像 `l3docstrip` 那样替换 `@@`。

这些命令允许在大多数下划线后进行连字符处理。默认情况下，会使用连字符标记连字符位置，但可以使用 `cs-break-nohyphen` 类选项进行更改。若要完全禁用控制序列的连字符处理，使用 `cs-break = false`。

`\tn` `\tn` [*options*] {*csname*}

与 `\cs` 类似，但用于“传统”`TeX` 或 `LaTeX 2ε` 命令；它们会相应地进行索引。实际上，这相当于 `\cs [module=TeX, replace=false, <options>] {<csname>}`。

`\meta` `\meta` {*<name>*}

`\meta` 以斜体在 *<angle brackets>* 中排版 *<name>*。在 `function` 等环境中，尖括号 `<...>` 被设置为 `\meta{...}` 的简写。

与其 `ltxdoc` 版本相比，此函数有额外功能；下划线可以用于标记数学模式中的下标。例如，`\meta{arg_{xy}}` 生成 “*<arg_{xy}>*”。

`\Arg` `\Arg` {*<name>*}

`\marg` 将 *<name>* 以 `\meta` 的方式排版，并用大括号包裹。

`\oarg` `\marg/\oarg/\parg` 版本从 `ltxdoc` 派生，分别用于 `LaTeX 2ε` 语法中的“必选”、“可选”或“图片”方括号。

`\file` `\pkg` `{\name}`

`\env` 这些命令都接受一个参数，用于表示文件、环境、包名和类名的语义命令。

`\pkg`

`\cls`

`\NB` `\NB` `{\tag}` `{\comments}`

`\NOTE` `\begin{NOTE}` `{\tag}`

`\end{NOTE}`

`\end{NOTE}`

在源文件中做注释，默认情况下不进行排版。当激活 `show-notes` 类选项时，注释以非标记和抄录的方式排版。

4.5 在文档中描述函数

`function` (*env.*) 有两个经常使用的环境来描述 `expl3` 的函数和变量。如果描述一个变量，使用后者的环境；它与 `function` 环境的行为完全相同。通常，上述两个环境会与 `syntax` `syntax` (*env.*) 环境结合使用，以描述它们的语法。

```
\begin{function}{\package_function_one:N, \package_function_two:n}
  \begin{syntax}
    \cs{package_function_one:N} \meta{cs}
    \cs{package_function_two:n} \marg{Argument}
  \end{syntax}
  这里是描述的文字 ...
\end{function}
```

`\package_function_one:N` `\package_function_one:N` `<cs>`

`\package_function_two:n` `\package_function_two:n` `{\Argument}`

这里是描述的文字...

函数环境可以带有可选参数，表示所描述的函数是可展开的（使用`EXP`）、受限可展开的（使用`rEXP`），或以条件形式定义（使用`TF`、`pTF`或`noTF`）。注意，`pTF` 意味着 `EXP`，因为谓词必须始终是可展开的，而`noTF`表示函数在没有`TF`的情况下应该另外进行文档化。对于条件形式`TF`和`pTF`，`function`环境的参数实际上并不是一个存在的命令：在下面的示例中，`\tl_if_empty:N`并不存在，但它的条件形式`\tl_if_empty:NT`、`\tl_if_empty:NF`、`\tl_if_empty:NTF`，以及谓词形式`\tl_if_empty_p:N`是存在的：


```

\begin{function}[pTF]{\tl_if_empty:N, \tl_if_empty:c}
  \begin{syntax}
    \cs{tl_if_empty_p:N} \meta{tl~var}
    \cs{tl_if_empty:NTF} \meta{tl~var} \Arg{true code} \Arg{false code}
  \end{syntax}
  检查\meta{token list variable} 是否完全为空（即不包含任何标记）。
\end{function}

```

```

\tl_if_empty_p:N * \tl_if_empty_p:N <tl var>
\tl_if_empty_p:c * \tl_if_empty:NTF <tl var> {\true code}
\tl_if_empty:NTF * {\false code}
\tl_if_empty:c * 检查<token list variable>是否完全为空
                  (即不包含任何标记)。

```

`texnote (env.)` 这个环境用于突出显示仅对经验丰富的 T_EX 开发人员感兴趣的 `function` 和类似环境中的部分内容。

4.6 描述实现中的函数

`macro (env.)` 在 L^AT_EX 2_ε 中用于标记宏/函数实现的常用环境仍然是 `macro` 环境。在 l3doc 中有一些变化：现在它接受逗号分隔的函数列表，以避免大量连续的 `\end{macro}` 语句。空格和换行被忽略（选项 `[verb]` 可以防止这种情况）。

```

% \begin{macro}{\foo:N, \foo:c}
%   \begin{macrocode}
... code for \foo:N and \foo:c ...
%   \end{macrocode}
% \end{macro}

```

如果你正在文档化辅助宏，通常不需要如此突出它，也不需要检查它是否具有测试函数，是否在 `function` 环境中先有文档块。l3doc 将从名称中的 `__` 的存在或使用 `\begin{macro}[int]` 强制标记为内部来识别这些情况。对于这些情况，边距标注将以灰色打印出来。

对于文档化 expl3 类型的条件语句，你也可以在环境中传递 TF 选项（并从函数名称中省略它），表示该函数提供了 T、F 和 TF 后缀。类似的 pTF 选项会打印出 TF 和 _p 谓词形式。选项 noTF 会打印出 TF 形式和既没有 T 也没有 F 的形式，用于文档化诸如 `\prop_get:NN` 这样也有条件形式的函数（`\prop_get:NNTF`）。

在极少数情况下，一个“公共”函数没有用户文档。在这些罕见情况下，可以添加选项 `no-user-doc` 来抑制未定义引用。

`\TestFiles` `\TestFiles{<文件列表>}` 用于指示当前代码使用的测试文件；它们将在文档中打印出来。

`\UnitTested` 在 `macro` 环境中，标记命令是否已创建单元测试是个好主意。这可通过在 `\begin{macro} ... \end{macro}` 之间的任何位置写入 `\UnitTested` 来表示。

如果启用了类选项 `checktest`，那么在没有调用 `Testfiles` 的 `macro` 环境中会产生一个错误。这是为了像 `expl3` 这样的大型包设计的，这些包应该有完全详尽的测试套件，其作者在添加新代码时可能不总是如应该般及时添加新测试。

`\TestMissing` 如果一个函数缺少测试，可以通过写（需要多次）`\TestMissing {<explanation of test required>}` 来标记这些缺失的测试。这些缺失的测试将在编译运行结束时的列表中进行总结打印。

`variable (env.)` 在文档化变量定义时，使用 `variable` 环境代替。它的行为与 `macro` 环境完全相同，只是如果启用了类选项 `checktest`，则不需要为变量提供测试文件。

`arguments (env.)` 在 `macro` 环境中，你可以使用 `arguments` 环境描述函数的参数。它的行为类似于修改后的 `enumerate` 环境。

```
% \begin{macro}{\foo:nn, \foo:VV}
% \begin{arguments}
%   \item Name of froozle to be frazzled
%   \item Name of muble to be jubled
% \end{arguments}
%   \begin{macrocode}
... code for \foo:nn and \foo:VV ...
%   \end{macrocode}
% \end{macro}
```

4.7 保持一致性

每当使用 `function` 或 `macro` 文档化或定义一个函数时，其名称都会存储在一个序列中以供以后处理。

在文档末尾（即在处理完 `.dtx` 文件之后），会分析名称列表，检查是否所有已定义的函数都已经文档化，反之亦然。结果将打印在控制台输出中。

如果你需要对这些名称列表进行更严格的处理，可以查看数据结构和用于直接存储和访问它们的方法的实现。

4.8 文档化模板

提供以下宏用于文档化模板；可能最终会变成完全不同的内容，但谁知道呢。

```
\begin{TemplateInterfaceDescription} {\langle template type name \rangle}
  \TemplateArgument{none}{---}
```

或者一个或多个这些：

```
\TemplateArgument {\langle arg no \rangle} {\langle meaning \rangle}
```

和

```
\TemplateSemantics
  \langle text describing the template type semantics \rangle
```

```
\end{TemplateInterfaceDescription}
```

```
\begin{TemplateDescription} {\langle template type name \rangle} {\langle name \rangle}
```

一个或多个这些：

```
\TemplateKey {\langle key name \rangle} {\langle type of key \rangle}
```

```
{\langle textual description of meaning \rangle}
```

```
{\langle default value if any \rangle}
```

和

```
\TemplateSemantics
  \langle text describing special additional semantics of the template \rangle
```

```
\end{TemplateDescription}
```

```
\begin{InstanceDescription} [\langle text to specify key column width (optional) \rangle]
```

```
{\langle template type name \rangle} {\langle instance name \rangle} {\langle template name \rangle}
```

一个或多个这些：

```
\InstanceKey {\langle key name \rangle} {\langle value \rangle}
```

和

```
\InstanceSemantics
  \langle text describing the result of this instance \rangle
```

```
\end{InstanceDescription}
```

5 l3doc 代码实现

1 `*class`

2 `\@@=codedoc`

5.1 变量

`\g_docinput_clist` 通过 `\DocInput` 输入的文件列表。

```
3 \clist_new:N \g_docinput_clist
```

(End of definition for `\g_docinput_clist`. This variable is documented on page ??.)

`\g_doc_functions_seq` 通过 `function` 文档化的所有函数和通过 `macro` 引入的所有宏。可以进行比较，查看文档或代码缺失的部分。

`\g_doc_macros_seq`

```
4 \seq_new:N \g_doc_functions_seq
```

```
5 \seq_new:N \g_doc_macros_seq
```

(End of definition for `\g_doc_functions_seq` and `\g_doc_macros_seq`. These variables are documented on page ??.)

`\l_codedoc_detect_internals_bool` If `true`, `l3doc` will check for use of internal commands `_<pkg>_...` from other packages in the argument of the `macro` environment, and in the code typeset in `macrocode` environments, but not in `\cs`. Also a token list to store temporary data for this purpose.

`\l_codedoc_detect_internals_tl`

```
6 \bool_new:N \l_codedoc_detect_internals_bool
```

```
7 \bool_set_true:N \l_codedoc_detect_internals_bool
```

```
8 \tl_new:N \l_codedoc_detect_internals_tl
```

```
9 \tl_new:N \l_codedoc_detect_internals_cs_tl
```

(End of definition for `\l_codedoc_detect_internals_bool` and `\l_codedoc_detect_internals_tl`.)

`\l__codedoc_output_coffin` The `function` environment is typeset by combining coffins containing various pieces (function names, description, *etc.*) into this coffin.

```
10 \coffin_new:N \l__codedoc_output_coffin
```

(End of definition for `\l__codedoc_output_coffin`.)

`\l__codedoc_functions_coffin` These coffins contain respectively the list of function names (argument of the `function` environment), the text between `\begin{function}` and `\end{function}`,
`\l__codedoc_descr_coffin` and the syntax given in the `syntax` environment.
`\l__codedoc_syntax_coffin`

```
11 \coffin_new:N \l__codedoc_functions_coffin
```

```
12 \coffin_new:N \l__codedoc_descr_coffin
```

```
13 \coffin_new:N \l__codedoc_syntax_coffin
```

(End of definition for `\l__codedoc_functions_coffin`, `\l__codedoc_descr_coffin`, and `\l__codedoc_syntax_coffin`.)

<code>\g__codedoc_syntax_box</code>	<p>The contents of the <code>syntax</code> environment are typeset in this box before being transferred to <code>\l__codedoc_syntax_coffin</code>.</p> <pre>14 \box_new:N \g__codedoc_syntax_box</pre> <p>(End of definition for <code>\g__codedoc_syntax_box</code>.)</p>
<code>\l__codedoc_in_function_bool</code>	<p>True when inside a <code>function</code> or <code>variable</code> environment. Used by the <code>syntax</code> environment to determine its behaviour.</p> <pre>15 \bool_new:N \l__codedoc_in_function_bool</pre> <p>(End of definition for <code>\l__codedoc_in_function_bool</code>.)</p>
<code>\l__codedoc_long_name_bool</code> <code>\l__codedoc_trial_width_dim</code>	<p>The boolean <code>\l__codedoc_long_name_bool</code> is true if the width <code>\l__codedoc_trial_width_dim</code> of the coffin <code>\l__codedoc_functions_coffin</code> (containing the current function names) is bigger than the space available in the margin.</p> <pre>16 \bool_new:N \l__codedoc_long_name_bool 17 \dim_new:N \l__codedoc_trial_width_dim</pre> <p>(End of definition for <code>\l__codedoc_long_name_bool</code> and <code>\l__codedoc_trial_width_dim</code>.)</p>
<code>\l__codedoc_nested_macro_int</code>	<p>The nesting of <code>macro</code> environments (this is now 0 outside a <code>macro</code> environment).</p> <pre>18 \int_new:N \l__codedoc_nested_macro_int</pre> <p>(End of definition for <code>\l__codedoc_nested_macro_int</code>.)</p>
<code>\l__codedoc_macro_tested_bool</code> <code>\g__codedoc_missing_tests_prop</code> <code>\g__codedoc_not_tested_seq</code> <code>\g__codedoc_testfiles_seq</code>	<p>A boolean describing whether the current macro has tests, and some global structures which contain information about test files and which tests are missing.</p> <pre>19 \bool_new:N \l__codedoc_macro_tested_bool 20 \prop_new:N \g__codedoc_missing_tests_prop 21 \seq_new:N \g__codedoc_not_tested_seq 22 \seq_new:N \g__codedoc_testfiles_seq</pre> <p>(End of definition for <code>\l__codedoc_macro_tested_bool</code> and others.)</p>
<code>\l__codedoc_macro_deprecated_bool</code> <code>\l__codedoc_macro_internal_bool</code> <code>\l__codedoc_macro_nodoc_bool</code> <code>\l__codedoc_macro_TF_bool</code> <code>\l__codedoc_macro_pTF_bool</code> <code>\l__codedoc_macro_noTF_bool</code> <code>\l__codedoc_macro_EXP_bool</code> <code>\l__codedoc_macro_rEXP_bool</code> <code>\l__codedoc_macro_var_bool</code> <code>\l__codedoc_override_module_tl</code> <code>\l__codedoc_macro_documented_tl</code>	<p>Contain information about some options of function/macro environments. We initialize <code>\l__codedoc_override_module_tl</code> to avoid overriding module names by an empty name (meaning no module).</p> <pre>23 \bool_new:N \l__codedoc_macro_deprecated_bool 24 \bool_new:N \l__codedoc_macro_internal_bool 25 \bool_new:N \l__codedoc_macro_nodoc_bool 26 \bool_new:N \l__codedoc_macro_TF_bool 27 \bool_new:N \l__codedoc_macro_pTF_bool 28 \bool_new:N \l__codedoc_macro_noTF_bool</pre>

```

29 \bool_new:N \l__codedoc_macro_EXP_bool
30 \bool_new:N \l__codedoc_macro_rEXP_bool
31 \bool_new:N \l__codedoc_macro_var_bool
32 \tl_new:N \l__codedoc_override_module_tl
33 \tl_set:Nn \l__codedoc_override_module_tl { \q_no_value }
34 \tl_new:N \l__codedoc_macro_documented_tl

```

(End of definition for `\l__codedoc_macro_deprecated_bool` and others.)

`\g__codedoc_lmodern_bool` Information about package options.

```

\g__codedoc_checkfunc_bool 35 \bool_new:N \g__codedoc_lmodern_bool
\g__codedoc_checktest_bool 36 \bool_new:N \g__codedoc_checkfunc_bool
\g__codedoc_cs_break_bool 37 \bool_new:N \g__codedoc_checktest_bool
\g__codedoc_show_notes_bool 38 \bool_new:N \g__codedoc_kernel_bool
\g__codedoc_kernel_bool 39 \bool_new:N \g__codedoc_cs_break_bool
40 \bool_new:N \g__codedoc_show_notes_bool
41 \bool_gset_true:N \g__codedoc_cs_break_bool

```

(End of definition for `\g__codedoc_lmodern_bool` and others.)

`\l__codedoc_tmpa_tl` Some temporary variables.

```

\l__codedoc_tmpb_tl 42 \tl_new:N \l__codedoc_tmpa_tl
\l__codedoc_tmpa_int 43 \tl_new:N \l__codedoc_tmpb_tl
\l__codedoc_tmpa_seq 44 \int_new:N \l__codedoc_tmpa_int
45 \int_new:N \l__codedoc_tmpa_seq

```

(End of definition for `\l__codedoc_tmpa_tl` and others.)

`\l__codedoc_names_block_tl` List of local sequence variables (produced through `__codedoc_lseq_name:n`), one for each set of variants in a `function` or `macro` environment. More precisely these sequences are named after the base forms, such as `\clist_count:n` or `\clist_count:N` (which are not variants). Each of these sequences have the base name (without any signature) as their first item, followed by the list of variant's signatures, or `\scan_stop:` to denote the absence of signature (no colon).

```

46 \tl_new:N \l__codedoc_names_block_tl

```

(End of definition for `\l__codedoc_names_block_tl`.)

`\g__codedoc_variants_seq` Stores rather temporarily the list of variants (signatures only) of a function/macro that is being documented. It is global because we need it to keep its value throughout cells of an alignment.

```

47 \seq_new:N \g__codedoc_variants_seq

```

(End of definition for `\g__codedoc_variants_seq`.)

`\l__codedoc_names_verb_bool` Set to **true** if the main argument of a macro/function environment should be used as is, without removing any comma or space.

```
48 \bool_new:N \l__codedoc_names_verb_bool
```

(End of definition for `\l__codedoc_names_verb_bool`.)

`\l__codedoc_names_seq` List of functions/environments/... appearing as arguments of a given **function** or **macro** environment. These are the names after conversion of `_@@` and `@@` to `__` (*module name*) and other sanitizing.

```
49 \seq_new:N \l__codedoc_names_seq
```

(End of definition for `\l__codedoc_names_seq`.)

`\g__codedoc_nested_names_seq` Collects all macros in nested **macro** environments, to use them in the “End definition” text.

```
50 \seq_new:N \g__codedoc_nested_names_seq
```

(End of definition for `\g__codedoc_nested_names_seq`.)

`\l__codedoc_index_macro_tl` When analyzing a control sequence found within a **macrocode** environment, `\l__codedoc_index_macro_tl` holds the control sequence (partially a string), `\l__codedoc_index_key_tl` holds the future sort key in the index, and `\l__codedoc_index_module_tl` is the subindex in which the control sequence should be listed. `\l__codedoc_index_internal_bool` indicates when the control sequence is internal and should be indexed in a slightly different subindex. Finally, `\l__codedoc_macro_do_not_index_tl` indicates control sequences which should not be indexed in a specific **macro** environment.

```
51 \tl_new:N \l__codedoc_index_macro_tl
```

```
52 \tl_new:N \l__codedoc_index_key_tl
```

```
53 \tl_new:N \l__codedoc_index_module_tl
```

```
54 \tl_new:N \l__codedoc_macro_do_not_index_tl
```

```
55 \bool_new:N \l__codedoc_index_internal_bool
```

(End of definition for `\l__codedoc_index_macro_tl` and others.)

`\g__codedoc_module_name_tl` The module name, set when reading a line `<@@=<module>`.

```
56 \tl_new:N \g__codedoc_module_name_tl
```

(End of definition for `\g__codedoc_module_name_tl`.)

<code>\c__codedoc_iow_rule_tl</code>	40 equal signs.
<code>\c__codedoc_iow_midrule_tl</code>	<pre> 57 \tl_const:Nn \c__codedoc_iow_rule_tl 58 { ===== } 59 \tl_const:Nn \c__codedoc_iow_mid_rule_tl 60 { ----- } </pre> <p>(End of definition for <code>\c__codedoc_iow_rule_tl</code> and <code>\c__codedoc_iow_midrule_tl</code>.)</p>
<code>\l__codedoc_macro_box</code>	A vertical box in which the names given to the macro environment are typeset, a
<code>\l__codedoc_macro_index_box</code>	horizontal box in which we store the targets created by indexing commands, and the
<code>\l__codedoc_macro_int</code>	number of macros so far (including those from surrounding macro environments).
	<pre> 61 \box_new:N \l__codedoc_macro_box 62 \box_new:N \l__codedoc_macro_index_box 63 \int_new:N \l__codedoc_macro_int </pre> <p>(End of definition for <code>\l__codedoc_macro_box</code>, <code>\l__codedoc_macro_index_box</code>, and <code>\l__codedoc_macro_int</code>.)</p>
<code>\l__codedoc_cmd_tl</code>	Variables used to control the behaviour of <code>\cmd</code> , <code>\cs</code> and <code>\tn</code> .
<code>\l__codedoc_cmd_index_tl</code>	
<code>\l__codedoc_cmd_module_tl</code>	
<code>\l__codedoc_cmd_noindex_bool</code>	
<code>\l__codedoc_cmd_replace_bool</code>	
	<pre> 64 \tl_new:N \l__codedoc_cmd_tl 65 \tl_new:N \l__codedoc_cmd_index_tl 66 \tl_new:N \l__codedoc_cmd_module_tl 67 \bool_new:N \l__codedoc_cmd_noindex_bool 68 \bool_new:N \l__codedoc_cmd_replace_bool </pre> <p>(End of definition for <code>\l__codedoc_cmd_tl</code> and others.)</p>
<code>\l__codedoc_in_implementation_bool</code>	This boolean is <code>true</code> within the <code>implementation</code> environment, and <code>false</code> anywhere else.
	<pre> 69 \bool_new:N \l__codedoc_in_implementation_bool </pre> <p>(End of definition for <code>\l__codedoc_in_implementation_bool</code>.)</p>
<code>\g__codedoc_typeset_documentation_bool</code>	These booleans control whether the documentation/implementation should be type-
<code>\g__codedoc_typeset_implementation_bool</code>	set. By default both should be.
	<pre> 70 \bool_new:N \g__codedoc_typeset_documentation_bool 71 \bool_new:N \g__codedoc_typeset_implementation_bool 72 \bool_set_true:N \g__codedoc_typeset_documentation_bool 73 \bool_set_true:N \g__codedoc_typeset_implementation_bool </pre> <p>(End of definition for <code>\g__codedoc_typeset_documentation_bool</code> and <code>\g__codedoc_typeset_implementation_bool</code>.)</p>

`\g__codedoc_base_name_tl` The name of the macro which is being documented (without its signature), and a property list mapping base forms of variants to all variants which have the same base form.

```
74 \tl_new:N \g__codedoc_base_name_tl
75 \prop_new:N \l__codedoc_variants_prop
```

(End of definition for `\g__codedoc_base_name_tl` and `\l__codedoc_variants_prop`.)

`\l__codedoc_function_label_clist` Option of a **function** environment which replaces the label that would normally be inserted by labels for the given list of control sequences. This is only useful to avoid duplicate labels when a function's documentation appears multiple times.

```
76 \clist_new:N \l__codedoc_function_label_clist
77 \bool_new:N \l__codedoc_no_label_bool
```

(End of definition for `\l__codedoc_function_label_clist` and `\l__codedoc_no_label_bool`.)

`\l__codedoc_date_added_tl` Values of some options of the **function** environment.

```
\l__codedoc_date_updated_tl 78 \tl_new:N \l__codedoc_date_added_tl
79 \tl_new:N \l__codedoc_date_updated_tl
```

(End of definition for `\l__codedoc_date_added_tl` and `\l__codedoc_date_updated_tl`.)

`\l__codedoc_macro_argument_tl` Save the argument of a **macro** or **function** environment for use in error messages.

```
80 \tl_new:N \l__codedoc_macro_argument_tl
```

(End of definition for `\l__codedoc_macro_argument_tl`.)

```
81 % \int_new:N \c@CodelineNo
```

5.2 Variants and helpers

`__codedoc_tmpa:w` Auxiliary macros for temporary use.

```
\__codedoc_tmpb:w 82 \cs_new_eq:NN \__codedoc_tmpa:w ?
83 \cs_new_eq:NN \__codedoc_tmpb:w ?
```

(End of definition for `__codedoc_tmpa:w` and `__codedoc_tmpb:w`.)

`\seq_set_split:NoV` A few missing variants.

```
\tl_to_str:f 84 \cs_generate_variant:Nn \seq_set_split:Nnn { NoV }
85 \cs_generate_variant:Nn \tl_to_str:n { f }
```

(End of definition for `\seq_set_split:NoV` and `\tl_to_str:f`. These functions are documented on page ??.)

`__codedoc_if_almost_str:nTF` Used to test if the argument of `\cmd` or other macros to be indexed is almost a string or not: for instance this is `false` if `#1` contains `\meta{...}`. The surprising `f`-expansion is there to cope with the case of `#1` starting with `\c_backslash_str` which should be expanded and considered to be “normal”.

```

86 \prg_new_protected_conditional:Npnn \__codedoc_if_almost_str:n #1 { TF , T , F }
87   {
88     \int_compare:nNnTF
89       { \tl_count:n {#1} }
90       < { \tl_count:e { \tl_to_str:f {#1} } }
91       { \prg_return_false: }
92       { \prg_return_true: }
93   }
94 \prg_generate_conditional_variant:Nnn \__codedoc_if_almost_str:n { V } { T }

```

(End of definition for `__codedoc_if_almost_str:nTF`.)

`__codedoc_trim_right:Nn` Removes all material after `#2` in the token list variable `#1`. Perhaps combine with `__codedoc_trim_right:No` `__codedoc_key_trim_module:n?`

```

95 \cs_new_protected:Npn \__codedoc_trim_right:Nn #1#2
96   {
97     \cs_set:Npn \__codedoc_tmp:w ##1 #2 ##2 \q_stop { \exp_not:n {##1} }
98     \__kernel_tl_set:Ne #1 { \exp_after:wN \__codedoc_tmp:w #1 #2 \q_stop }
99   }
100 \cs_generate_variant:Nn \__codedoc_trim_right:Nn { No }

```

(End of definition for `__codedoc_trim_right:Nn`.)

`__codedoc_str_if_begin:nnTF` True if the first string starts with the second.

```

101 \prg_new_protected_conditional:Npnn \__codedoc_str_if_begin:nn #1#2 { TF , T , F }
102   {
103     \tl_if_in:ooTF
104       { \exp_after:wN \scan_stop: \tl_to_str:n {#1} }
105       { \exp_after:wN \scan_stop: \tl_to_str:n {#2} }
106       { \prg_return_true: }
107       { \prg_return_false: }
108   }
109 \prg_generate_conditional_variant:Nnn \__codedoc_str_if_begin:nn
110   { oo } { TF , T , F }

```

(End of definition for `__codedoc_str_if_begin:nnTF`.)

`__codedoc_replace_at_at:N` The goal is to replace @@ by the current module name. We take advantage of this function to also detect internal macros. If there is no *<module name>*, do nothing. Otherwise, sanitize the catcodes of @ and _, temporarily change @@@@ to aa with different catcodes and later to @@, and replace __@@ and _@@ and @@ by __*<module name>*. The result contains _ with category code letter because this is what the `macrocode` environment expects. Other use cases can apply `\tl_to_str:n` if needed. Note that we include spaces between the @ in the code below, since it is also processed through the same replacement rules.

```

111 \cs_new_protected:Npn \__codedoc_replace_at_at:N #1
112 {
113   \tl_if_empty:NF \g__codedoc_module_name_tl
114   {
115     \exp_args:NNo \__codedoc_replace_at_at_aux:Nn
116     #1 \g__codedoc_module_name_tl
117   }
118 }
119 \cs_new_protected:Npe \__codedoc_replace_at_at_aux:Nn #1#2
120 {
121   \tl_replace_all:Nnn #1 { \token_to_str:N @ } { @ }
122   \tl_replace_all:Nnn #1 { \token_to_str:N _ } { _ }
123   \tl_replace_all:Nnn #1 { @ @ @ @ } { \token_to_str:N a a }
124   \tl_replace_all:Nnn #1 { _ _ @ @ } { _ _ #2 }
125   \tl_replace_all:Nnn #1 { _ _ @ @ } { _ _ #2 }
126   \tl_replace_all:Nnn #1 { @ @ } { _ _ #2 }
127   \tl_replace_all:Nnn #1 { \token_to_str:N a a } { @ @ }
128 }

```

(End of definition for `__codedoc_replace_at_at:N` and `__codedoc_replace_at_at_aux:Nn`.)

`__codedoc_detect_internals:N` After splitting at each __ and removing the leading item from the sequence (since it does not follow __), remove everything after any space or end-of-line to get a good approximation of the control sequence (for the warning message). Then check if that starts with something allowed: @@ module name and : or _, or if the relevant boolean is set `kernel_` (it seems safe to assume we will not define a `__kernel:...` command). For the message itself remove anything after any _ or : (with either catcode) to get a guess of the module name.

```

129 \cs_new_protected:Npn \__codedoc_detect_internals:N #1
130 {
131   \bool_if:NT \l__codedoc_detect_internals_bool
132   { \__codedoc_detect_internals_aux:N #1 }

```

```

133 }
134 \group_begin:
135 \char_set_catcode_active:N \^^M
136 \cs_new_protected:Npn \__codedoc_detect_internals_aux:N #1
137 {
138 \tl_set_eq:NN \l__codedoc_detect_internals_tl #1
139 \tl_replace_all:NVn \l__codedoc_detect_internals_tl \c_underscore_str { _ }
140 \seq_set_split:NnV \l__codedoc_tmpa_seq { _ _ } \l__codedoc_detect_internals_tl
141 \seq_pop_left:NN \l__codedoc_tmpa_seq \l__codedoc_detect_internals_tl
142 \seq_map_variable:NNn \l__codedoc_tmpa_seq \l__codedoc_detect_internals_tl
143 {
144 \__codedoc_trim_right:No \l__codedoc_detect_internals_tl
145 \c_catcode_active_space_tl
146 \__codedoc_trim_right:Nn \l__codedoc_detect_internals_tl ^^M
147 \__codedoc_if_detect_internals_ok:NF \l__codedoc_detect_internals_tl
148 {
149 \tl_set_eq:NN \l__codedoc_detect_internals_cs_tl \l__codedoc_detect_internals_
150 \__codedoc_trim_right:Nn \l__codedoc_detect_internals_tl _
151 \__codedoc_trim_right:Nn \l__codedoc_detect_internals_tl :
152 \__codedoc_trim_right:No \l__codedoc_detect_internals_tl { \token_to_str:N : }
153 \msg_warning:nneee { l3doc } { foreign-internal }
154 { \tl_to_str:N \l__codedoc_detect_internals_cs_tl }
155 { \tl_to_str:N \l__codedoc_detect_internals_tl }
156 { \tl_to_str:N \g__codedoc_module_name_tl }
157 }
158 }
159 }
160 \group_end:
161 \prg_new_protected_conditional:Npnn \__codedoc_if_detect_internals_ok:N #1 { F }
162 {
163 \__codedoc_str_if_begin:ooTF {#1} { \g__codedoc_module_name_tl _ }
164 { \prg_return_true: }
165 {
166 \__codedoc_str_if_begin:ooTF {#1} { \g__codedoc_module_name_tl : }
167 { \prg_return_true: }
168 {
169 \bool_if:NTF \g__codedoc_kernel_bool
170 {
171 \__codedoc_str_if_begin:ooTF {#1} { kernel _ }
172 { \prg_return_true: }
173 { \prg_return_false: }
174 }

```

```

175         { \prg_return_false: }
176     }
177 }
178 }

```

(End of definition for `__codeloc_detect_internals:N`, `__codeloc_detect_internals_aux:N`, and `__codeloc_if_detect_internals_ok:NF`.)

`__codeloc_signature_base_form:n` Expands to the “base form” of the signature. For instance, given `noxcfvV` it would obtain `nnnNnnn`, or given `ow` it would obtain `nw`. The loop stops at the first token that is not recognized; the rest is enclosed in `\exp_not:n`.

```

179 \cs_new:Npn \__codeloc_signature_base_form:n #1
180 { \__codeloc_signature_base_form_aux:n #1 \q_stop }
181 \cs_new:Npn \__codeloc_signature_base_form_aux:n #1
182 {
183   \str_case:nnTF {#1}
184   {
185     { N } { N }
186     { c } { N }
187     { n } { n }
188     { o } { n }
189     { f } { n }
190     { e } { n }
191     { x } { n }
192     { V } { n }
193     { v } { n }
194   }
195   { \__codeloc_signature_base_form_aux:n }
196   { \__codeloc_signature_base_form_aux:w #1 }
197 }
198 \cs_new:Npn \__codeloc_signature_base_form_aux:w #1 \q_stop
199 { \exp_not:n {#1} }

```

(End of definition for `__codeloc_signature_base_form:n`, `__codeloc_signature_base_form_aux:n`, and `__codeloc_signature_base_form_aux:w`.)

`__codeloc_predicate_from_base:n` Get predicate from a function’s base name. The code is not broken by functions with no signature. The `n`-type version can be used for keys and other non-control sequences. The output after `e`-expansion is a string.

```

200 \cs_new:Npn \__codeloc_predicate_from_base:n #1
201 {
202   \__codeloc_get_function_name:n {#1}
203   \tl_to_str:n { _p: }

```

```

204     \_codedoc_get_function_signature:n {#1}
205 }

```

(End of definition for _codedoc_predicate_from_base:n.)

```

\_codedoc_split_function_do:nn Similar to internal functions defined in l3basics, but here we operate on strings di-
\_codedoc_split_function_do:on rectly rather than control sequences.
\_codedoc_get_function_name:n 206 \cs_new:Npn \_codedoc_get_function_name:n #1
\_codedoc_get_function_signature:n 207 { \_codedoc_split_function_do:nn {#1} { \use_i:nnn } }
\_codedoc_split_function_auxi:w 208 \cs_new:Npn \_codedoc_get_function_signature:n #1
\_codedoc_split_function_auxii:w 209 { \_codedoc_split_function_do:nn {#1} { \use_ii:nnn } }
210 \cs_set_protected:Npn \_codedoc_tmpa:w #1
211 {
212     \cs_new:Npn \_codedoc_split_function_do:nn ##1
213     {
214         \exp_after:wN \_codedoc_split_function_auxi:w
215         \tl_to_str:n {##1} \q_mark \c_true_bool
216         #1 \q_mark \c_false_bool
217         \q_stop
218     }
219     \cs_new:Npn \_codedoc_split_function_auxi:w
220     ##1 #1 ##2 \q_mark ##3##4 \q_stop ##5
221     { \_codedoc_split_function_auxii:w {##5} ##1 \q_mark \q_stop {##2} ##3 }
222     \cs_new:Npn \_codedoc_split_function_auxii:w
223     ##1##2 \q_mark ##3 \q_stop
224     { ##1 {##2} }
225 }
226 \exp_args:No \_codedoc_tmpa:w { \token_to_str:N : }
227 \cs_generate_variant:Nn \_codedoc_split_function_do:nn { o }

```

(End of definition for _codedoc_split_function_do:nn and others.)

_codedoc_key_get_base:nN Get the base form of a function and store it. As part of getting the base form, change trailing T or F to TF, skipping that change if the function contains no colon to avoid changing for instance some names ending in PDF or similar. The various letters z serve as end-delimiters different from any outcome of \tl_to_str:n.

```

228 \cs_new_protected:Npn \_codedoc_key_get_base:nN #1#2
229 {
230     \_codedoc_if_almost_str:nTF {#1}
231     {
232         \_codedoc_key_get_base_TF:nN {#1} \l__codedoc_tmpa_tl
233         \_kernel_tl_set:Ne #2

```

```

234         { \__codedoc_split_function_do:on \l__codedoc_tmpa_tl { \__codedoc_base_form_aux:n
235     }
236     { \tl_set:Nn #2 {#1} }
237 }
238 \cs_new:Npe \__codedoc_key_get_base_TF:nN #1#2
239 {
240     \__kernel_tl_set:Ne #2 { \exp_not:N \tl_to_str:n {#1} }
241     \tl_if_in:NoF #2 { \tl_to_str:n {:} }
242     { \exp_not:N \prg_break: }
243     \tl_if_in:onT { #2 z } { \tl_to_str:n {TF} z }
244     { \exp_not:N \prg_break: }
245     \tl_if_in:onT { #2 z } { \tl_to_str:n {T} z }
246     {
247         \tl_put_right:Nn #2 { \tl_to_str:n {F} }
248         \exp_not:N \prg_break:
249     }
250     \tl_if_in:onT { #2 z } { \tl_to_str:n {F} z }
251     {
252         \tl_put_right:Nn #2 { z }
253         \tl_replace_once:Nnn #2 { \tl_to_str:n {F} z } { \tl_to_str:n {TF} }
254         \exp_not:N \prg_break:
255     }
256     \exp_not:N \prg_break_point:
257 }
258 \cs_new:Npn \__codedoc_base_form_aux:nnN #1#2#3
259 {
260     \exp_not:n {#1}
261     \bool_if:NT #3
262     {
263         \token_to_str:N :
264         \bool_lazy_or:nnTF
265             { \str_if_eq_p:nn { #1 ~ } { \exp_args } }
266             { \str_if_eq_p:nn { #1 ~ } { \exp_last_unbraced } }
267         { \exp_not:n {#2} }
268         { \__codedoc_signature_base_form:n {#2} }
269     }
270 }

```

(End of definition for __codedoc_key_get_base:nN.)

__codedoc_base_form_signature_do:nnn Do #2{#1} if there is no signature, or if #1 contains two colons in a row (this covers the weird function \::N and so on). Otherwise apply #3 with the following two

arguments: the base form of #1, and the original signature with an extra pair of braces.

```

271 \cs_new_protected:Npn \__codedoc_base_form_signature_do:nnn #1#2#3
272 {
273   \__codedoc_split_function_do:nn {#1}
274   { \__codedoc_base_form_aux:nnnnnN {#1} {#2} {#3} }
275 }
276 \cs_new_protected:Npn \__codedoc_base_form_aux:nnnnnN #1#2#3#4#5#6
277 {
278   \bool_if:NTF #6
279   {
280     \tl_if_head_eq_charcode:nNTF {#4} :
281     { #2 {#1} }
282     {
283       \use:e
284       {
285         \exp_not:n {#3}
286         { \__codedoc_base_form_aux:nnN {#4} {#5} #6 }
287       }
288       {#4} {#5}
289     }
290   }
291   { #2 {#1} }
292 }

```

(End of definition for `__codedoc_base_form_signature_do:nnn`.)

`__codedoc_date_compare_p:nNn` Expects #1 and #3 to be dates in the format YYYY-MM-DD (but accepts YYYY or YYYY-MM too). Compares them using #2 (one of <, =, >), filling in zeros for missing data.

```

\__codedoc_date_compare_aux:nnnNnnn
\__codedoc_date_compare_aux:w
293 \prg_new_conditional:Npnn \__codedoc_date_compare:nNn #1#2#3 { TF , T , F , p }
294 { \__codedoc_date_compare_aux:w #1--- \q_mark #2 #3--- \q_stop }
295 \cs_new:Npn \__codedoc_date_compare_aux:w
296   #1 - #2 - #3 - #4 \q_mark #5 #6 - #7 - #8 - #9 \q_stop
297 {
298   \__codedoc_date_compare_aux:nnnNnnn
299   { \tl_if_empty:nTF {#1} { 0 } {#1} }
300   { \tl_if_empty:nTF {#2} { 0 } {#2} }
301   { \tl_if_empty:nTF {#3} { 0 } {#3} }
302   #5
303   { \tl_if_empty:nTF {#6} { 0 } {#6} }
304   { \tl_if_empty:nTF {#7} { 0 } {#7} }

```



```

305     { \tl_if_empty:nTF {#8} { 0 } {#8} }
306   }
307   \cs_new:Npn \__codedoc_date_compare_aux:nnnNnnn #1#2#3#4#5#6#7
308   {
309     \int_compare:nNnTF {#1} = {#5}
310     {
311       \int_compare:nNnTF {#2} = {#6}
312       {
313         \int_compare:nNnTF {#3} #4 {#7}
314         { \prg_return_true: } { \prg_return_false: }
315       }
316       {
317         \int_compare:nNnTF {#2} #4 {#6}
318         { \prg_return_true: } { \prg_return_false: }
319       }
320     }
321     {
322       \int_compare:nNnTF {#1} #4 {#5}
323       { \prg_return_true: } { \prg_return_false: }
324     }
325     \use_none:n
326     \q_stop
327   }

```

(End of definition for `__codedoc_date_compare:nNnTF`, `__codedoc_date_compare_aux:nnnNnnn`, and `__codedoc_date_compare_aux:w`.)

`__codedoc_gprop_name:n` We need to keep track of some information about control sequences (and other strings) that are being (or have been) documented. Some is stored into global props and some into local seqs, whose name does not follow conventions: it is `\g__codedoc` or `\l__codedoc` followed by a space and by the string, which can be arbitrary. We cannot reasonably use a single big prop for speed reasons.

```

328   \cs_new:Npn \__codedoc_gprop_name:n #1 { g__codedoc ~ \tl_to_str:n {#1} }
329   \cs_new:Npn \__codedoc_lseq_name:n #1 { l__codedoc ~ \tl_to_str:n {#1} }

```

(End of definition for `__codedoc_gprop_name:n` and `__codedoc_lseq_name:n`.)

5.3 Messages

```

330   \msg_new:nnnn { l3doc } { no-signature-TF }
331   { Function/macro~'#1'~cannot~be~turned~into~a~conditional. }
332   {
333     A~function~or~macro~environment~with~option~pTF,~TF~or~noTF~

```

```

334     received~the~argument~'#1'.~This~function's~name~has~no~
335     ':'~hence~it~is~not~clear~where~to~add~'_p'~or~'TF'.~
336     Please~follow~expl3~naming~conventions.
337 }
338 \msg_new:nnn { l3doc } { date-format }
339 { The~date~'#1'~should~be~given~in~YYYY-MM-DD~format. }
340 \msg_new:nnn { l3doc } { future-date }
341 { The~added/updated~date~'#2'~of~'#1'~is~in~the~future. }
342 \msg_new:nnn { l3doc } { syntax-nested-function }
343 {
344     The~'syntax'~environment~should~be~used~in~the~
345     innermost~'function'~environment.
346 }
347 \msg_new:nnn { l3doc } { multiple-syntax }
348 {
349     The~'syntax'~environment~should~only~be~used~once~in~
350     a~'function'~environment.
351 }
352 \msg_new:nnn { l3doc } { deprecated-option }
353 { The~option~'#1'~has~been~deprecated~for~'#2'. }
354 \msg_new:nnn { l3doc } { foreign-internal }
355 {
356     A~control~sequence~of~the~form~'..._#1'~was~used.~
357     It~should~only~be~used~in~the~module~'#2'
358     \tl_if_empty:nF {#3} { ,~not~in~'#3' } .
359 }

```

5.4 Options and configuration

```

360 \DeclareKeys [ l3doc / options ]
361 {
362     a5paper .code:n = \@latexerr { Option~not~supported } { } ,
363     full .code:n =
364     {
365         \bool_gset_true:N \g__codedoc_typeset_documentation_bool
366         \bool_gset_true:N \g__codedoc_typeset_implementation_bool
367     } ,
368     onlydoc .code:n =
369     {
370         \bool_gset_true:N \g__codedoc_typeset_documentation_bool
371         \bool_gset_false:N \g__codedoc_typeset_implementation_bool
372     } ,
373     check .bool_gset:N = \g__codedoc_checkfunc_bool ,

```

```

374     checktest .bool_gset:N = \g__codeloc_checktest_bool ,
375     kernel .bool_gset:N = \g__codeloc_kernel_bool ,
376     stdmodule .bool_gset_inverse:N = \g__codeloc_kernel_bool ,
377     lm-default .bool_gset:N = \g__codeloc_lmodern_bool ,
378     cs-break .bool_gset_inverse:N = \g__codeloc_cs_break_bool ,
379     cs-break-nohyphen .code:n = \PassOptionsToPackage{nohyphen}{underscore} ,
380     show-notes .bool_gset:N = \g__codeloc_show_notes_bool,
381     hide-notes .bool_gset_inverse:N = \g__codeloc_show_notes_bool
382 }

383 \DeclareUnknownKeyHandler [ l3doc / options ]
384 { \PassOptionsToClass { \CurrentOption } { article } }
385 \SetKeys [ l3doc / options ]
386 { full , kernel , check = false , checktest = false , lm-default }
387 \PassOptionsToClass { a4paper } { article }

```

Input a local configuration file, if it exists, with a message to the console that this has happened. Since we distribute a `.cfg` file with the class, this should usually always be true. Therefore, check for `\ExplMakeTitle` (defined in “our” `.cfg` file) and only output the informational message if it’s not found.

```

388 \msg_new:nnn { l3doc } { input-cfg }
389 { Local-config-file~l3doc.cfg~loaded. }
390 \file_if_exist:nT { l3doc.cfg }
391 {
392     \file_input:n { l3doc.cfg }
393     \cs_if_exist:cF { ExplMakeTitle }
394     { \msg_info:nn { l3doc } { input-cfg } }
395 }

396 \ProcessKeyOptions [ l3doc / options ]

```

5.5 Class and package loading

```

397 \LoadClass{article}
398 \RequirePackage{doc}
399 \RequirePackage
400 {
401     array,
402     alphalph,
403     amsmath,
404     amssymb,
405     booktabs,
406     color,
407     colortbl,
408     hologo,

```

```

409     enumitem,
410     pifont,
411     textcomp,
412     trace,
413     csquotes,
414     fancyvrb,
415     underscore,
416     verbatim
417 }
418 \raggedbottom

```

Depending on the option, load the package `lmodern` to set the font. Then replace the italic typewriter font with the oblique shape instead; the former makes my skin crawl. (Will, Aug 2011)

```

419 \bool_if:NT \g__codedoc_lmodern_bool
420 {
421     \RequirePackage[T1]{fontenc}
422     \RequirePackage{lmodern}
423     \group_begin:
424         \ttfamily
425         \DeclareFontShape{T1}{lmtt}{m}{it}{<->ec-lmtto10}{ }
426     \group_end:
427 }

```

Must be last, as usual.

```

428 \RequirePackage{hypdoc}

```

5.6 Configuration and tweaks

\MakePrivateLetters A few more letters are “private” in a L^AT_EX3 programming environment.

```

429 \cs_gset:Npn \MakePrivateLetters
430 {
431     \char_set_catcode_letter:N \@
432     \char_set_catcode_letter:N \_
433     \char_set_catcode_letter:N \:
434 }

```

(End of definition for \MakePrivateLetters. This function is documented on page ??.)

CodelineNo Some configurations which have to do with line numbering.

```

435 \setcounter{StandardModuleDepth}{1}
436 \@addtoreset{CodelineNo}{part}
437 \tl_replace_once:Nnn \theCodelineNo
438 { \HDorg@theCodelineNo }
439 { \textcolor[gray]{0.5} { \sffamily\tiny\arabic{CodelineNo} } }

```

(End of definition for `CodelineNo`. This function is documented on page ??.)

`\verbatim` In `.dtx` documents, the `verbatim` environment adds extra space because it only
`\endverbatim` removes the first “%” sign, and not the indentation (typically a space). Fix it with
`fancyvrb`:

```
440 \fvset{gobble=2}  
441 \cs_gset_eq:NN \verbatim \Verbatim  
442 \cs_gset_eq:NN \endverbatim \endVerbatim
```

(End of definition for `\verbatim` and `\endverbatim`. These functions are documented on page ??.)

`\ifnot@excluded` This function tests whether a macro name stored in `\macro@namepart` was excluded
from indexing by `\DoNotIndex`. Rather than trying to fix catcodes that come into
here, turn everything to string catcodes. This is slightly inefficient as we could have
ensured that `\index@excludelist` has string catcodes in the first place.

```
443 \cs_set_protected:Npn \ifnot@excluded  
444 {  
445   \exp_args:Nee \expanded@notin  
446     { \c_backslash_str \tl_to_str:N \macro@namepart , }  
447     { \exp_args:NV \tl_to_str:n \index@excludelist }  
448 }
```

(End of definition for `\ifnot@excluded`. This function is documented on page ??.)

`\pdfstringnewline` We avoid some hyperref warnings by making `\\` (almost) trivial in bookmarks: more
`_codedoc_pdfstring_newline:w` precisely it might be used with a star and an optional argument, which we thus
remove using an `ltxcmd` expandable command. Since there cannot be trailing optional
arguments, pick up an extra mandatory one and put it back.

```
449 \cs_new:Npn \pdfstringnewline { : ~ }  
450 \DeclareExpandableDocumentCommand  
451 { \_codedoc_pdfstring_newline:w } { s o m } { \pdfstringnewline #3 }  
452 \pdfstringdefDisableCommands  
453 { \cs_set_eq:NN \\ \_codedoc_pdfstring_newline:w }
```

(End of definition for `\pdfstringnewline` and `_codedoc_pdfstring_newline:w`. This function is documented
on page ??.)

5.7 Design

Increase the text width slightly so that width the standard fonts 72 columns of
code may appear in a `macrocode` environment. Increase the `marginpar` width slightly,
for long command names. And increase the left margin by a similar amount.

```

454 \setlength \textwidth { 385pt }
455 \addtolength \marginparwidth { 30pt }
456 \addtolength \oddsidemargin { 20pt }
457 \addtolength \evensidemargin { 20pt }

```

(These were introduced when `article` was the documentclass, but I’ve left them here for now to remind me to do something about them later.)

`\list` Customise lists.

```

\__codedoc_oldlist:nn 458 \cs_new_eq:NN \__codedoc_oldlist:nn \list
459 \cs_gset:Npn \list #1 #2
460 { \__codedoc_oldlist:nn {#1} { #2 \dim_zero:N \listparindent } }
461 \setlength \parindent { 2em }
462 \setlength \itemindent { 0pt }
463 \setlength \parskip { 0pt plus 3pt minus 0pt }

```

(End of definition for `\list` and `__codedoc_oldlist:nn`. This function is documented on page ??.)

`\partname` Use “File” as a name in Part titles.

```

464 \tl_gset:Nn \partname {File}

```

(End of definition for `\partname`. This function is documented on page ??.)

`\l@section` Customise the table of contents (as we have so many sections). Different design

`\l@subsection` and/or structure is called for).

```

465 \@addtoreset{section}{part}
466 \cs_gset:Npn \l@section #1#2
467 {
468   \ifnum \c@tocdepth >\z@
469     \addpenalty\@secpenalty
470     \addvspace{1.0em \@plus\p@}
471     \setlength\@tempdima{2.5em} % was 1.5em
472     \begingroup
473       \parindent \z@ \rightskip \@pnumwidth
474       \parfillskip -\@pnumwidth
475       \leavevmode \bfseries
476       \advance\leftskip\@tempdima
477       \hskip -\leftskip
478       #1\nobreak\hfil \nobreak\hb@xt@\@pnumwidth{\hss #2}\par
479     \endgroup
480   \fi
481 }
482 \cs_gset:Npn \l@subsection
483 { \@dottedtocline{2}{2.5em}{2.3em} } % #2 = 1.5em

```

(End of definition for `\l@section` and `\l@subsection`. These functions are documented on page ??.)

5.8 Text markup

Make `|` and `"` be “short verb” characters, but not in the document preamble, where an active character may interfere with packages that are loaded. Remove these short-hands at the end of the document before reading the `.aux` file, as they may appear in labels (for instance, `l3fp` documents an operation `||`).

```

484 \AtBeginDocument
485 {
486   \MakeShortVerb \"
487   \MakeShortVerb \|
488 }
489 \AtEndDocument
490 {
491   \DeleteShortVerb \"
492   \DeleteShortVerb \|
493 }
```

```

\TeX Some commands for logos.
\IniTeX 494 \providecommand*\eTeX{\hologo{eTeX}}
\Lua 495 \providecommand*\IniTeX{\hologo{iniTeX}}
\LuaTeX 496 \providecommand*\Lua{Lua}
\pdfTeX 497 \providecommand*\LuaTeX{\hologo{LuaTeX}}
\XeTeX 498 \providecommand*\pdfTeX{\hologo{pdfTeX}}
\pTeX 499 \providecommand*\XeTeX{\hologo{XeTeX}}
\upTeX 500 \providecommand*\pTeX{\p\kern-.2em\hologo{TeX}}
\upTeX 501 \providecommand*\upTeX{\up\kern-.2em\hologo{TeX}}
\epTeX 502 \providecommand*\epTeX{${\varepsilon}$-\pTeX}
\epTeX 503 \providecommand*\eupTeX{${\varepsilon}$-\upTeX}
\epTeX 504 \providecommand*\ConTeXt{\hologo{ConTeXt}}
```

(End of definition for `\eTeX` and others. These functions are documented on page ??.)

`\cmd` They rely on a common auxiliary `_codedoc_cmd:nn` which receives as arguments
`\cs` the options and some tokens whose string representation starts with a backslash (to
`\tn` support cases such as `\cs{pkg_\ldots}`, we do not turn the whole argument into a string).

```

505 \DeclareDocumentCommand \cmd { 0 } m }
506 { \_codedoc_cmd:no {#1} { \token_to_str:N #2 } }
507 \DeclareDocumentCommand \cs { 0 } m }
```

```

508 { \_codeloc_cmd:no {#1} { \c_backslash_str #2 } }
509 \DeclareDocumentCommand \tn { 0{ } m }
510 {
511   \_codeloc_cmd:no
512     { module = TeX , replace = false , #1 }
513     { \c_backslash_str #2 }
514 }

```

(End of definition for `\cmd`, `\cs`, and `\tn`. These functions are documented on page 7.)

\meta A document-level command.

```

515 \DeclareDocumentCommand \meta { m }
516 { \_codeloc_meta:n {#1} }

```

(End of definition for `\meta`. This function is documented on page 7.)

_codeloc_pdfstring_cmd:w To work within a bookmark, these commands must be expandable.

```

\_codeloc_pdfstring_cs:w 517 \DeclareExpandableDocumentCommand
\_codeloc_pdfstring_meta:w 518 { \_codeloc_pdfstring_cmd:w } { o m } { \token_to_str:N #2 }
519 \DeclareExpandableDocumentCommand
520 { \_codeloc_pdfstring_cs:w } { o m } { \textbackslash \tl_to_str:n {#2} }
521 \cs_new:Npn \_codeloc_pdfstring_meta:w #1
522 { < \tl_to_str:n {#1} > }
523 \pdfstringdefDisableCommands
524 {
525   \cs_set_eq:NN \cmd \_codeloc_pdfstring_cmd:w
526   \cs_set_eq:NN \cs \_codeloc_pdfstring_cs:w
527   \cs_set_eq:NN \tn \_codeloc_pdfstring_cs:w
528   \cs_set_eq:NN \meta \_codeloc_pdfstring_meta:w
529 }

```

(End of definition for `_codeloc_pdfstring_cmd:w`, `_codeloc_pdfstring_cs:w`, and `_codeloc_pdfstring_meta:w`.)

\Arg `\marg{text}` prints `{<text>}`, “mandatory argument”.

\marg `\oarg{text}` prints `[<text>]`, “optional argument”.

\oarg `\parg{te,xt}` prints `(<te,xt>)`, “picture mode argument”. Finally, **\Arg** is the same

\parg as **\marg**.

```

530 \newcommand\Arg[1]
531 { \texttt{\char`{}\ \meta{#1} \texttt{\char`{}} }
532 \providecommand\marg[1]{ \Arg{#1} }
533 \providecommand\oarg[1]{ \texttt[ \meta{#1} \texttt ] }
534 \providecommand\parg[1]{ \texttt( \meta{#1} \texttt ) }

```


(End of definition for \Arg and others. These functions are documented on page 7.)

`\file` This list may change...this is just my preference for markup.

```
\env 535 \DeclareRobustCommand \file {\nolinkurl}
\pkg 536 \DeclareRobustCommand \env {\texttt}
\cls 537 \DeclareRobustCommand \pkg {\textsf}
      538 \DeclareRobustCommand \cls {\textsf}
```

(End of definition for \file and others. These functions are documented on page 8.)

`\EnableDocumentation` Control whether to typeset the documentation/implementation or not. These simply
`\EnableImplementation` set two switches.

```
\DisableDocumentation 539 \NewDocumentCommand \EnableDocumentation { }
\DisableImplementation 540 { \bool_gset_true:N \g__codedoc_typeset_documentation_bool }
                        541 \NewDocumentCommand \EnableImplementation { }
                        542 { \bool_gset_true:N \g__codedoc_typeset_implementation_bool }
                        543 \NewDocumentCommand \DisableDocumentation { }
                        544 { \bool_gset_false:N \g__codedoc_typeset_documentation_bool }
                        545 \NewDocumentCommand \DisableImplementation { }
                        546 { \bool_gset_false:N \g__codedoc_typeset_implementation_bool }
```

(End of definition for \EnableDocumentation and others. These functions are documented on page ??.)

`documentation (env.)` If the documentation/implementation should be typeset, then simply set the boolean
`implementation (env.)` `\l__codedoc_in_implementation_bool` which indicates whether we are within the
implementation section. Otherwise use `\comment` (and a paired `\endcomment`).

```
547 \NewDocumentEnvironment { documentation } { }
548 {
549   \bool_if:NTF \g__codedoc_typeset_documentation_bool
550     { \bool_set_false:N \l__codedoc_in_implementation_bool }
551     { \comment }
552 }
553 { \bool_if:NF \g__codedoc_typeset_documentation_bool { \endcomment } }
554 \NewDocumentEnvironment { implementation } { }
555 {
556   \bool_if:NTF \g__codedoc_typeset_implementation_bool
557     { \bool_set_true:N \l__codedoc_in_implementation_bool }
558     { \comment }
559 }
560 { \bool_if:NF \g__codedoc_typeset_implementation_bool { \endcomment } }
```

variable (*env.*) The **variable** environment behaves as a **function** or **macro** environment depending on the part of the document.

```
561 \DeclareDocumentEnvironment { variable } { 0{} +v }
562 {
563   \bool_if:NTF \l__codedoc_in_implementation_bool
564     { \__codedoc_macro:nnw { var , #1 } {#2} }
565     { \__codedoc_function:nnw {#1} {#2} }
566 }
567 {
568   \bool_if:NTF \l__codedoc_in_implementation_bool
569     { \__codedoc_macro_end: }
570     { \__codedoc_function_end: }
571 }
```

function (*env.*) Environment for documenting function(s), and environment for documenting the **macro** (*env.*) implementation of a macro.

```
572 \DeclareDocumentEnvironment { function } { 0{} +v }
573 { \__codedoc_function:nnw {#1} {#2} }
574 { \__codedoc_function_end: }
575 \DeclareDocumentEnvironment { macro } { 0{} +v }
576 { \__codedoc_macro:nnw {#1} {#2} }
577 { \__codedoc_macro_end: }
```

syntax (*env.*) Syntax block placed next to the list of functions to illustrate their use. TODO: test that the **syntax** environment is only used inside the **function** environment, and that it only appears once.

```
578 \NewDocumentEnvironment { syntax } { }
579 { \__codedoc_syntax:w }
580 {
581   \__codedoc_syntax_end:
582   \ignorespacesafterend
583 }
```

texnote (*env.*) Used to describe information destined to T_EX experts only.

```
584 \NewDocumentEnvironment { texnote } { }
585 {
586   \endgraf
587   \vspace{3mm}
588   \small\textbf{\TeX-hackers-note:}
589 }
590 {
```

```

591     \vspace{3mm}
592 }

```

arguments (*env.*) This environment is designed to be used within a **macro** environment to describe the arguments of the macro/function.

```

593 \NewDocumentEnvironment { arguments } { }
594 {
595     \enumerate [
596         nolistsep ,
597         label = \texttt{\#\arabic*} ~ : ,
598         labelsep = * ,
599     ]
600 }
601 {
602     \endenumerate
603 }

```

\CodedocExplain Explanation of stars and TF notations, for use in third-party packages.

```

\CodedocExplainEXP 604 \NewDocumentCommand { \CodedocExplain } { }
\CodedocExplainREXP 605 { \CodedocExplainEXP \ \CodedocExplainREXP \ \CodedocExplainTF }
\CodedocExplainTF 606 \NewDocumentCommand { \CodedocExplainEXP } { }
607 {
608     \raisebox{\baselineskip}[0pt][0pt]{\hypertarget{expstar}{}}%
609     \write \@auxout { \def \string \Codedoc@expstar { } }
610     \__codedoc_typeset_exp:\ indicates~fully~expandable~functions,~which~
611     can~be~used~within~an~\texttt{e}-type~argument~(inside~an~\tn{expanded}),~
612     \texttt{x}-type~argument~(in~plain~\TeX{}~terms,~inside~an~\tn{edef}),~
613     as~well~as~within~an~\texttt{f}-type~argument.
614 }
615 \NewDocumentCommand { \CodedocExplainREXP } { }
616 {
617     \raisebox{\baselineskip}[0pt][0pt]{\hypertarget{rexpstar}{}}%
618     \write \@auxout { \def \string \Codedoc@rexpstar { } }
619     \__codedoc_typeset_rexp:\ indicates~
620     restricted~expandable~functions,~which~can~be~used~within~an~
621     \texttt{x}-type~argument~or~an~\texttt{e}-type~argument,~
622     but~cannot~be~fully~expanded~within~an~\texttt{f}-type~argument.
623 }
624 \NewDocumentCommand { \CodedocExplainTF } { }
625 {
626     \raisebox{\baselineskip}[0pt][0pt]{\hypertarget{explTF}{}}%
627     \write \@auxout { \def \string \Codedoc@explTF { } }

```

```

628   \_codeloc_typeset_TF:\ indicates~conditional~(\texttt{if})~functions~
629   whose~variants~with~\texttt{T},~\texttt{F}~and~\texttt{TF}~
630   argument~specifiers~expect~different~
631   \enquote{true}/\enquote{false}~branches.
632 }

```

(End of definition for `\CodedocExplain` and others. These functions are documented on page ??.)

5.9 Implementing text markup

Keys for `\cmd`, `\cs` and `\tn`.

```

633 \keys_define:nn { l3doc/cmd }
634 {
635   index      .tl_set:N      = \l__codeloc_cmd_index_tl      ,
636   module     .tl_set:N      = \l__codeloc_cmd_module_tl     ,
637   no-index   .bool_set:N    = \l__codeloc_cmd_noindex_bool   ,
638   replace    .bool_set:N    = \l__codeloc_cmd_replace_bool   ,
639 }

```

`_codeloc_cmd:nn` Apply the key–value *<options>* #1 after setting some default values. Then (unless `_codeloc_cmd:no` `replace=false`) replace @@ in #2, which is a bit tricky: the _ must be given the catcode expected by `_codeloc_replace_at_at:N`, but should be reverted to their original catcode (normally active, needed for line-breaking) without rescanning the whole argument. Then typeset the command in `\verbatim@font`, after turning it to harmless characters if needed (and keeping the underscore breakable); in any case, spaces must be turned into `\@xobeysp` and we must use \@ to avoid longer spaces after a control sequence that ends for instance with a colon (empty signature). Finally, produce an index entry. Indexing is suppressed when `\l__codeloc_cmd_noindex_bool` is true.

```

640 \cs_new_protected:Npn \_codeloc_cmd:nn #1#2
641 {
642   \bool_set_false:N \l__codeloc_cmd_noindex_bool
643   \bool_set_true:N \l__codeloc_cmd_replace_bool
644   \tl_set:Nn \l__codeloc_cmd_index_tl { \q_no_value }
645   \tl_set:Nn \l__codeloc_cmd_module_tl { \q_no_value }
646   \keys_set:nn { l3doc/cmd } {#1}
647   \tl_set:Nn \l__codeloc_cmd_tl {#2}
648   \bool_if:NT \l__codeloc_cmd_replace_bool
649   {
650     \tl_set_rescan:Nnn \l__codeloc_tmpb_tl { } { _ }
651     \tl_replace_all:NVn \l__codeloc_cmd_tl \l__codeloc_tmpb_tl { _ }

```

```

652     \l__codedoc_replace_at_at:N \l__codedoc_cmd_tl
653     \tl_replace_all:NnV \l__codedoc_cmd_tl { _ } \l__codedoc_tmpb_tl
654 }

```

Typesetting. Note the replacement for the underscore is to permit linebreaks. The `underscore` package adds the linebreak, and the regex results in applying the breakable underscore only to the *last* of a run of underscores, and not if the underscore follows a backslash.

```

655     \mode_if_math:T { \mbox }
656     {
657         \bool_if:NT \l__codedoc_allow_indexing_bool { \l__codedoc_target: }
658         \verbatim@font
659         \l__codedoc_if_almost_str:VT \l__codedoc_cmd_tl
660         {
661             \l__kernel_tl_set:Ne \l__codedoc_cmd_tl { \tl_to_str:N \l__codedoc_cmd_tl }
662             \bool_if:NT \g__codedoc_cs_break_bool
663             {
664                 \regex_replace_all:nnN
665                 { ([^\\\_]\_*) \_ ([^\_]) }
666                 { \1 \c{BreakableUnderscore} \2 }
667                 \l__codedoc_cmd_tl
668             }
669         }
670         \tl_replace_all:Nnn \l__codedoc_cmd_tl { ~ } { \@xobeysp }
671         \l__codedoc_cmd_tl
672         \@
673     }

```

Indexing.

```

674     \bool_if:NT \l__codedoc_allow_indexing_bool
675     {
676         \bool_if:NF \l__codedoc_cmd_noindex_bool
677         {
678             \quark_if_no_value:NF \l__codedoc_cmd_index_tl
679             {
680                 \l__kernel_tl_set:Ne \l__codedoc_cmd_tl
681                 { \c_backslash_str \exp_not:o { \l__codedoc_cmd_index_tl } }
682             }
683             \exp_args:No \l__codedoc_key_get:n { \l__codedoc_cmd_tl }
684             \quark_if_no_value:NF \l__codedoc_cmd_module_tl
685             {
686                 \l__kernel_tl_set:Ne \l__codedoc_index_module_tl
687                 { \tl_to_str:N \l__codedoc_cmd_module_tl }

```

```

688     }
689     \__codedoc_special_index_module:ooonN
690     { \l__codedoc_index_key_tl }
691     { \l__codedoc_index_macro_tl }
692     { \l__codedoc_index_module_tl }
693     { usage }
694     \l__codedoc_index_internal_bool
695   }
696 }
697 }
698 \cs_generate_variant:Nn \__codedoc_cmd:nn { no }

```

(End of definition for `__codedoc_cmd:nn`.)

`__codedoc_meta:n` Store #1 in `\l__codedoc_tmpa_tl` and replaces every underscore, regardless of its category (“math toggle”, “alignment”, “superscript”, “subscript”, “letter”, “other”, “active”) by `__codedoc_ensuremath_sb:n` (which creates math subscripts), then runs the code used for `\meta` in `doc.sty`.

```

699 \cs_new_protected:Npn \__codedoc_meta:n #1
700 {
701   \tl_set:Nn \l__codedoc_tmpa_tl {#1}
702   \tl_map_inline:nn
703     { { 3 } { 4 } { 7 } { 8 } { 11 } { 12 } { 13 } }
704     {
705       \tl_set_rescan:Nnn \l__codedoc_tmpb_tl
706         { \char_set_catcode:nn { ` } {##1} } { _ }
707       \tl_replace_all:NVn \l__codedoc_tmpa_tl \l__codedoc_tmpb_tl
708         { \__codedoc_ensuremath_sb:n }
709     }
710   \exp_args:NV \__codedoc_meta_original:n \l__codedoc_tmpa_tl
711 }
712 \cs_new_protected:Npn \__codedoc_ensuremath_sb:n #1
713 { \ensuremath { \sb {#1} } }
714 \cs_new_protected:Npn \__codedoc_meta_original:n #1
715 {
716   \ensuremath \langle
717   \mode_if_math:T { \nfss@text }
718   {
719     \meta@font@select
720     \edef \meta@hyphen@restore
721       { \hyphenchar \the \font \the \hyphenchar \font }
722     \hyphenchar \font \m@ne

```

```

723     \language \l@nohyphenation
724     #1 \/  

725     \meta@hyphen@restore  

726   }  

727   \ensuremath \rangle  

728 }

```

(End of definition for `__codedoc_meta:n`, `__codedoc_ensuremath_sb:n`, and `__codedoc_meta_original:n`.)

5.9.1 Common between macro and function

`__codedoc_typeset_exp:` Used by `__codedoc_macro_single:nNN` and in the function environment to typeset conditionals and auxiliary functions.

```

\__codedoc_typeset_TF: 729 \cs_new_protected:Npn \__codedoc_typeset_exp:
\__codedoc_typeset_aux:n 730 {
731   \cs_if_exist:NTF \Codedoc@expstar
732     { \hyperlink { expstar } }
733     { \mbox {
734       { $\star$ }
735     } }
736 \cs_new_protected:Npn \__codedoc_typeset_rexp:
737 {
738   \cs_if_exist:NTF \Codedoc@rexpstar
739     { \hyperlink { rexpstar } }
740     { \mbox {
741       { \ding { 73 } } } % hollow star
742     } }
743 \cs_new_protected:Npn \__codedoc_typeset_TF:
744 {
745   \cs_if_exist:NTF \Codedoc@explTF
746     { \hyperlink { explTF } }
747     { \mbox {
748       {
749         \color{black}
750         \itshape TF
751         \makebox[0pt][r]
752         {
753           \cs_if_exist:NT \Codedoc@explTF { \color{red} }
754           \underline { \phantom{\itshape TF} \kern-0.1em }
755         }
756       }
757     } }
758 \cs_new_protected:Npn \__codedoc_typeset_aux:n #1

```

```

759 {
760   { \color[gray]{0.5} #1 }
761 }

```

(End of definition for `__codedoc_typeset_exp`: and others.)

`__codedoc_get_hyper_target:nN` Create a `hyperref` anchor from a macro name `#1` and stores it in the token list variable `#2`. For instance, `\prg_replicate:nn` gives `doc/function//prg/replicate:nn`.

```

\__codedoc_get_hyper_target:oN
\__codedoc_get_hyper_target:eN
762 \cs_new_protected:Npn \__codedoc_get_hyper_target:nN #1#2
763 {
764   \__kernel_tl_set:Ne #2 { \tl_to_str:n {#1} }
765   \tl_replace_all:Nvn #2 \c_underscore_str { / }
766   \tl_remove_all:Nv #2 \c_backslash_str
767   \tl_put_left:Nn #2 { doc/function// }
768 }
769 \cs_generate_variant:Nn \__codedoc_get_hyper_target:nN { o , e }

```

(End of definition for `__codedoc_get_hyper_target:nN`.)

`__codedoc_names_get_seq:nN` The argument `#1` (argument of a function or macro environment) has catcodes 10 (space), 12 (other) and 13 (active). Sanitize catcodes. If the `verb` option was used, output a one-item sequence. Otherwise, remove any “%” character at the beginning of a line. Remove tabs and newlines. Finally, convert `__` and `__` to `__`(*module name*) (if it is non-empty). At this point, `\l__codedoc_tmpa_tl` contains a comma-delimited list of names, where `@` and `_` have category code letter. Turn it to a string, parse it as a comma-delimited list (in particular this removes spaces), and output a sequence of function/macro names.

```

770 \cs_new_protected:Npn \__codedoc_names_get_seq:nN #1#2
771 {
772   \__kernel_tl_set:Ne \l__codedoc_tmpa_tl { \tl_to_str:n {#1} }
773   \bool_if:NTF \l__codedoc_names_verb_bool
774   {
775     \seq_clear:N #2
776     \seq_put_right:Nv #2 \l__codedoc_tmpa_tl
777   }
778   {
779     \tl_remove_all:Ne \l__codedoc_tmpa_tl
780     { \iow_char:N \^^M \c_percent_str }
781     \tl_remove_all:Ne \l__codedoc_tmpa_tl { \tl_to_str:n { ^ ^ A } }
782     \tl_remove_all:Ne \l__codedoc_tmpa_tl { \iow_char:N \^^I }
783     \tl_remove_all:Ne \l__codedoc_tmpa_tl { \iow_char:N \^^M }
784     \__codedoc_detect_internals:N \l__codedoc_tmpa_tl

```



```

785     \__codedoc_replace_at_at:N \l__codedoc_tmpa_tl
786     \exp_args:NNe \seq_set_from_clist:Nn #2
787     { \tl_to_str:N \l__codedoc_tmpa_tl }
788   }
789 }

```

(End of definition for __codedoc_names_get_seq:nN.)

`__codedoc_names_parse:` The goal is to group variants together. We populate `\l__codedoc_names_block_tl` with local sequence variable named with `__codedoc_lseq_name:n` after the base forms. When encountering a new base form, set the corresponding local sequence to hold the *base name* (stripped of the signature) and add the local sequence to the list `\l__codedoc_names_block_tl`. In all cases append the signature to the local sequence, which thus takes the form *base name*, *signature*₁, *signature*₂ and so on. If the original function had no signature (no colon) then use `\scan_stop:` as the signature (there can be no variant). We special case commands `#1` starting with `\:.`, namely weird functions named `\:.`N and the like.

```

790 \cs_new_protected:Npn \__codedoc_names_parse:
791 {
792   \tl_clear:N \l__codedoc_names_block_tl
793   \seq_map_function:NN
794     \l__codedoc_names_seq
795     \__codedoc_names_parse_one:n
796 }
797 \cs_new_protected:Npn \__codedoc_names_parse_one:n #1
798 {
799   \__codedoc_split_function_do:nn {#1}
800   { \__codedoc_names_parse_one_aux:nnNn }
801   {#1}
802 }
803 \cs_new_protected:Npn \__codedoc_names_parse_one_aux:nnNn #1#2#3#4
804 {
805   \bool_if:NTF #3
806   {
807     \tl_if_head_eq_charcode:nNTF {#2} :
808     { \__codedoc_names_parse_aux:nnn {#4} {#4} { \scan_stop: } }
809     {
810       \exp_args:Ne \__codedoc_names_parse_aux:nnn
811       { \__codedoc_base_form_aux:nnN {#1} {#2} #3 }
812       {#1} {#2}
813     }
814   }
815 }

```

```

814     }
815     {
816         \bool_if:NT \l__codedoc_macro_TF_bool
817         { \msg_error:nne { l3doc } { no-signature-TF } {#4} }
818         \__codedoc_names_parse_aux:nnn {#4} {#4} { \scan_stop: }
819     }
820 }
821 \cs_new_protected:Npn \__codedoc_names_parse_aux:nnn #1
822 { \exp_args:Nc \__codedoc_names_parse_aux:Nnn { \__codedoc_lseq_name:n {#1} } }
823 \cs_new_protected:Npn \__codedoc_names_parse_aux:Nnn #1#2#3
824 {
825     \tl_if_in:NnF \l__codedoc_names_block_tl {#1}
826     {
827         \tl_put_right:Nn \l__codedoc_names_block_tl {#1}
828         \seq_clear_new:N #1
829         \seq_put_right:Nn #1 {#2}
830     }
831     \seq_put_right:Nn #1 {#3}
832 }

```

(End of definition for `__codedoc_names_parse:` and `__codedoc_names_parse_one:n`.)

`__codedoc_names_typeset:` This code is in particular used when typesetting function names in a function environment. The mapping over `\l__codedoc_names_block_tl` cannot use `\tl_map-inline:Nn` because the code following `\` would not be expandable, thus breaking `\bottomrule`.

Call `__codedoc_names_typeset_auxi:n` on each local sequence (which holds a set of variants). The first step is to pop the base form and change spaces to category other so that they get displayed eventually. Then store the variants in `\g__codedoc_variants_seq`, remove the first, which will be displayed more prominently, and reconstruct the actual name, passing it to `__codedoc_names_typeset_auxii:n`.

```

833 \cs_new_protected:Npn \__codedoc_names_typeset:
834 {
835     \tl_map_function:NN \l__codedoc_names_block_tl
836     \__codedoc_names_typeset_auxi:n
837 }
838 \cs_new_protected:Npn \__codedoc_names_typeset_auxi:n #1
839 {
840     \seq_pop:NN #1 \l__codedoc_tmpa_tl
841     \tl_gset_eq:NN \g__codedoc_base_name_tl \l__codedoc_tmpa_tl

```

```

842 \tl_greplace_all:NnV \g__codedoc_base_name_tl
843 { ~ } \c_catcode_other_space_tl
844 \seq_get:NN #1 \l__codedoc_tmpa_tl
845 \str_if_eq:VnTF \l__codedoc_tmpa_tl { \scan_stop: }
846 {
847   \seq_gclear:N \g__codedoc_variants_seq
848   \__codedoc_names_typeset_auxii:e { \g__codedoc_base_name_tl }
849 }
850 {
851   \seq_gset_eq:NN \g__codedoc_variants_seq #1
852   \seq_gpop:NN \g__codedoc_variants_seq \l__codedoc_tmpb_tl
853   \__codedoc_names_typeset_auxii:e
854   { \g__codedoc_base_name_tl : \l__codedoc_tmpb_tl }
855 }
856 }

```

(End of definition for `__codedoc_names_typeset:` and `__codedoc_names_typeset_auxii:n`.)

`__codedoc_names_typeset_auxii:n` In case the option `pTF` was given, typeset predicates before the TF functions. In case
`__codedoc_names_typeset_auxii:e` the option `noTF` was given, typeset the non-TF function as well. Pass the relevant boolean in both cases to control whether to append TF.

```

857 \cs_new_protected:Npn \__codedoc_names_typeset_auxii:n #1
858 {
859   \bool_if:NT \l__codedoc_macro_pTF_bool
860   {
861     \__codedoc_names_typeset_block:eN
862     { \__codedoc_predicate_from_base:n {#1} }
863     \c_false_bool
864   }
865   \bool_if:NT \l__codedoc_macro_noTF_bool
866   { \__codedoc_names_typeset_block:nN {#1} \c_false_bool }
867   \__codedoc_names_typeset_block:nN {#1} \l__codedoc_macro_TF_bool
868 }
869 \cs_generate_variant:Nn \__codedoc_names_typeset_auxii:n { e }

```

(End of definition for `__codedoc_names_typeset_auxii:n`.)

`__codedoc_names_typeset_block:nN` Names in `function` and `macro` environments are typeset differently. To distinguish
`__codedoc_names_typeset_block:eN` the two note that `\l__codedoc_nested_macro_int` is at least one when in an `macro` environment (we assume `function` is not nested inside it). A block is a function with all its variants.

```

870 \cs_new_protected:Npn \__codedoc_names_typeset_block:nN

```

```

871 {
872   \int_compare:nNnTF \l__codedoc_nested_macro_int = 0
873     { \__codedoc_typeset_function_block:nN }
874     { \__codedoc_macro_typeset_block:nN }
875   }
876   \cs_generate_variant:Nn \__codedoc_names_typeset_block:nN { e }

```

(End of definition for `__codedoc_names_typeset_block:nN`.)

`__codedoc_if_macro_internal_p:n` Determines whether the given macro should be considered internal or public. If an option such as `int` was given then the answer is `\l__codedoc_macro_internal_bool`, otherwise check for whether the macro name contains `__`.

```

877 \prg_new_conditional:Npnn \__codedoc_if_macro_internal:n #1 { p , T , F , TF }
878 {
879   \bool_if:NTF \l__codedoc_macro_internal_bool
880     { \prg_return_true: }
881     {
882       \tl_if_empty:eTF
883         {
884           \exp_after:wN \__codedoc_if_macro_internal_aux:w
885           \tl_to_str:n { #1 ~ __ }
886         }
887         { \prg_return_false: } { \prg_return_true: }
888       }
889     }
890   \exp_last_unbraced:NNNNo
891   \cs_new:Npn \__codedoc_if_macro_internal_aux:w #1 { \tl_to_str:n { __ } } { }

```

(End of definition for `__codedoc_if_macro_internal:nTF` and `__codedoc_if_macro_internal_aux:w`.)

`__codedoc_names_block_base_map:N` The `\l__codedoc_names_block_tl` contains sequence variables corresponding to different base functions and their variants. For each such sequence, put the first and second items in `\l__codedoc_tmpa_tl` and `\l__codedoc_tmpb_tl` and build the base function's name.

```

892 \cs_new_protected:Npn \__codedoc_names_block_base_map:N #1
893 {
894   \tl_map_inline:Nn \l__codedoc_names_block_tl
895     {
896       \group_begin:
897       \seq_set_eq:NN \l__codedoc_tmpa_seq ##1
898       \seq_pop:NN \l__codedoc_tmpa_seq \l__codedoc_tmpa_tl
899       \seq_get:NN \l__codedoc_tmpa_seq \l__codedoc_tmpb_tl

```

```

900         \exp_args:NNe
901     \group_end:
902     #1
903     {
904         \l__codedoc_tmpa_tl
905         \str_if_eq:VnF \l__codedoc_tmpb_tl { \scan_stop: }
906         { : \l__codedoc_tmpb_tl }
907         \bool_if:NT \l__codedoc_macro_TF_bool { TF }
908     }
909 }
910 }

```

(End of definition for `__codedoc_names_block_base_map:N`.)

5.9.2 The function environment

```

911 \keys_define:nn { l3doc/function }
912 {
913     TF .value_forbidden:n = true ,
914     TF .code:n =
915     {
916         \bool_set_true:N \l__codedoc_macro_TF_bool
917     } ,
918     EXP .value_forbidden:n = true ,
919     EXP .code:n =
920     {
921         \bool_set_true:N \l__codedoc_macro_EXP_bool
922         \bool_set_false:N \l__codedoc_macro_rEXP_bool
923     } ,
924     rEXP .value_forbidden:n = true ,
925     rEXP .code:n =
926     {
927         \bool_set_false:N \l__codedoc_macro_EXP_bool
928         \bool_set_true:N \l__codedoc_macro_rEXP_bool
929     } ,
930     pTF .value_forbidden:n = true ,
931     pTF .code:n =
932     {
933         \bool_set_true:N \l__codedoc_macro_pTF_bool
934         \bool_set_true:N \l__codedoc_macro_TF_bool
935         \bool_set_true:N \l__codedoc_macro_EXP_bool
936         \bool_set_false:N \l__codedoc_macro_rEXP_bool
937     } ,

```

```

938     noTF .value_forbidden:n = true ,
939     noTF .code:n =
940     {
941         \bool_set_true:N \l__codedoc_macro_noTF_bool
942         \bool_set_true:N \l__codedoc_macro_TF_bool
943     } ,
944     added .code:n = { \__codedoc_date_set_past:Nn \l__codedoc_date_added_tl {#1} },
945     updated .code:n = { \__codedoc_date_set_past:Nn \l__codedoc_date_updated_tl {#1} } ,
946     deprecated .bool_set:N = \l__codedoc_macro_deprecated_bool ,
947     no-user-doc .bool_set:N = \l__codedoc_macro_nodoc_bool ,
948     tested .code:n = { } ,
949     label .code:n =
950     {
951         \clist_set:Nn \l__codedoc_function_label_clist {#1}
952         \bool_set_true:N \l__codedoc_no_label_bool
953     } ,
954     verb .value_forbidden:n = true ,
955     verb .bool_set:N = \l__codedoc_names_verb_bool ,
956     module .tl_set:N = \l__codedoc_override_module_tl ,
957 }

```

`__codedoc_date_set:Nn` Normalize the date into the format YYYY-MM-DD; more precisely month and day are allowed to be single digits. The `__codedoc_date_set_past:Nn` function only allows dates in the past (or same day).

```

958 \cs_new_protected:Npn \__codedoc_date_set:Nn #1#2
959 {
960     \tl_set:Nn #1 {#2}
961     \regex_replace_once:nnNF
962     { \A(\d\d\d\d)[-/](\d\d?)[-/](\d\d?)\Z } { \1-\2-\3 } #1
963     {
964         \msg_error:nnn { l3doc } { date-format } {#2}
965         \tl_set:Nn #1 { 1970-01-01 }
966     }
967 }
968 \cs_new_protected:Npn \__codedoc_date_set_past:Nn #1#2
969 {
970     \__codedoc_date_set:Nn #1 {#2}
971     \exp_args:No \__codedoc_date_compare:nNnT
972     {#1} > { \c_sys_year_int - \c_sys_month_int - \c_sys_day_int }
973     {
974         \msg_error:nnee { l3doc } { future-date }
975         { \tl_to_str:N \l__codedoc_macro_argument_tl }

```

```

976         {#1}
977     }
978 }

```

(End of definition for `__codedoc_date_set:Nn` and `__codedoc_date_set_past:Nn`.)

`__codedoc_function:nnw` **#1** : Key–value list.
#2 : Comma-separated list of functions; input has already been sanitised by catcode changes before reading the argument.

`__codedoc_function_end:` Make sure any paragraph is finished, and similar safe practices at the beginning of an environment which will typeset material. Initialize some variables. Parse the key–value list. Clean up the list of functions, then go through them to extract some data. After this, typeset the function names in the coffin `\l__codedoc_functions_coffin` and measure it to know if it fits in the margin. Finally, start a vertical coffin for the main part of the environment. This coffin stops when the environment ends, then all the pieces are assembled into a single coffin, which is typeset.

```

979 \cs_new_protected:Npn \__codedoc_function:nnw #1#2
980 {
981     \__codedoc_function_typeset_start:
982     \__codedoc_function_init:
983     \tl_set:Nn \l__codedoc_macro_argument_tl {#2}
984     \keys_set:nn { l3doc/function } {#1}
985     \__codedoc_names_get_seq:nN {#2} \l__codedoc_names_seq
986     \__codedoc_names_parse:
987     \__codedoc_function_typeset:
988     \__codedoc_function_reset:
989     \__codedoc_function_descr_start:w
990 }
991 \cs_new_protected:Npn \__codedoc_function_end:
992 {
993     \__codedoc_function_descr_stop:
994     \__codedoc_function_assemble:
995     \__codedoc_function_typeset_stop:
996 }

```

(End of definition for `__codedoc_function:nnw` and `__codedoc_function_end:.`)

`__codedoc_function_typeset_start:` At the start of the `function` environment, before performing any assignment, close the last paragraph, and set up the typesetting scene. Further code typesets a coffin, so we end the paragraph and allow a page break.

```

997 \cs_new_protected:Npn \__codedoc_function_typeset_start:

```

```

998 {
999   \par \bigskip \noindent
1000 }
1001 \cs_new_protected:Npn \__codedoc_function_typeset_stop:
1002 {
1003   \par
1004   \dim_set:Nn \prevdepth { \box_dp:N \l__codedoc_descr_coffin }
1005   \allowbreak
1006 }

```

(End of definition for __codedoc_function_typeset_start: and __codedoc_function_typeset_stop:.)

__codedoc_function_init: Complain if function environments are nested. Clear various variables.

```

1007 \cs_new_protected:Npn \__codedoc_function_init:
1008 {
1009   \box_if_empty:NF \g__codedoc_syntax_box
1010   { \msg_error:nn { l3doc } { syntax-nested-function } }
1011   \coffin_clear:N \l__codedoc_descr_coffin
1012   \box_gclear:N \g__codedoc_syntax_box
1013   \coffin_clear:N \l__codedoc_syntax_coffin
1014   \coffin_clear:N \l__codedoc_functions_coffin
1015   \bool_set_false:N \l__codedoc_macro_TF_bool
1016   \bool_set_false:N \l__codedoc_macro_pTF_bool
1017   \bool_set_false:N \l__codedoc_macro_noTF_bool
1018   \bool_set_false:N \l__codedoc_macro_EXP_bool
1019   \bool_set_false:N \l__codedoc_macro_rEXP_bool
1020   \bool_set_false:N \l__codedoc_no_label_bool
1021   \bool_set_false:N \l__codedoc_names_verb_bool
1022   \bool_set_true:N \l__codedoc_in_function_bool
1023   \clist_clear:N \l__codedoc_function_label_clist
1024   \tl_set:Nn \l__codedoc_override_module_tl { \q_no_value }
1025   \char_set_active_eq:NN \< \__codedoc_shorthand_meta:
1026   \char_set_catcode_active:N \<
1027 }

```

(End of definition for __codedoc_function_init:.)

__codedoc_shorthand_meta: Allow <...> to be used as markup for \meta{...}.

```

\__codedoc_shorthand_meta:w 1028 \cs_new_protected:Npn \__codedoc_shorthand_meta:
1029 { \mode_if_math:TF { < } { \__codedoc_shorthand_meta:w } }
1030 \cs_new_protected_nopar:Npn \__codedoc_shorthand_meta:w #1 > { \meta {#1} }

```

(End of definition for __codedoc_shorthand_meta: and __codedoc_shorthand_meta:w.)

`__codedoc_function_reset:` Clear some variables.

```
1031 \cs_new_protected:Npn \__codedoc_function_reset:
1032 {
1033     \tl_set:Nn \l__codedoc_override_module_tl { \q_no_value }
1034 }
```

(End of definition for __codedoc_function_reset:.)

`__codedoc_function_typeset:` Typeset in the coffin `\l__codedoc_functions_coffin` the functions listed in `\l__codedoc_names_block_tl` and the relevant dates, then set `\l__codedoc_long_name_bool` to be true if this coffin is larger than the available width in the margin. The function `__codedoc_typeset_functions:` is quite involved hence given later.

```
1035 \cs_new_protected:Npn \__codedoc_function_typeset:
1036 {
1037     \dim_zero:N \l__codedoc_trial_width_dim
1038     \hcoffin_set:Nn \l__codedoc_functions_coffin { \__codedoc_typeset_functions: }
1039     \dim_set:Nn \l__codedoc_trial_width_dim
1040     { \box_wd:N \l__codedoc_functions_coffin }
1041     \bool_set:Nn \l__codedoc_long_name_bool
1042     { \dim_compare_p:nNn \l__codedoc_trial_width_dim > \marginparwidth }
1043 }
```

(End of definition for __codedoc_function_typeset:.)

`__codedoc_function_descr_start:w` The last step in `__codedoc_function:nnw` (the beginning of a function environment) is to open a coffin which will capture the description of the function, namely the body of the function environment. This is closed by `__codedoc_function_end:` (the end of a function environment).

`__codedoc_function_descr_stop:`

```
1044 \cs_new_protected:Npn \__codedoc_function_descr_start:w
1045 {
1046     \vcoffin_set:Nnw \l__codedoc_descr_coffin { \textwidth }
1047     \noindent \ignorespaces
1048 }
1049 \cs_new_protected:Npn \__codedoc_function_descr_stop:
1050 { \vcoffin_set_end: }
```

(End of definition for __codedoc_function_descr_start:w and __codedoc_function_descr_stop:.)

`__codedoc_function_assemble:` The box `\g__codedoc_syntax_box` contains the contents of the syntax environment if it was used. Now that we have all the pieces, join together the syntax coffin, the names coffin, and the description coffin. The relative positions depend on whether the names coffin fits in the margin. Then typeset the combination.

```

1051 \cs_new_protected:Npn \__codedoc_function_assemble:
1052 {
1053   \hcoffin_set:Nn \l__codedoc_syntax_coffin
1054     { \box_use_drop:N \g__codedoc_syntax_box }
1055   \bool_if:NTF \l__codedoc_long_name_bool
1056     {
1057       \coffin_join:NnnNnnnn
1058         \l__codedoc_output_coffin {hc} {vc}
1059         \l__codedoc_syntax_coffin {l} {T}
1060         {Opt} {Opt}
1061       \coffin_join:NnnNnnnn
1062         \l__codedoc_output_coffin {l} {t}
1063         \l__codedoc_functions_coffin {r} {t}
1064         {-\marginparsep} {Opt}
1065       \coffin_join:NnnNnnnn
1066         \l__codedoc_output_coffin {l} {b}
1067         \l__codedoc_descr_coffin {l} {t}
1068         {0.75\marginparwidth + \marginparsep} {-\medskipamount}
1069       \coffin_typeset:Nnnnn \l__codedoc_output_coffin
1070         {\l__codedoc_descr_coffin-l} {\l__codedoc_descr_coffin-t}
1071         {Opt} {Opt}
1072     }
1073   {
1074     \coffin_join:NnnNnnnn
1075       \l__codedoc_output_coffin {hc} {vc}
1076       \l__codedoc_syntax_coffin {l} {t}
1077       {Opt} {Opt}
1078     \coffin_join:NnnNnnnn
1079       \l__codedoc_output_coffin {l} {b}
1080       \l__codedoc_descr_coffin {l} {t}
1081       {Opt} {-\medskipamount}
1082     \coffin_join:NnnNnnnn
1083       \l__codedoc_output_coffin {l} {t}
1084       \l__codedoc_functions_coffin {r} {t}
1085       {-\marginparsep} {Opt}
1086     \coffin_typeset:Nnnnn \l__codedoc_output_coffin
1087       {\l__codedoc_syntax_coffin-l} {\l__codedoc_syntax_coffin-T}
1088       {Opt} {Opt}
1089   }
1090 }

```

(End of definition for __codedoc_function_assemble:.)

`__codedoc_typeset_functions:` This function builds the `\l__codedoc_functions_coffin` by typesetting the function names (with variants) and the relevant dates in a `tabular` environment. The use of rules `\toprule`, `\midrule` and `\bottomrule` requires whatever lies between the last `\\` and the rule to be expandable, making our lives a bit complicated.

```

1091 \cs_new_protected:Npn \__codedoc_typeset_functions:
1092 {
1093   \small\ttfamily
1094   \__codedoc_target:
1095   \Hy@MakeCurrentHref { HD. \int_use:N \c@HD@hypercount }
1096   \begin{tabular} [t] { @{} l @{} >{\hspace{\tabcolsep}} r @{} }
1097     \toprule
1098     \__codedoc_function_extra_labels:
1099     \__codedoc_names_typeset:
1100     \__codedoc_typeset_dates:
1101     \bottomrule
1102   \end{tabular}
1103   \normalfont\normalsize
1104 }

```

(End of definition for `__codedoc_typeset_functions:.`)

`__codedoc_typeset_function_block:nN` #1 is a csname, #2 a boolean indicating whether to add TF or not.

`__codedoc_typeset_function_block:eN` 1105 `\cs_new_protected:Npn __codedoc_typeset_function_block:nN #1#2`

```

1106 {
1107   \__codedoc_function_index:e
1108   { #1 \bool_if:NT #2 { \tl_to_str:n {TF} } }
1109   \__codedoc_function_label:eN {#1} #2
1110   #1
1111   \bool_if:NT #2 { \__codedoc_typeset_TF: }
1112   \__codedoc_typeset_expandability:
1113   \seq_if_empty:NF \g__codedoc_variants_seq
1114   { \__codedoc_typeset_variant_list:nN {#1} #2 }
1115   \\
1116 }
1117 \cs_generate_variant:Nn \__codedoc_typeset_function_block:nN { e }
1118 \cs_new_protected:Npn \__codedoc_function_index:n #1
1119 {
1120   \seq_gput_right:Nn \g_doc_functions_seq {#1}
1121   \__codedoc_special_index:nn {#1} { usage }
1122 }
1123 \cs_generate_variant:Nn \__codedoc_function_index:n { e }

```

```

1124 \cs_new_protected:Npn \__codedoc_typeset_expandability:
1125 {
1126   &
1127   \bool_if:NT \l__codedoc_macro_EXP_bool { \__codedoc_typeset_exp: }
1128   \bool_if:NT \l__codedoc_macro_rEXP_bool { \__codedoc_typeset_rexp: }
1129 }

```

#1 is the function, #2 whether to add TF.

```

1130 \cs_new_protected:Npn \__codedoc_typeset_variant_list:nN #1#2
1131 {
1132   \\\
1133   \__codedoc_typeset_aux:n { \__codedoc_get_function_name:n {#1} }
1134   :
1135   \int_compare:nTF { \seq_count:N \g__codedoc_variants_seq == 1 }
1136   { \seq_use:Nn \g__codedoc_variants_seq { } }
1137   {
1138     \hbox_set:Nn \l_tmpa_box
1139     { \seq_use:Nn \g__codedoc_variants_seq { \textrm| \nolinebreak[2] } }
1140     \textrm(

```

Set long variant lists in a parbox, short lists set natural length.

```

1141     \dim_compare:nNnTF { \box_wd:N \l_tmpa_box } > { .4\columnwidth }
1142     {
1143       \parbox[t]{.4\columnwidth}
1144       {
1145         \raggedright
1146         \hbox_unpack_drop:N \l_tmpa_box
1147         \textrm)
1148         \bool_if:NT #2 { \__codedoc_typeset_TF: }
1149       }
1150     }
1151     {
1152       \hbox_unpack_drop:N \l_tmpa_box
1153       \textrm)
1154       \bool_if:NT #2 { \__codedoc_typeset_TF: }
1155     }
1156   }
1157   \__codedoc_typeset_expandability:
1158 }

```

#1 is the function name, #2 whether to add TF.

```

1159 \cs_new_protected:Npn \__codedoc_function_extra_labels:
1160 {
1161   \bool_if:NT \l__codedoc_no_label_bool

```

```

1162     {
1163       \clist_map_inline:Nn \l__codedoc_function_label_clist
1164       {
1165         \__codedoc_get_hyper_target:oN { \token_to_str:N ##1 }
1166         \l__codedoc_tmpa_tl
1167         \exp_args:No \label { \l__codedoc_tmpa_tl }
1168       }
1169     }
1170   }
1171   \cs_new_protected:Npn \__codedoc_function_label:nN #1#2
1172   {
1173     \bool_if:NF \l__codedoc_no_label_bool
1174     {
1175       \__codedoc_get_hyper_target:eN
1176       {
1177         \exp_not:n {#1}
1178         \bool_if:NT #2 { \tl_to_str:n {TF} }
1179       }
1180       \l__codedoc_tmpa_tl
1181       \exp_args:No \label { \l__codedoc_tmpa_tl }
1182     }
1183   }
1184   \cs_generate_variant:Nn \__codedoc_function_label:nN { e }

```

(End of definition for `__codedoc_typeset_function_block:nN` and `__codedoc_function_index:n`.)

`__codedoc_typeset_dates:` To display metadata for when functions are added/modified. This function must be expandable since it produces rules for use in alignments.

```

1185   \cs_new:Npn \__codedoc_typeset_dates:
1186   {
1187     \bool_lazy_and:nnF
1188     { \tl_if_empty_p:N \l__codedoc_date_added_tl }
1189     { \tl_if_empty_p:N \l__codedoc_date_updated_tl }
1190     { \midrule }
1191     \tl_if_empty:NF \l__codedoc_date_added_tl
1192     {
1193       \multicolumn { 2 } { @{} r @{} }
1194       { \scriptsize New: \, \l__codedoc_date_added_tl } \\
1195     }
1196
1197     \tl_if_empty:NF \l__codedoc_date_updated_tl
1198     {
1199       \multicolumn { 2 } { @{} r @{} }

```

```

1200         { \scriptsize Updated: \, \l__codedoc_date_updated_tl } \\
1201     }
1202 }

```

(End of definition for `__codedoc_typeset_dates:`.)

`__codedoc_syntax:w` Implement the `syntax` environment.

```

\__codedoc_syntax_end: 1203 \dim_new:N \l__codedoc_syntax_dim
1204 \cs_new_protected:Npn \__codedoc_syntax:w
1205 {
1206     \box_if_empty:NF \g__codedoc_syntax_box
1207     { \msg_error:nn { l3doc } { multiple-syntax } }
1208     \dim_set:Nn \l__codedoc_syntax_dim
1209     {
1210         \textwidth
1211         \bool_if:NT \l__codedoc_long_name_bool
1212         { + 0.75 \marginparwidth - \l__codedoc_trial_width_dim }
1213     }
1214     \hbox_gset:Nw \g__codedoc_syntax_box
1215     \small \ttfamily
1216     \arrayrulecolor{white}
1217     \begin{tabular} { @{} l @{} }
1218         \toprule
1219         \begin{minipage}[t]{\l__codedoc_syntax_dim}
1220             \raggedright
1221             \obeyspaces
1222             \obeylines
1223         }
1224     \cs_new_protected:Npn \__codedoc_syntax_end:
1225     {
1226         \end{minipage}
1227         \end{tabular}
1228         \arrayrulecolor{black}
1229     \hbox_gset_end:
1230     \bool_if:NF \l__codedoc_in_function_bool
1231     {
1232         \begin{quote}
1233             \mode_leave_vertical:
1234             \box_use_drop:N \g__codedoc_syntax_box
1235         \end{quote}
1236     }
1237 }

```

(End of definition for `__codedoc_syntax:w` and `__codedoc_syntax_end:`.)

5.9.3 The macro environment

Keyval for the macro environment. TODO: provide document command for documenting keys.

```
1238 \keys_define:nn { l3doc/macro }
1239 {
1240   aux .value_forbidden:n = true ,
1241   aux .code:n =
1242     {
1243       \msg_warning:nnnn { l3doc } { deprecated-option }
1244       { aux } { function/macro }
1245     } ,
1246   deprecated .bool_set:N = \l__codedoc_macro_deprecated_bool ,
1247   internal .value_forbidden:n = true ,
1248   internal .code:n =
1249     { \bool_set_true:N \l__codedoc_macro_internal_bool } ,
1250   int .value_forbidden:n = true ,
1251   int .code:n =
1252     { \bool_set_true:N \l__codedoc_macro_internal_bool } ,
1253   no-user-doc .bool_set:N = \l__codedoc_macro_nodoc_bool ,
1254   var .value_forbidden:n = true ,
1255   var .code:n =
1256     { \bool_set_true:N \l__codedoc_macro_var_bool } ,
1257   TF .value_forbidden:n = true ,
1258   TF .code:n =
1259     { \bool_set_true:N \l__codedoc_macro_TF_bool } ,
1260   pTF .value_forbidden:n = true ,
1261   pTF .code:n =
1262     {
1263       \bool_set_true:N \l__codedoc_macro_TF_bool
1264       \bool_set_true:N \l__codedoc_macro_pTF_bool
1265       \bool_set_true:N \l__codedoc_macro_EXP_bool
1266       \bool_set_false:N \l__codedoc_macro_rEXP_bool
1267     } ,
1268   noTF .value_forbidden:n = true ,
1269   noTF .code:n =
1270     {
1271       \bool_set_true:N \l__codedoc_macro_TF_bool
1272       \bool_set_true:N \l__codedoc_macro_noTF_bool
1273     } ,
1274   EXP .value_forbidden:n = true ,
1275   EXP .code:n =
```

```

1276     {
1277         \bool_set_true:N \l__codedoc_macro_EXP_bool
1278         \bool_set_false:N \l__codedoc_macro_rEXP_bool
1279     } ,
1280     rEXP .value_forbidden:n = true ,
1281     rEXP .code:n =
1282     {
1283         \bool_set_false:N \l__codedoc_macro_EXP_bool
1284         \bool_set_true:N \l__codedoc_macro_rEXP_bool
1285     } ,
1286     tested .code:n =
1287     {
1288         \bool_set_true:N \l__codedoc_macro_tested_bool
1289     } ,
1290     added .code:n = {} , % TODO
1291     updated .code:n = {} , % TODO
1292     verb .bool_set:N = \l__codedoc_names_verb_bool ,
1293     module .tl_set:N = \l__codedoc_override_module_tl ,
1294     documented-as .tl_set:N = \l__codedoc_macro_documented_tl ,
1295     do-not-index .value_required:n = true ,
1296     do-not-index .tl_set:N = \l__codedoc_macro_do_not_index_tl ,
1297     % do-not-index .default:n = \q_no_value ,
1298 }

```

`__codedoc_macro:nnw` The arguments are a key–value list of $\langle options \rangle$ and a comma-list of $\langle names \rangle$, read verbatim by `ltxcmd`. First initialize some variables before applying the $\langle options \rangle$, then parse the $\langle names \rangle$ to get a sequence of macro names, then apply `__codedoc_macro_single:nNN` to each (this step is more subtle than `\seq_map_function:NN` because of TF/pTF/noTF). Finally typeset the macro names in the margin.

```

1299 \cs_new_protected:Npn \__codedoc_macro:nnw #1#2
1300 {
1301     \__codedoc_macro_init:
1302     \tl_set:Nn \l__codedoc_macro_argument_tl {#2}
1303     \keys_set:nn { l3doc/macro } {#1}
1304     \__codedoc_names_get_seq:nN {#2} \l__codedoc_names_seq
1305     \__codedoc_names_parse:
1306     \__codedoc_macro_exclude_index:
1307     \__codedoc_macro_save_names:
1308     \__codedoc_names_typeset:
1309     \__codedoc_macro_dump:
1310     \__codedoc_macro_reset:
1311 }

```


(End of definition for `__codedoc_macro:nmw.`)

`__codedoc_macro_init:` The booleans hold various key-value options, `\l__codedoc_nested_macro_int` counts the number of macro environments around the current point (is 0 outside).

```
1312 \cs_new_protected:Npn \__codedoc_macro_init:
1313 {
1314   \int_incr:N \l__codedoc_nested_macro_int
1315   \bool_set_false:N \l__codedoc_macro_deprecated_bool
1316   \bool_set_false:N \l__codedoc_macro_internal_bool
1317   \bool_set_false:N \l__codedoc_macro_TF_bool
1318   \bool_set_false:N \l__codedoc_macro_pTF_bool
1319   \bool_set_false:N \l__codedoc_macro_noTF_bool
1320   \bool_set_false:N \l__codedoc_macro_EXP_bool
1321   \bool_set_false:N \l__codedoc_macro_rEXP_bool
1322   \bool_set_false:N \l__codedoc_macro_var_bool
1323   \bool_set_false:N \l__codedoc_macro_tested_bool
1324   \bool_set_false:N \l__codedoc_names_verb_bool
1325   \tl_set:Nn \l__codedoc_override_module_tl { \q_no_value }
1326   \tl_clear:N \l__codedoc_macro_documented_tl
1327   \cs_set_eq:NN \testfile \__codedoc_print_testfile:n
1328   \box_clear:N \l__codedoc_macro_index_box
1329   \vbox_set:Nn \l__codedoc_macro_box
1330   {
1331     \hbox:n
1332     {
1333       \strut
1334       \int_compare:nNnT \l__codedoc_macro_int = 0 { \__codedoc_target: }
1335     }
1336     \vskip \int_eval:n { \l__codedoc_macro_int - 1 } \baselineskip
1337   }
1338 }
```

(End of definition for `__codedoc_macro_init:.`)

`__codedoc_macro_reset:` We ensure that `\cs` commands nested inside a macro whose module is imposed are not affected.

```
1339 \cs_new_protected:Npn \__codedoc_macro_reset:
1340 {
1341   \tl_set:Nn \l__codedoc_override_module_tl { \q_no_value }
1342 }
```

(End of definition for `__codedoc_macro_reset:.`)

`__codedoc_macro_save_names:` The list of names defined in a set of `macro` environments is eventually used to display on which page they are documented. If the `documented-as` key is given, use that, otherwise find names in `\l__codedoc_names_block_tl`.

```

1343 \cs_new_protected:Npn \__codedoc_macro_save_names:
1344 {
1345   \tl_if_empty:NTF \l__codedoc_macro_documented_tl
1346   { \__codedoc_names_block_base_map:N \__codedoc_macro_save_names_aux:n }
1347   {
1348     \seq_gput_right:Ne \g__codedoc_nested_names_seq
1349     { \tl_to_str:N \l__codedoc_macro_documented_tl }
1350   }
1351 }
1352 \cs_new_protected:Npn \__codedoc_macro_save_names_aux:n #1
1353 { \seq_gput_right:Nn \g__codedoc_nested_names_seq {#1} }

```

(End of definition for __codedoc_macro_save_names:.)

`__codedoc_macro_exclude_index:` Some control sequences in a `macrocode` environment shouldn't be indexed, for different reasons. This macro parses the argument of the `do-not-index` option and locally removes the given macros from the index.

The optional argument to `macro` is not scanned with verbatim catcodes, so we use `\tl_set_rescan:NnV` to rescan the commands with the same catcodes as `\DoNotIndex`. The scanned token list contains spaces after control sequences, which are not there when `\DoNotIndex` is used. Since `\seq_set_from_clist:Nn` removes spaces around the items, we can abuse that and `\seq_use:Nn` to normalise each item. After that `\DoNotIndex` can do its thing.

```

1354 \cs_new_protected:Npn \__codedoc_macro_exclude_index:
1355 {
1356   \tl_if_empty:NF \l__codedoc_macro_do_not_index_tl
1357   {
1358     \tl_set_rescan:NnV \l__codedoc_macro_do_not_index_tl
1359     { \MakePrivateLetters \catcode`\12 }
1360     \l__codedoc_macro_do_not_index_tl
1361     \exp_args:NNV \seq_set_from_clist:Nn
1362     \l__codedoc_tmpa_seq \l__codedoc_macro_do_not_index_tl
1363     \__kernel_tl_set:Ne \l__codedoc_macro_do_not_index_tl
1364     { \seq_use:Nn \l__codedoc_tmpa_seq { , } }
1365     \exp_args:NV \DoNotIndex \l__codedoc_macro_do_not_index_tl
1366   }
1367 }

```

(End of definition for `__codedoc_macro_exclude_index:.`)

`__codedoc_macro_dump:` This calls `\makelabel{}`

```

1368 \cs_new_protected:Npn \__codedoc_macro_dump:
1369 {
1370   \topsep\MacroTopsep
1371   \trivlist
1372   \cs_set:Npn \makelabel ##1
1373   {
1374     \llap
1375     {
1376       \hbox_unpack_drop:N \l__codedoc_macro_index_box
1377       \vtop to \baselineskip
1378       {
1379         \vbox_unpack_drop:N \l__codedoc_macro_box
1380         \vss
1381       }
1382     }
1383   }
1384   \item [ ]
1385 }
```

(End of definition for `__codedoc_macro_dump:.`)

`__codedoc_macro_typeset_block:nN` Used to typeset a macro and its variants. #1 is the macro name, #2 is a boolean controlling whether to add TF.

```

1386 \cs_new_protected:Npn \__codedoc_macro_typeset_block:nN #1#2
1387 {
1388   \__codedoc_macro_single:nNN {#1} \c_true_bool #2
1389   \seq_if_empty:NF \g__codedoc_variants_seq
1390   {
1391     \__codedoc_macro_typeset_variant_list:eN
1392     { \__codedoc_get_function_name:n {#1} } #2
1393   }
1394 }
1395 \cs_new_protected:Npn \__codedoc_macro_typeset_variant_list:nN #1#2
1396 {
1397   \seq_map_inline:Nn \g__codedoc_variants_seq
1398   { \__codedoc_macro_single:nNN { #1 : #1 } \c_false_bool #2 }
1399 }
1400 \cs_generate_variant:Nn \__codedoc_macro_typeset_variant_list:nN { e }
```

(End of definition for `__codedoc_macro_typeset_block:nN.`)

`__codedoc_macro_single:nNN` The arguments are `#1` a macro name (without TF), `#2` a boolean determining whether or not to index, and `#3` whether or not to add TF. Let's start to mess around with doc's `macro` environment. See `doc.dtx` for a full explanation of the original environment. It's rather *enthusiastically* commented.

`#1` : Macro/function/whatever name; input has already been sanitised.

The assignments to `\saved@macroname` and `\saved@indexname` are used by doc's `\changes` mechanism.

```

1401 \cs_new_protected:Npn \__codedoc_macro_single:nNN #1#2#3
1402 {
1403   \tl_set:Nn \saved@macroname {#1}
1404   \__codedoc_macro_typeset_one:nN {#1} #3
1405   \bool_if:NT #3 { \DoNotIndex {#1} }
1406   \exp_args:Ne \__codedoc_macro_index:nN
1407     { #1 \bool_if:NT #3 { \tl_to_str:n { TF } } }
1408   #2
1409 }
1410 \cs_new_protected:Npn \__codedoc_macro_index:nN #1#2
1411 {
1412   \DoNotIndex {#1}
1413   \bool_if:NT #2
1414   {
1415     \bool_lazy_any:nF
1416     {
1417       { \__codedoc_if_macro_internal_p:n {#1} }
1418       { \l__codedoc_macro_deprecated_bool }
1419       { \l__codedoc_macro_nodoc_bool }
1420     }
1421     { \seq_gput_right:Nn \g_doc_macros_seq {#1} }
1422     \hbox_set:Nw \l__codedoc_macro_index_box
1423     \hbox_unpack_drop:N \l__codedoc_macro_index_box
1424     \int_gincr:N \c@CodelineNo
1425     \__codedoc_special_index:nn {#1} { main }
1426     \int_gdecr:N \c@CodelineNo
1427     \exp_args:NNNo \hbox_set_end:
1428     \tl_set:Nn \saved@indexname { \l__codedoc_index_key_tl }
1429   }
1430 }

```

(End of definition for `__codedoc_macro_single:nNN`.)

`__codedoc_macro_typeset_one:nN` For a long time, `l3doc` collected the macro names as labels in the first items of nested `\trivlist`, but these were not closed properly with `\endtrivlist`. Also,

it interacted in surprising ways with `hyperref` targets. Now, we collect typeset macro names by hand in the box `\l__codedoc_macro_box`. The fixed-size space `\MacroFont\` could be replaced by an customizable horizontal space; it is important for it to be the same for all macros. `#1` is the macro name, `#2` whether to add TF.

```

1431 \cs_new_protected:Npn \__codedoc_macro_typeset_one:nN #1#2
1432 {
1433   \vbox_set:Nn \l__codedoc_macro_box
1434   {
1435     \vbox_unpack_drop:N \l__codedoc_macro_box
1436     \hbox { \llap { \__codedoc_print_macroname:nN {#1} #2
1437               \MacroFont \
1438             } }
1439   }
1440   \int_incr:N \l__codedoc_macro_int
1441 }

```

(End of definition for `__codedoc_macro_typeset_one:nN`.)

`__codedoc_print_macroname:nN` In the name, spaces are replaced by other spaces to ensure they get displayed in case there are any.

```

1442 \cs_new_protected:Npn \__codedoc_print_macroname:nN #1#2
1443 {
1444   \strut
1445   \__codedoc_get_hyper_target:eN
1446   {
1447     \exp_not:n {#1}
1448     \bool_if:NT #2 { \tl_to_str:n {TF} }
1449   }
1450   \l__codedoc_tmpa_tl
1451   \cs_if_exist:cTF { r@ \l__codedoc_tmpa_tl }
1452   { \exp_last_unbraced:NNo \hyperref [ \l__codedoc_tmpa_tl ] }
1453   { \use:n }
1454   {
1455     \int_compare:nTF { \str_count:n {#1} <= 28 }
1456     { \MacroFont } { \MacroLongFont }
1457     \tl_set:Nn \l__codedoc_tmpa_tl {#1}
1458     \tl_replace_all:NnV \l__codedoc_tmpa_tl
1459     { ~ } \c_catcode_other_space_tl
1460     \__codedoc_macroname_prefix:o \l__codedoc_tmpa_tl
1461     \__codedoc_macroname_suffix:N #2
1462   }

```

```

1463 }
1464 \cs_new_protected:Npn \__codedoc_macroname_prefix:n #1
1465 {
1466   \__codedoc_if_macro_internal:nTF {#1}
1467   { \__codedoc_typeset_aux:n {#1} } {#1}
1468 }
1469 \cs_generate_variant:Nn \__codedoc_macroname_prefix:n { o }
1470 \cs_new_protected:Npn \__codedoc_macroname_suffix:N #1
1471 { \bool_if:NTF #1 { \__codedoc_typeset_TF: } { } }

```

(End of definition for __codedoc_print_macroname:nN.)

\MacroLongFont

```

1472 \providecommand \MacroLongFont
1473 {
1474   \fontfamily{lmtt}\fontseries{lc}\small
1475 }

```

(End of definition for \MacroLongFont. This function is documented on page ??.)

__codedoc_print_testfile:n Used to show that a macro has a test, somewhere.

```

\__codedoc_print_testfile_aux:n 1476 \cs_new_protected:Npn \__codedoc_print_testfile:n #1
1477 {
1478   \bool_set_true:N \l__codedoc_macro_tested_bool
1479   \tl_if_eq:nnF {#1} {*}
1480   {
1481     \seq_if_in:NnF \g__codedoc_testfiles_seq {#1}
1482     {
1483       \seq_gput_right:Nn \g__codedoc_testfiles_seq {#1}
1484       \par
1485       \__codedoc_print_testfile_aux:n {#1}
1486     }
1487   }
1488 }
1489 \cs_new_protected:Npn \__codedoc_print_testfile_aux:n #1
1490 {
1491   \footnotesize
1492   (
1493   \textit
1494   {
1495     The~ test~ suite~ for~ this~ command,~
1496     and~ others~ in~ this~ file,~ is~ \textsf{#1}
1497   }~.

```

```

1498     )\par
1499   }

```

(End of definition for `__codedoc_print_testfile:n` and `__codedoc_print_testfile_aux:n`.)

`\TestFiles`

```

1500 \DeclareDocumentCommand \TestFiles {m}
1501 {
1502   \par
1503   \textit
1504   {
1505     The~ following~ test~ files~ are~
1506     used~ for~ this~ code:~ \textsf{#1}.
1507   }
1508   \par \ignorespaces
1509 }

```

(End of definition for `\TestFiles`. This function is documented on page ??.)

`\UnitTested`

```

1510 \DeclareDocumentCommand \UnitTested { } { \testfile* }

```

(End of definition for `\UnitTested`. This function is documented on page ??.)

`\TestMissing`

```

1511 \DeclareDocumentCommand \TestMissing { m }
1512 { \__codedoc_test_missing:n {#1} }

```

(End of definition for `\TestMissing`. This function is documented on page ??.)

`__codedoc_test_missing:n` Keys in `\g__codedoc_missing_tests_prop` are lists of macros given as arguments of one macro environment. Values are pairs of a file name and a comment about the missing tests.

```

1513 \cs_new_protected:Npn \__codedoc_test_missing:n #1
1514 {
1515   \__codedoc_test_missing_aux:Nen
1516   \g__codedoc_missing_tests_prop
1517   { \seq_use:Nn \l__codedoc_names_seq { , } }
1518   { { \g_file_curr_name_str \c_space_tl (#1) } }
1519 }
1520 \cs_new_protected:Npn \__codedoc_test_missing_aux:Nnn #1#2#3
1521 {
1522   \prop_get:NnNTF #1 {#2} \l__codedoc_tmpa_tl

```

```

1523     { \tl_put_right:Nn \l__codedoc_tmpa_tl { , #3 } }
1524     { \tl_set:Nn \l__codedoc_tmpa_tl {#3} }
1525     \prop_put:Nno #1 {#2} \l__codedoc_tmpa_tl
1526   }
1527   \cs_generate_variant:Nn \__codedoc_test_missing_aux:Nnn { Ne }

```

(End of definition for `__codedoc_test_missing:n`.)

`__codedoc_macro_end:` It is too late for anyone to declare a test file for this macro, so we can check now whether the macro is tested. If the `macro` environment which is being ended is the outermost one, then wrap each macro in `\texttt` (with the addition of `TF` if relevant) and typeset two informations: that this ends the definition of some macros, and that they are documented on some page.

```

1528   \cs_new_protected:Npn \__codedoc_macro_end:
1529   {
1530     \endtrivlist
1531     \__codedoc_macro_end_check_tested:
1532     \int_compare:nNnT \l__codedoc_nested_macro_int = 1
1533       { \__codedoc_macro_end_style:n { \__codedoc_print_end_definition: } }
1534   }

```

(End of definition for `__codedoc_macro_end:.`)

`__codedoc_macro_end_check_tested:` If the `checktest` option was issued and the macro is not an auxiliary nor a variable (and it does not have a test), then add it to the sequence of non-tested macros.

```

1535   \cs_new_protected:Npn \__codedoc_macro_end_check_tested:
1536   {
1537     \bool_lazy_all:nT
1538     {
1539       { \g__codedoc_checktest_bool }
1540       { ! \l__codedoc_macro_var_bool }
1541       { ! \l__codedoc_macro_tested_bool }
1542     }
1543     {
1544       \seq_set_filter:NNn \l__codedoc_tmpa_seq \l__codedoc_names_seq
1545       { ! \__codedoc_if_macro_internal_p:n {##1} }
1546       \seq_gput_right:Ne \g__codedoc_not_tested_seq
1547       {
1548         \seq_use:Nn \l__codedoc_tmpa_seq { , }
1549         \bool_if:NTF \l__codedoc_macro_pTF_bool {~(pTF)}
1550         { \bool_if:NT \l__codedoc_macro_TF_bool {~(TF)} }
1551       }

```



```

1552     }
1553 }

```

(End of definition for `__codeloc_macro_end_check_tested:.`)

`__codeloc_macro_end_style:n` Style for the extra information at the end of a top-level macro environment.

```

1554 \cs_new_protected:Npn \__codeloc_macro_end_style:n #1
1555 {
1556     \nobreak \noindent
1557     { \footnotesize ( \emph{#1} ) \par }
1558 }

```

(End of definition for `__codeloc_macro_end_style:n`.)

`__codeloc_print_end_definition:` Surround each item by `\texttt`, replacing `_` by `_` as well. Then list the macro names through `\seq_use:Nnnn`, unless there are too many. Finally, if the macro is neither auxiliary nor internal, add a link to where it is documented.

```

1559 \cs_new_protected:Npn \__codeloc_macro_end_wrap_item:n #1
1560 {
1561     \tl_set:Nn \l__codeloc_tmpa_tl {#1}
1562     \tl_replace_all:Nvn \l__codeloc_tmpa_tl
1563         \c_underscore_str { \_ }
1564     \texttt { \l__codeloc_tmpa_tl }
1565 }
1566 \cs_new_protected:Npn \__codeloc_print_end_definition:
1567 {
1568     \seq_set_map:Nnn \l__codeloc_tmpa_seq
1569         \g__codeloc_nested_names_seq
1570         { \__codeloc_macro_end_wrap_item:n {##1} }
1571     End~ of~ definition~ for~
1572     \int_compare:nTF { \seq_count:N \l__codeloc_tmpa_seq <= 3 }
1573     {
1574         \seq_use:Nnnn \l__codeloc_tmpa_seq
1575         { \,~and~ } { \,,~ } { \,,~and~ }
1576     }
1577     { \seq_item:Nn \l__codeloc_tmpa_seq {1}\,~and~others }
1578     \@.
1579     \__codeloc_print_documented:
1580 }
1581 \cs_new_protected:Npn \__codeloc_print_documented:
1582 {
1583     \seq_gset_filter:Nnn \g__codeloc_nested_names_seq
1584         \g__codeloc_nested_names_seq

```

```

1585     {
1586         ! \bool_lazy_any_p:n
1587         {
1588             { \__codedoc_if_macro_internal_p:n {##1} }
1589             { \l__codedoc_macro_deprecated_bool }
1590             { \l__codedoc_macro_nodoc_bool }
1591         }
1592     }
1593 \seq_if_empty:NF \g__codedoc_nested_names_seq
1594 {
1595     \int_set:Nn \l__codedoc_tmpa_int
1596     { \seq_count:N \g__codedoc_nested_names_seq }
1597     \int_compare:nNnTF \l__codedoc_tmpa_int = 1 {~This~} {~These~}
1598     \bool_if:NTF \l__codedoc_macro_var_bool {variable} {function}
1599     \int_compare:nNnTF \l__codedoc_tmpa_int = 1 {~is~} {s~are~}
1600     documented-on~page~
1601     \__codedoc_get_hyper_target:eN
1602     { \seq_item:Nn \g__codedoc_nested_names_seq { 1 } }
1603     \l__codedoc_tmpa_tl
1604     \exp_args:Ne \pageref { \l__codedoc_tmpa_tl } .
1605 }
1606 \seq_gclear:N \g__codedoc_nested_names_seq
1607 }

```

(End of definition for `__codedoc_print_end_definition:`, `__codedoc_macro_end_wrap_item:n`, and `__codedoc_print_documented:`.)

5.9.4 Misc

`\DescribeOption` For describing package options: retained for consistency, but updated for doc v3.

```

1608 \NewDocElement[idxtype = option, idxgroup = options]{Option}{optionenv}

```

(End of definition for `\DescribeOption`. This function is documented on page ??.)

Here are some definitions for additional markup that helps to structure your documentation.

```

danger (env.) \begin{[d]danger}
              dangerous code
ddanger (env.) \end{[d]danger}

```



Provides a danger bend, as known from the `TEXbook`.

The actual character from the font `manfnt`:

```

1609 \font \manual = manfnt \scan_stop:
1610 \cs_gset:Npn \dbend { {\manual\char127} }

```

Defines the single danger bend. Use it whenever there is a feature in your package that might be tricky to use. **FIXME:** Has to be fixed when in combination with a macro-definition.

```

1611 \newenvironment {danger}
1612 {
1613     \begin{trivlist}\item[]\noindent
1614     \begin{group}\hangindent=2pc\hangafter=-2
1615     \cs_set:Npn \par{\endgraf\endgroup}
1616     \hbox to0pt{\hskip-\hangindent\dbend\hfill}\ignorespaces
1617 }
1618 {
1619     \par\end{trivlist}
1620 }

```



Use the double danger bend if there is something which could cause serious problems when used in a wrong way. Better the normal user does not know about such things.

```

1621 \newenvironment {ddanger}
1622 {
1623     \begin{trivlist}\item[]\noindent
1624     \begin{group}\hangindent=3.5pc\hangafter=-2
1625     \cs_set:Npn \par{\endgraf\endgroup}
1626     \hbox to0pt{\hskip-\hangindent\dbend\kern2pt\dbend\hfill}\ignorespaces
1627 }{
1628     \par\end{trivlist}
1629 }

```

5.9.5 NB and NOTE

These macros are intended for additional notes added to the source that are not typeset.

\NB **\NB{wspr}{this is what I think about this!}**

```

1630 \bool_if:NTF \g__codedoc_show_notes_bool
1631 {
1632     \NewDocumentCommand\NB{mm}
1633     {
1634         (\emph{Note}\footnote{\ttfamily [#1]:~\detokenize{#2}})
1635     }
1636 }

```

```

1637 {
1638   \NewDocumentCommand\NB{mm}{-}
1639 }

```

(End of definition for \NB. This function is documented on page 8.)

```

NOTE (env.) \begin{NOTE}{wspr}
            this is what I #${\& think about this!
            \end{NOTE}

```

```

1640 \bool_if:NTF \g__codedoc_show_notes_bool
1641 {
1642   \NewDocumentEnvironment{NOTE}{m}
1643   {
1644     \par\noindent (\emph{Note}~[\texttt{#1}]):\par
1645     \verbatim
1646   }
1647   {
1648     \endverbatim
1649     \par\noindent \emph{Note~end})\par
1650   }
1651 }
1652 {
1653   \NewDocumentEnvironment{NOTE}{m}{\comment}{\endcomment}
1654 }

```

5.10 Footnote support

The environments `function` and `variable` are boxes and so looses footnotes. The following implements support. It relies currently on an internal from `hyperref` to get the correct targets.

```

1655 \providecommand\Hy@footnote@currentHref{}
1656 \prop_new:N\g__codedoc_fnmark_prop
1657 \cs_new_protected:Npn \__codedoc_fn_store:
1658 {
1659   \prop_gput:Nee\g__codedoc_fnmark_prop
1660     {fn\int_use:N\c@footnote}{\Hy@footnote@currentHref}{\int_use:N\c@footnote}}
1661 }
1662 \cs_new_protected:Npn \__codedoc_fn_restore:n #1
1663 {
1664   \prop_get:NnN \g__codedoc_fnmark_prop {fn#1}\l__codedoc_tmpa_tl

```

```

1665 \tl_gset:N\Hy@footnote@currentHref
1666 {\exp_last_unbraced:N\use_i:nn \l__codedoc_tmpa_tl }
1667 \setcounter{footnote}{\exp_last_unbraced:N\use_ii:nn \l__codedoc_tmpa_tl}
1668 }
1669
1670 \cs_generate_variant:Nn \hook_gput_next_code:nn {ne}
1671 \cs_new_protected:Npn \__codedoc_fn_footnote:nn #1 #2
1672 {
1673 \footnotemark
1674 \__codedoc_fn_store:
1675 \hook_gput_next_code:ne {env/#1/after}
1676 {\exp_not:N\__codedoc_fn_restore:n{\int_use:N\c@footnote}{\exp_not:n{\footnotetext{#2}}
1677
1678 \AddToHook{env/function/begin}{\def\footnote{\__codedoc_fn_footnote:nn{function}}}
1679 \AddToHook{env/variable/begin}{\def\footnote{\__codedoc_fn_footnote:nn{variable}}}

```

5.11 Documenting templates

```

1680 \newenvironment{TemplateInterfaceDescription}[1]
1681 {
1682 \subsection{The~object~type~`#1'}
1683 \begingroup
1684 \@beginparpenalty\@M
1685 \description
1686 \def\TemplateArgument##1##2{\item[Arg:~##1]##2\par}
1687 \def\TemplateSemantics
1688 {
1689 \enddescription\endgroup
1690 \subsubsection*{Semantics:}
1691 }
1692 }
1693 {
1694 \par\bigskip
1695 }
1696 \newenvironment{TemplateDescription}[2]
1697 {
1698 \subsection{The~template~`#2'~(object~type~#1)}
1699 \subsubsection*{Attributes:}
1700 \begingroup
1701 \@beginparpenalty\@M
1702 \description
1703 \def\TemplateKey##1##2##3##4

```

```

1704     {
1705         \item[##1~(##2)]##3%
1706         \ifx\TemplateKey##4\TemplateKey\else
1707 %         \hskip0ptplus3em\penalty-500\hskip 0pt plus 1filll Default:~##4%
1708         \hfill\penalty500\hbox{ }\hfill Default:~##4%
1709         \nobreak\hskip-\parfillskip\hskip0pt\relax
1710     \fi
1711     \par
1712 }
1713 \def\TemplateSemantics
1714 {
1715     \enddescription\endgroup
1716     \subsubsection*{Semantics~\&~Comments:}
1717 }
1718 }
1719 { \par \bigskip }

1720 \newenvironment{InstanceDescription}[4] [xxxxxxxxxxxxxxxx]
1721 {
1722     \subsubsection{The~instance~`#3'~(template~#2/#4)}
1723     \subsubsection*{Attribute~values:}
1724     \begingroup
1725     \@beginparpenalty\@M
1726     \def\InstanceKey##1##2{\>\textbf{##1}\>##2\\}
1727     \def\InstanceSemantics{\endtabbing\endgroup
1728         \vskip-30pt\vskip0pt
1729         \subsubsection*{Layout~description~\&~Comments:}}
1730     \tabbing
1731     xxxx\=#1\=\kill
1732 }
1733 { \par \bigskip }

```

5.12 Inheriting doc

Code here is taken from doc, stripped of comments and translated into expl3 syntax. New features are added in various places.

<pre> \StopEventually \MaybeStop \Finale \AlsoImplementation \OnlyDescription \g__codedoc_finale_tl </pre>	<pre> TODO: remove these four commands altogether, document that it is better to use the documentation and implementation environments. 1734 \DeclareDocumentCommand \OnlyDescription { } 1735 { \bool_gset_false:N \g__codedoc_typeset_implementation_bool } 1736 \DeclareDocumentCommand \AlsoImplementation { } 1737 { \bool_gset_true:N \g__codedoc_typeset_implementation_bool } </pre>
--	---

```

1738 \DeclareDocumentCommand \StopEventually { m }
1739 {
1740   \bool_if:NTF \g__codedoc_typeset_implementation_bool
1741   {
1742     \@bsphack
1743     \tl_gset:Nn \g__codedoc_finale_tl { #1 \check@checksum }
1744     \init@checksum
1745     \@esphack
1746   }
1747   { #1 \endinput }
1748 }

```

We also need to support doc V3 \MaybeStop if it is around (which may not be the case).

```

1749 \cs_if_exist:NT \MaybeStop
1750 { \RenewCommandCopy \MaybeStop \StopEventually }

1751 \DeclareDocumentCommand \Finale { }
1752 { \tl_use:N \g__codedoc_finale_tl }
1753 \tl_new:N \g__codedoc_finale_tl

```

(End of definition for \StopEventually and others. These functions are documented on page ??.)

__codedoc_input:n Inputting a file, with some setup: the module name should be empty before the first <@@=<module> line in the file.

```

1754 \cs_new_protected:Npn \__codedoc_input:n #1
1755 {
1756   \tl_gclear:N \g__codedoc_module_name_tl
1757   \MakePercentIgnore
1758   \input{#1}
1759   \MakePercentComment
1760 }

```

(End of definition for __codedoc_input:n.)

\DocInput Modified from doc to accept comma-list input (who has commas in filenames?).

```

1761 \DeclareDocumentCommand \DocInput { m }
1762 {
1763   \clist_map_inline:nn {#1}
1764   {
1765     \clist_put_right:Nn \g_docinput_clist {##1}
1766     \__codedoc_input:n {##1}
1767   }
1768 }

```

(End of definition for `\DocInput`. This function is documented on page ??.)

`\DocInputAgain` Uses `\g_docinput_clist` to re-input whatever's already been `\DocInput`-ed until now. May be used multiple times.

```
1769 \DeclareDocumentCommand \DocInputAgain { }
1770 { \clist_map_function:NN \g_docinput_clist \__codedoc_input:n }
```

(End of definition for `\DocInputAgain`. This function is documented on page ??.)

`\DocInclude` More or less exactly the same as `\include`, but uses `\DocInput` on a `.dtx` file, not `\input` on a `.tex` file.

```
1771 \NewDocumentCommand \DocInclude { m }
1772 {
1773   \relax\clearpage
1774   \docincludeaux
1775   \IfFileExists{#1.fdd}
1776     { \cs_set:Npn \currentfile{#1.fdd} }
1777     { \cs_set:Npn \currentfile{#1.dtx} }
1778   \int_compare:nNnTF \@auxout = \@partaux
1779     { \@latexerr{\string\include\space cannot~be~nested}\@eha }
1780     { \@docinclude {#1} }
1781 }

1782 \cs_gset:Npn \@docinclude #1
1783 {
1784   \clearpage
1785   \immediate\write\@mainaux{\string\@input{#1.aux}}
1786   \@tempswtrue
1787   \if@partsw
1788     \@tempswafalse
1789     \cs_set:Npe \@tempb {#1}
1790     \clist_map_inline:Nn \@partlist
1791       {
1792         \if_meaning:w \@tempa \@tempb
1793           \@tempswtrue
1794         \fi:
1795       }
1796   \fi
1797   \if@tempswa
1798     \cs_set_eq:NN \@auxout \@partaux
1799     \immediate\openout\@partaux #1.aux
1800     \immediate\write\@partaux{\relax}
1801     \cs_set_eq:NN \@ltxdoc@PrintIndex \PrintIndex
```



```

1802     \cs_set_eq:NN \PrintIndex          \relax
1803     \cs_set_eq:NN \@ltxdoc@PrintChanges \PrintChanges
1804     \cs_set_eq:NN \PrintChanges        \relax
1805     \cs_set_eq:NN \@ltxdoc@theglossary  \theglossary
1806     \cs_set_eq:NN \@ltxdoc@endtheglossary \endtheglossary
1807     \part{\currentfile}
1808     {
1809         \cs_set_eq:NN \ttfamily\relax
1810         \cs_gset:Npe \filekey
1811         { \filekey,~ \thepart = { \ttfamily \currentfile } }
1812     }
1813     \DocInput{\currentfile}
1814     \cs_set_eq:NN \PrintIndex          \@ltxdoc@PrintIndex
1815     \cs_set_eq:NN \PrintChanges        \@ltxdoc@PrintChanges
1816     \cs_set_eq:NN \theglossary        \@ltxdoc@theglossary
1817     \cs_set_eq:NN \endtheglossary     \@ltxdoc@endtheglossary
1818     \clearpage
1819     \@writeckpt{#1}
1820     \immediate \closeout \@partaux
1821     \else
1822         \@nameuse{cp@#1}
1823     \fi
1824     \cs_set_eq:NN \@auxout \@mainaux
1825 }

```

Here, MMMMI (for page references) and MMMMV (for codeline references) are interpreted by `makeindex` as an uppercase Roman number pages, and should be large enough to avoid collisions with other uses of uppercase Roman number pages. Two subtle differences between `\@wrindex` and `\codeline@wrindex` are that the first must be a delayed write because the page number is not known yet, and it must close a group and finish some space-hack.

We also provide versions for our use that refer

```

1826 \cs_gset_protected:Npn \@wrindex #1
1827 {
1828     \protected@write \@indexfile {}
1829     { \string \indexentry {#1} { MMMMI - \thepage } }
1830     \endgroup \@esphack
1831 }
1832 \cs_gset_protected:Npn \codeline@wrindex #1
1833 {
1834     \immediate\write\@indexfile
1835     {

```

```

1836         \string\indexentry{#1}
1837         { MMMMV - \filesep \int_use:N \c@CodelineNo }
1838     }
1839 }
1840 \tl_gclear:N \filesep
1841 \cs_new_protected:Npn \__codedoc_index_page_hc:nn #1#2
1842 {
1843     \protected@write \@indexfile {}
1844     {
1845         \string \indexentry { #1 \encapchar hdpindex{#2} }
1846         { MMMMI - \thepage }
1847     }
1848 }
1849 \cs_new_protected:Npn \__codedoc_index_codeline_hc:nn #1#2
1850 {
1851     \immediate\write\@indexfile
1852     {
1853         \string \indexentry { #1 \encapchar hdclindex{\the\c@HD@hypercount}{#2} }
1854         { MMMMV - \filesep \int_use:N \c@CodelineNo - MMMD - \the\c@HD@hypercount - M }
1855     }
1856 }

```

We already have a single HD.xx target per code line. It would be better to have a target CL.\the\c@CodelineNo per code line and change hdclindex{\the\c@HD@hypercount} to a mechanism closer to hdpindex, but we need to understand better the different types of indexings, and there are subtleties with indexing \{ and \}.

(End of definition for \DocInclude. This function is documented on page ??.)

`\docincludeaux`

```

1857 \cs_gset:Npn \docincludeaux
1858 {
1859     \tl_set:Nn \thepart { \alphalph { part } }
1860     \tl_set:Nn \filesep { \thepart - }
1861     \cs_set_eq:NN \filekey \use_none:n
1862     \tl_gput_right:Nn \index@prologue
1863     {
1864         \cs_gset:Npn \@oddfoot
1865         {
1866             \parbox { \textwidth }
1867             {
1868                 \strut \footnotesize
1869                 \raggedright { \bfseries File-Key: } ~ \filekey

```

```

1870         }
1871     }
1872     \cs_set_eq:NN \@evenfoot \@oddfoot
1873 }
1874 \cs_gset_eq:NN \docincludeaux \relax
1875 \cs_gset:Npn \@oddfoot
1876 {
1877     \cs_if_exist:cTF { ver @ \currentfile }
1878     { File~\thepart :~{\ttfamily\currentfile}~ }
1879     {
1880         \GetFileInfo{\currentfile}
1881         File~\thepart :~{\ttfamily\filename}~
1882         Date:~\ExplFileDate\ % space
1883         Version~\ExplFileVersion
1884     }
1885     \hfill \thepage
1886 }
1887 \cs_set_eq:NN \@evenfoot \@oddfoot
1888 }

```

(End of definition for `\docincludeaux`. This function is documented on page ??.)

5.12.1 The macrocode environment

`\xmacro@code` Hook into the `macrocode` environment in a dirty way: `\xmacro@code` is responsible for grabbing (and tokenizing) the body of the environment. Redefine it to pass what it grabs to `__codedoc_xmacro_code:n`. This new macro replaces all `<<=>` by the appropriate module name. One exceptional case is the `<<=<module>>` lines themselves, where `<<=>` should not be modified. Actually, we search for such lines, to set the module name automatically. We need to be careful: no `<<=>` should appear as such in the code below since `l3doc` is also typeset using this code. At each `<<=>` found, replace the `<module>` in the code behind it, update the `<module>`, and loop to check for further occurrences of `<<=>`.

```

1889 \group_begin:
1890   \char_set_catcode_other:N \^^A
1891   \char_set_catcode_active:N \^^S
1892   \char_set_catcode_active:N \^^B
1893   \char_set_catcode_other:N \^^L
1894   \char_set_catcode_other:N \^^R
1895   \char_set_lccode:nn { \^^A } { \% }
1896   \char_set_lccode:nn { \^^S } { \ }

```

```

1897 \char_set_lccode:nn { ``^B } { ``\ }
1898 \char_set_lccode:nn { ``^L } { ``{ }
1899 \char_set_lccode:nn { ``^R } { ``} }
1900 \tex_lowercase:D
1901 {
1902   \group_end:
1903   \cs_set_protected:Npn \xmacro@code
1904     #1 ^^A ^^S^^S^^S^^S ^^Bend ^^Lmacrocode^^R
1905     { \__codedoc_xmacro_code:n {#1} \end{macrocode} }
1906 }
1907 \group_begin:
1908 \char_set_catcode_active:N \<
1909 \char_set_catcode_active:N \>
1910 \cs_new_protected:Npn \__codedoc_xmacro_code:n #1
1911 {
1912   \tl_clear:N \l__codedoc_tmpa_tl
1913   \tl_if_in:nnTF {#1} { < @ @ = }
1914   { \__codedoc_xmacro_code:w #1 < @ @ = \q_recursion_tail > \q_recursion_stop }
1915   {
1916     \tl_set:Nn \l__codedoc_tmpa_tl {#1}
1917     \__codedoc_detect_internals:N \l__codedoc_tmpa_tl
1918     \__codedoc_replace_at_at:N \l__codedoc_tmpa_tl
1919     \tl_use:N \l__codedoc_tmpa_tl
1920   }
1921 }
1922 \cs_new_protected:Npn \__codedoc_xmacro_code:w #1 < @ @ = #2 >
1923 {
1924   % Add code before <@@=...>
1925   \tl_set:Nn \l__codedoc_tmpb_tl {#1}
1926   \__codedoc_detect_internals:N \l__codedoc_tmpb_tl
1927   \__codedoc_replace_at_at:N \l__codedoc_tmpb_tl
1928   \tl_put_right:NV \l__codedoc_tmpa_tl \l__codedoc_tmpb_tl
1929   % Check for \q_recursion_tail
1930   \quark_if_recursion_tail_stop_do:nn {#2}
1931   { \tl_use:N \l__codedoc_tmpa_tl }
1932   % Change module name and add <@@=#2> to typeset output
1933   \tl_gset:Nn \g__codedoc_module_name_tl {#2}
1934   \tl_put_right:Nn \l__codedoc_tmpa_tl { < \text { \verbatim@font @ @ = #2 } > }
1935   % Loop
1936   \__codedoc_xmacro_code:w
1937 }
1938 \group_end:

```

(End of definition for `\xmacro@code`, `__codedoc_xmacro_code:n`, and `__codedoc_xmacro_code:w`. This function is documented on page ??.)

5.13 At end document

Print all defined and documented macros/functions.

```

1939 \iow_new:N \g__codedoc_func_iow

1940 \tl_new:N \l__codedoc_doc_def_tl
1941 \tl_new:N \l__codedoc_doc_undef_tl
1942 \tl_new:N \l__codedoc_undoc_def_tl
1943 \tl_const:Nn \c__codedoc_iow_separator_tl { ---- }
1944 \tl_const:Nn \c__codedoc_iow_midrule_tl { -- }

1945 \cs_new_protected:Npn \__codedoc_show_functions_defined:
1946 {
1947   \bool_lazy_and:nnT
1948     { \g__codedoc_typeset_implementation_bool } { \g__codedoc_checkfunc_bool }
1949   {
1950     \iow_term:e { \c__codedoc_iow_separator_tl \iow_newline: }
1951     \iow_open:Nn \g__codedoc_func_iow { \c_sys_jobname_str .cmds }
1952
1953     \tl_clear:N \l__codedoc_doc_def_tl
1954     \tl_clear:N \l__codedoc_doc_undef_tl
1955     \tl_clear:N \l__codedoc_undoc_def_tl
1956     \seq_gremove_duplicates:N \g_doc_functions_seq
1957     \seq_gremove_duplicates:N \g_doc_macros_seq
1958     \seq_map_inline:Nn \g_doc_functions_seq
1959       {
1960         \seq_if_in:NnTF \g_doc_macros_seq {##1}
1961         {
1962           \tl_put_right:Ne \l__codedoc_doc_def_tl
1963             { \iow_newline: > ~ ##1 }
1964         }
1965         {
1966           \tl_put_right:Ne \l__codedoc_doc_undef_tl
1967             { \iow_newline: ! ~ ##1 }
1968         }
1969       }
1970     \seq_map_inline:Nn \g_doc_macros_seq
1971       {
1972         \seq_if_in:NnF \g_doc_functions_seq {##1}
1973         {

```

```

1974         \tl_put_right:Ne \l__codedoc_undoc_def_tl
1975         { \iow_newline: ? ~ ##1 }
1976     }
1977 }
1978 \__codedoc_functions_typeout:nN
1979 {
1980     Functions~both~documented~and~defined: \iow_newline:
1981     (In~order~of~being~documented)
1982 }
1983 \l__codedoc_doc_def_tl
1984 \__codedoc_functions_typeout:nN
1985 { Functions~documented~but~not~defined: }
1986 \l__codedoc_doc_undef_tl
1987 \__codedoc_functions_typeout:nN
1988 { Functions~defined~but~not~documented: }
1989 \l__codedoc_undoc_def_tl
1990
1991 \iow_close:N \g__codedoc_func_iow
1992 \iow_term:e { \c__codedoc_iow_separator_tl }
1993 }
1994 }
1995 \AtEndDocument { \__codedoc_show_functions_defined: }

    TODO: use \iow_term:e.

1996 \cs_new_protected:Npn \__codedoc_functions_typeout:nN #1#2
1997 {
1998     \tl_if_empty:NF #2
1999     {
2000         \iow_now:Ne \g__codedoc_func_iow
2001         {
2002             \c__codedoc_iow_midrule_tl \iow_newline:
2003             #1 \iow_newline:
2004             \c__codedoc_iow_midrule_tl
2005             #2
2006         }
2007         \tl_clear:N #2
2008     }
2009 }

2010 \cs_new_protected:Npn \__codedoc_show_not_tested:
2011 {
2012     \bool_if:NT \g__codedoc_checktest_bool
2013     {

```

```

2014 \tl_clear:N \l__codedoc_tmpa_tl
2015 \prop_if_empty:NF \g__codedoc_missing_tests_prop
2016 {
2017   \cs_set:Npn \__codedoc_tmpa:w ##1##2
2018   {
2019     \iow_newline:
2020     \space\space\space\space \exp_not:n {##1}
2021     \clist_map_function:nN {##2} \__codedoc_tmpb:w
2022   }
2023   \cs_set:Npn \__codedoc_tmpb:w ##1
2024   {
2025     \iow_newline:
2026     \space\space\space\space\space\space * ~ ##1
2027   }
2028   \tl_put_right:Ne \l__codedoc_tmpa_tl
2029   {
2030     \iow_newline: \iow_newline:
2031     The~ following~ macro(s)~ have~ incomplete~ tests:
2032     \iow_newline:
2033     \prop_map_function:NN
2034     \g__codedoc_missing_tests_prop \__codedoc_tmpa:w
2035   }
2036 }
2037 \seq_if_empty:NF \g__codedoc_not_tested_seq
2038 {
2039   \cs_set:Npn \__codedoc_tmpa:w ##1
2040   { \clist_map_function:nN {##1} \__codedoc_tmpb:w }
2041   \cs_set:Npn \__codedoc_tmpb:w ##1
2042   {
2043     \iow_newline:
2044     \space\space\space\space ##1
2045   }
2046   \tl_put_right:Ne \l__codedoc_tmpa_tl
2047   {
2048     \iow_newline:
2049     \iow_newline:
2050     The~ following~ macro(s)~ do~ not~ have~ any~ tests:
2051     \iow_newline:
2052     \seq_map_function:NN
2053     \g__codedoc_not_tested_seq \__codedoc_tmpa:w
2054   }
2055 }

```

```

2056     \tl_if_empty:NF \l__codedoc_tmpa_tl
2057     {
2058         \int_set:Nn \l__codedoc_tmpa_int { \tex_interactionmode:D }
2059         \errorstopmode
2060         \ClassError { l3doc } { \l__codedoc_tmpa_tl } { }
2061         \int_set:Nn \tex_interactionmode:D { \l__codedoc_tmpa_int }
2062     }
2063 }
2064 }
2065 \AtEndDocument { \__codedoc_show_not_tested: }

```

5.14 Indexing

5.14.1 Necessary patching

The following is useful to set up `hyperref` targets, for instance for the purpose of indexing. Contrarily to `hypdoc` we do not try to save pdf destinations, as this leads to too many pdf \TeX warnings on early runs.

```

2066 \cs_new_protected:Npn \__codedoc_target:
2067 {
2068     \mode_leave_vertical:
2069     \group_begin:
2070         \HD@savedestfalse \HD@target
2071     \group_end:
2072 }

```

Force targets on every code line.

```

2073 \cs_set_nopar:Npe \theCodelineNo
2074 {
2075     \group_begin:
2076         \exp_not:N \HD@savedestfalse
2077         \exp_not:o \theCodelineNo
2078     \group_end:
2079 }

```

Inside the table of contents (and other similar lists introduced by `\@starttoc`), we suppress indexing. This is because `\cmd`, `\cs`, or `\tn` appearing in titles only gets typeset in the second run, and getting their indexing right would require even more runs than we already need. Besides, it is not useful to index uses of some command in the table of contents.

```

2080 \bool_new:N \l__codedoc_allow_indexing_bool
2081 \bool_set_true:N \l__codedoc_allow_indexing_bool

```



```

2082 \use:e
2083 {
2084   \exp_not:n { \cs_set_nopar:Npn \@starttoc #1 }
2085   {
2086     \group_begin:
2087       \bool_set_false:N \l__codedoc_allow_indexing_bool
2088       \exp_not:o { \@starttoc {#1} }
2089     \group_end:
2090   }
2091 }

```

5.14.2 Userspace commands

Fix index (for now):

```

2092 \g@addto@macro \theindex { \MakePrivateLetters }
2093 \cs_gset:Npn \verbatimchar {&}
2094 \setcounter { IndexColumns } { 2 }

```

Set up the Index to use \part

```

2095 \IndexPrologue
2096 {
2097   \part*{Index}
2098   \markboth{Index}{Index}
2099   \addcontentsline{toc}{part}{Index}
2100   The~italic~numbers~denote~the~pages~where~the~
2101   corresponding~entry~is~described,~
2102   numbers~underlined~point~to~the~definition,~
2103   all~others~indicate~the~places~where~it~is~used.
2104 }

```

`\SpecialIndex` An attempt at affecting how commands which appear within the macrocode environment are treated in the index.

```

2105 \cs_gset_protected:Npn \SpecialIndex #1
2106 {
2107   \@bsphack
2108   \__codedoc_special_index:nn {#1} { }
2109   \@esphack
2110 }

```

(End of definition for \SpecialIndex. This function is documented on page ??.)

```

2111 \msg_new:nnn { l3doc } { print-index-howto }

```

```

2112 {
2113     Generate~the~index~by~executing\\
2114     \iow_indent:n
2115     { makeindex--s~gind.ist~-o~\c_sys_jobname_str.ind~\c_sys_jobname_str.idx }
2116 }
2117 \tl_gput_right:Nn \PrintIndex
2118 { \AtEndDocument { \msg_info:nn { l3doc } { print-index-howto } } }

```

5.14.3 Internal index commands

`\it@is@a` The index of one-character commands within the `macrocode` environment is produced using `\it@is@a <char>`. Alter that command.

```

2119 \cs_gset_protected:Npn \it@is@a #1
2120 {
2121     \use:e
2122     {
2123         \__codedoc_special_index_module:nnnnN
2124         {#1}
2125         { \bslash #1 }
2126         { }
2127         { }
2128         \c_false_bool
2129     }
2130 }

```

(End of definition for `\it@is@a`. This function is documented on page ??.)

`__codedoc_special_index:nn`

```

2131 \cs_new_protected:Npn \__codedoc_special_index:nn #1#2
2132 {
2133     \__codedoc_key_get:n {#1}
2134     \quark_if_no_value:NF \l__codedoc_override_module_tl
2135     { \tl_set_eq:NN \l__codedoc_index_module_tl \l__codedoc_override_module_tl }
2136     \__codedoc_special_index_module:oonN
2137     { \l__codedoc_index_key_tl }
2138     { \l__codedoc_index_macro_tl }
2139     { \l__codedoc_index_module_tl }
2140     {#2}
2141     \l__codedoc_index_internal_bool
2142 }
2143 \cs_generate_variant:Nn \__codedoc_special_index:nn { o }

```

(End of definition for `__codedoc_special_index:nn`.)

`_codedoc_special_index_module:nnnnN` Remotely based on Heiko's replacement to play nicely with hypdoc. We use `\verb`
`_codedoc_special_index_module:ooonN` or a `\verbatim@font` construction depending on whether the number of tokens in
`_codedoc_special_index_aux:nnnnnn` **#2** is equal to its number of characters: if it is not then that suggests that there is a
`_codedoc_special_index_set:Nn` construct such as `\meta{...}`.

```

2144 \tl_new:N \l__codedoc_index_escaped_macro_tl
2145 \tl_new:N \l__codedoc_index_escaped_key_tl

2146 \cs_new_protected:Npn \__codedoc_special_index_module:nnnnN #1#2#3#4#5

```

#1 : key
#2 : macro
#3 : module
#4 : index ‘type’ (*main/usage/etc.*)
#5 : boolean whether internal command

```

2147 {
2148   \use:e
2149   {
2150     \exp_not:n { \__codedoc_special_index_aux:nnnnnn {#1} {#2} }
2151     \tl_if_empty:nTF {#3}
2152       { { } { } { } }
2153       {
2154         \str_if_eq:nnTF {#3} { TeX }
2155         {
2156           { TeX~and~LaTeX2e }
2157           { \string\TeX{ }~and~\string\LaTeXe{ } }
2158         }
2159         {
2160           {#3}
2161           { \string\pkg{#3} }
2162         }
2163         { \bool_if:NT #5 { ~internal } ~commands: }
2164       }
2165     }
2166     {#4}
2167   }

```

```

2168 \cs_generate_variant:Nn \__codedoc_special_index_module:nnnnN { ooo }

```

```

2169 \cs_new_protected:Npn \__codedoc_special_index_aux:nnnnnn #1#2#3#4#5#6

```

#1 : key
#2 : macro

#3 : index subheading string
 #4 : index subheading text
 #5 : index subheading suffix (appended to both arg 3 and 4)
 #6 : index ‘type’ (*main/usage/etc.*)

```

2170 {
2171   \tl_set:Nn \l__codedoc_index_escaped_key_tl {#1}
2172   \__codedoc_quote_special_char:N \l__codedoc_index_escaped_key_tl
2173   \__codedoc_special_index_set:Nn \l__codedoc_index_escaped_macro_tl {#2}
2174   \str_if_eq:onTF { \@currentenv } { macrocode }
2175     { \__codedoc_index_codeline_hc:nn }
2176     {
2177       \str_case:nnF {#6}
2178       {
2179         { main } { \__codedoc_index_codeline_hc:nn }
2180         { usage } { \__codedoc_index_page_hc:nn }
2181       }
2182       { \__codedoc_target: \__codedoc_index_page_hc:nn }
2183     }
2184     {
2185       \tl_if_empty:nF { #3 #4 #5 }
2186       { #3 #5 \actualchar #4 #5 \levelchar }
2187       \l__codedoc_index_escaped_key_tl
2188       \actualchar
2189       {
2190         \token_to_str:N \verbatim@font \c_space_tl
2191         \l__codedoc_index_escaped_macro_tl
2192       }
2193     }
2194     {#6}
2195   }

```

Note that #3 here could contain MMMMI- or MMMMV- more than once if several successive code lines have been merged into a range somehow. Note incidentally that the dash is active in some of our sources, like *interface3.tex* or *source2e.tex*.

```

2196 \group_begin:
\hdpindex 2197 \char_set_active_eq:NN - \scan_stop:
\__codedoc_old_hdpindex:nn 2198 \tl_const:Ne \c__codedoc_active_minus_tl { \char_generate:nn { - } { 13 } }
\hdclindex 2199 \group_end:
\__codedoc_old_hdclindex:nnn 2200 \cs_new_eq:NN \__codedoc_old_hdpindex:nn \hdpindex
\__codedoc_hdindex:nn 2201 \cs_new_eq:NN \__codedoc_old_hdclindex:nnn \hdclindex
\c__codedoc_active_minus_tl 2202 \cs_gset_protected:Npn \hdpindex #1
\__codedoc_hdindex_aux:nn
\__codedoc_hdindex_aux:w

```

```

2203 { \__codedoc_hdindex:nn { \__codedoc_old_hdpindex:nn {#1} } }
2204 \cs_gset_protected:Npn \hdclindex #1#2
2205 { \__codedoc_hdindex:nn { \__codedoc_old_hdclindex:nnn {#1} {#2} } }
2206 \cs_new_protected:Npn \__codedoc_hdindex:nn #1#2
2207 {
2208   \tl_set:Nn \l__codedoc_tmpa_tl {#2}
2209   \tl_replace_all:Nen \l__codedoc_tmpa_tl
2210     { \exp_not:V \c__codedoc_active_minus_tl \exp_not:V \c__codedoc_active_minus_tl }
2211     { -- }
2212   \seq_set_split:NnV \l__codedoc_tmpa_seq { -- } \l__codedoc_tmpa_tl
2213   \seq_set_map:NnN \l__codedoc_tmpa_seq \l__codedoc_tmpa_seq
2214     { \__codedoc_hdindex_aux:nn {#1} {##1} }
2215   \seq_use:Nn \l__codedoc_tmpa_seq { -- }
2216 }
2217 \cs_new_protected:Npn \__codedoc_hdindex_aux:nn #1#2
2218 {
2219   \tl_set:Nn \l__codedoc_tmpa_tl {#2}
2220   \tl_replace_all:Nnn \l__codedoc_tmpa_tl { MMMM } { \use_none:nn }
2221   \tl_if_in:NnT \l__codedoc_tmpa_tl { MMMD }
2222   {
2223     \tl_replace_all:Nen \l__codedoc_tmpa_tl
2224       { \exp_not:V \c__codedoc_active_minus_tl MMMD } { - MMMD }
2225     \tl_replace_all:Nnn \l__codedoc_tmpa_tl { - MMMD } { \__codedoc_hdindex_aux:w }
2226   }
2227   \use:e { \exp_not:n {#1} { \exp_not:V \l__codedoc_tmpa_tl } }
2228 }
2229 \cs_new_protected:Npn \__codedoc_hdindex_aux:w #1 M { }

2230 \cs_new_protected:Npn \__codedoc_special_index_set:Nn #1#2
2231 {
2232   \__kernel_tl_set:Ne #1 { \tl_to_str:n {#2} }
2233   \__codedoc_if_almost_str:nTF {#2}
2234   {
2235     \tl_replace_all:Nen #1 { \tl_to_str:n { _ } }
2236     {
2237       \verbatimchar
2238       \token_to_str:N \_ \token_to_str:N \_
2239       \token_to_str:N \verb * \verbatimchar
2240     }
2241     \exp_args:Ne \tl_map_inline:nn
2242       { \tl_to_str:N \verbatimchar \token_to_str:N _ }
2243     {
2244       \tl_replace_all:Nnn #1 {##1}

```

```

2245         {
2246             \verbatimchar \c_backslash_str ##1
2247             \token_to_str:N \verb * \verbatimchar
2248         }
2249     }
2250     \__kernel_tl_set:Ne #1
2251     {
2252         \token_to_str:N \verb * \verbatimchar
2253         #1 \verbatimchar
2254     }
2255 }
2256 {
2257     \tl_set:Nn #1 {#2}
2258     \tl_replace_all:Nvn #1
2259         \c_backslash_str
2260         { \token_to_str:N \bslash \c_space_tl }
2261 }
2262 \__codedoc_quote_special_char:N #1
2263 }

```

(End of definition for `__codedoc_special_index_module:nnnnN` and others. These functions are documented on page ??.)

`__codedoc_quote_special_char:N` Quote some special characters.

```

2264 \cs_new_protected:Npn \__codedoc_quote_special_char:N #1
2265 {
2266     \tl_map_inline:nn { \quotechar \actualchar \encapchar \levelchar \bslash }
2267     {
2268         \tl_replace_all:Nen #1
2269         { \tl_to_str:N ##1 } { \quotechar \tl_to_str:N ##1 }
2270     }
2271 }

```

(End of definition for `__codedoc_quote_special_char:N`.)

5.14.4 Finding sort-key and module

`__codedoc_key_get:n` Sets `\l__codedoc_index_macro_tl`, `\l__codedoc_index_key_tl`, and `\l__codedoc_index_module_tl` from `#1`. The base function is stored by `__codedoc_key_get_base:nN` in `\l__codedoc_index_macro_tl`, falling back to `#1` if it contains markup or has no signature.

The starting point for the $\langle key \rangle$ is `\l__codedoc_index_key_tl` as a string. If it the first character is a backslash, remove it. Then recognize `expl` functions and

variables by the presence of `:` or `_` and $\text{T}_{\text{E}}\text{X}/\text{L}\text{A}\text{T}_{\text{E}}\text{X} 2_{\varepsilon}$ commands by the presence of `@`. For `expl` names, we call `__codedoc_key_func:` or `__codedoc_key_var:`, which are responsible for removing some characters and finding the module name, while for $\text{T}_{\text{E}}\text{X}/\text{L}\text{A}\text{T}_{\text{E}}\text{X} 2_{\varepsilon}$ commands the module name is `TeX`, and others have an empty module name.

```

2272 \cs_new_protected:Npe \__codedoc_key_get:n #1
2273 {
2274   \exp_not:N \__codedoc_key_get_base:nN {#1} \exp_not:N \l__codedoc_index_macro_tl
2275   \__kernel_tl_set:Ne \exp_not:N \l__codedoc_index_key_tl
2276   { \exp_not:N \tl_to_str:N \exp_not:N \l__codedoc_index_macro_tl }
2277   \tl_clear:N \exp_not:N \l__codedoc_index_module_tl
2278   \tl_if_in:NnTF \exp_not:N \l__codedoc_index_key_tl { \tl_to_str:n { __ } }
2279   { \bool_set_true:N \exp_not:N \l__codedoc_index_internal_bool }
2280   { \bool_set_false:N \exp_not:N \l__codedoc_index_internal_bool }
2281   \exp_not:N \tl_if_head_eq_charcode:VNT
2282   \exp_not:N \l__codedoc_index_key_tl \c_backslash_str
2283   { \exp_not:N \__codedoc_key_pop: }
2284   \tl_if_in:NnTF \exp_not:N \l__codedoc_index_key_tl { \token_to_str:N : }
2285   { \exp_not:N \__codedoc_key_func: }
2286   {
2287     \tl_if_in:NnTF \exp_not:N \l__codedoc_index_key_tl { \token_to_str:N _ }
2288     { \exp_not:N \__codedoc_key_var: }
2289     {
2290       \tl_if_in:NnT \exp_not:N \l__codedoc_index_key_tl { \token_to_str:N @ }
2291       { \tl_set:Nn \exp_not:N \l__codedoc_index_module_tl { TeX } }
2292     }
2293   }
2294 }
2295 \cs_new_protected:Npn \__codedoc_key_pop:
2296 {
2297   \__kernel_tl_set:Ne \l__codedoc_index_key_tl
2298   { \tl_tail:N \l__codedoc_index_key_tl }
2299 }

```

(End of definition for `__codedoc_key_get:n`.)

`__codedoc_key_trim_module:n` Helper that removes from `\l__codedoc_index_module_tl` everything after the first occurrence of `#1`. `__codedoc_key_drop_underscores:` Helper that removes any leading underscore from `\l__codedoc_index_key_tl`.

```

2300 \cs_new_protected:Npn \__codedoc_key_trim_module:n #1
2301 {
2302   \cs_set:Npn \__codedoc_tmpa:w ##1 #1 ##2 \q_stop

```

```

2303     { \exp_not:n {##1} }
2304     \__kernel_tl_set:Ne \l__codedoc_index_module_tl
2305     { \exp_after:wN \__codedoc_tmpa:w \l__codedoc_index_module_tl #1 \q_stop }
2306   }
2307   \cs_new_protected:Npn \__codedoc_key_drop_underscores:
2308   {
2309     \tl_if_head_eq_charcode:VNT \l__codedoc_index_key_tl _
2310     { \__codedoc_key_pop: \__codedoc_key_drop_underscores: }
2311   }

```

(End of definition for `__codedoc_key_trim_module:n` and `__codedoc_key_drop_underscores:.`)

`__codedoc_key_func:` The function `__codedoc_key_func:` is used if there is a colon, so either for usual `expl3` functions or for keys from `l3keys`. After removing from the key a leading dot (for the latter case), and any leading underscore, the module name is the part before any colon or underscore.

```

2312   \cs_new_protected:Npn \__codedoc_key_func:
2313   {
2314     \tl_if_head_eq_charcode:VNT \l__codedoc_index_key_tl .
2315     { \__codedoc_key_pop: }
2316     \__codedoc_key_drop_underscores:
2317     \tl_set_eq:NN \l__codedoc_index_module_tl \l__codedoc_index_key_tl
2318     \exp_args:No \__codedoc_key_trim_module:n { \token_to_str:N : }
2319     \exp_args:No \__codedoc_key_trim_module:n { \token_to_str:N _ }
2320   }

```

(End of definition for `__codedoc_key_func:.`)

`__codedoc_key_var:` The function `__codedoc_key_var:` covers cases with no `:` but with `_`, typically variables but occasionally non-`expl3` functions such as Lua function with underscores. `__codedoc_key_get_module:` First test the second character: if that is `_` then assume we have a proper variable, otherwise use the part before any underscore as the module name. For variables, distinguish quarks and scan marks (starting with `q` and `s`), then drop the first letter (local/global/constant marker) and underscores to improve the index sorting. Then get the module as the first (underscore-delimited) “word”. In the past, we distinguished according to how many such words there were, to detect commands like `\c_zero`, which should be sorted as `int` variables, and `\l_tmpa_dim`, which should be sorted in the `dim` and not the `tmpa` module. Now the first case has been deprecated for some time, while `tmpa` and similar are special-cased through an explicit list given below. The way it works is that if the module is in a list of special names that

are not valid modules, then we try the last word, and if that also fails (for instance in the deprecated `\c_one_hundred`) we empty the module completely.

```

2321 \cs_new_protected:Npn \__codedoc_key_var:
2322 {
2323   \exp_args:Ne \tl_if_head_eq_charcode:nNTF
2324   { \exp_args:No \str_tail:n \l__codedoc_index_key_tl } _
2325   {
2326     \str_case:en { \str_head:N \l__codedoc_index_key_tl }
2327     {
2328       { q } { \tl_set:Nn \l__codedoc_index_module_tl { quark } }
2329       { s } { \tl_set:Nn \l__codedoc_index_module_tl { scan } }
2330     }
2331     \__codedoc_key_pop:
2332     \__codedoc_key_pop:
2333     \__codedoc_key_drop_underscores:
2334     \tl_if_empty:NT \l__codedoc_index_module_tl
2335     {
2336       \seq_set_split:NoV \l__codedoc_tmpa_seq
2337       { \token_to_str:N _ } \l__codedoc_index_key_tl
2338       \seq_get_left:NN \l__codedoc_tmpa_seq \l__codedoc_index_module_tl
2339       \clist_if_in:NoT \g__codedoc_non_modules_clist \l__codedoc_index_module_tl
2340       {
2341         \seq_get_right:NN \l__codedoc_tmpa_seq \l__codedoc_index_module_tl
2342         \clist_if_in:NoT \g__codedoc_non_modules_clist \l__codedoc_index_module_tl
2343         {
2344           \tl_clear:N \l__codedoc_index_module_tl
2345         }
2346       }
2347     }
2348   }
2349   {
2350     \tl_set_eq:NN \l__codedoc_index_module_tl \l__codedoc_index_key_tl
2351     \exp_args:No \__codedoc_key_trim_module:n { \token_to_str:N _ }
2352   }
2353 }

```

(End of definition for `__codedoc_key_var:` and `__codedoc_key_get_module:.`)

`\g__codedoc_non_modules_clist` List of names that appear as the first word in an `expl3` command, but that are not true modules, so that they should be sorted differently in an index.

```

2354 \clist_new:N \g__codedoc_non_modules_clist
2355 \clist_gset:Ne \g__codedoc_non_modules_clist

```

```

2356 {
2357   \tl_to_str:n
2358   {
2359
2360     alignment, ampersand, atsign, backslash, catcode, circumflex,
2361     code, colon, document, dollar, e, empty, false, hash, inf,
2362     initex, job, left, log, math, mark, max, minus, nan, nil, no,
2363     novalue, other, parameter, percent, pi, recursion, right, space,
2364     stop, term, tilde, tmpa, tmpb, true, underscore, zero, one, two,
2365     three, four, five, six, seven, eight, nine, ten, eleven, twelve,
2366     thirteen, fourteen, fifteen, sixteen, thirty, hundred
2367
2368   }
2369 }

```

(End of definition for `\g__codedoc_non_modules_clist`.)

5.15 Change history

Set the change history to use `\part`. Allow control names to be hyphenated in here...

```

2370 \GlossaryPrologue
2371 {
2372   \part*{Change~History}
2373   {\GlossaryParms\ttfamily\hyphenchar\font=~\~}
2374   \markboth{Change~History}{Change~History}
2375   \addcontentsline{toc}{part}{Change~History}
2376 }
2377 \msg_new:nnn { l3doc } { print-changes-howto }
2378 {
2379   Generate~the~change~list~by~executing\\
2380   \iow_indent:n
2381     { makeindex~-s~gglo.ist~-o~\c_sys_jobname_str.gls~\c_sys_jobname_str.glo }
2382 }
2383 \tl_gput_right:Nn \PrintChanges
2384 { \AtEndDocument { \msg_info:nn { l3doc } { print-changes-howto } } }

```

5.16 Default configuration

```

2385 \bool_if:NTF \g__codedoc_typeset_implementation_bool
2386 {
2387   \RecordChanges

```

```

2388     \CodelineIndex
2389     \EnableCrossrefs
2390     \AlsoImplementation
2391 }
2392 {
2393     \CodelineNumbered
2394     \DisableCrossrefs
2395     \OnlyDescription
2396 }
2397 </class>

```

5.17 Internal macros for L^AT_EX3 sources

These definitions are only used by the L^AT_EX3 documentation; they are not necessary for third-party users of l3doc. In time this will be broken into a separate package that is specifically loaded in the various expl3 modules, *etc.*

```

2398 <*cfg>

    The Guilty Parties.

2399 \tl_const:Nn \Team
2400 {
2401     The~\LaTeX3~Project\thanks
2402     {\url{https://www.latex-project.org/latex3/}}
2403 }

2404 \NewDocumentCommand{\ExplMakeTitle}{mm}
2405 {
2406     \title
2407     {
2408         The~\pkg{#1}~package \\\ #2
2409     }
2410     \author
2411     {
2412         The~\LaTeX3~Project\thanks{E-mail:~
2413         \href{mailto:latex-l@listserv.uni-heidelberg.de}
2414             {latex-l@listserv.uni-heidelberg.de}}
2415     }
2416     \date{Released~\ExplFileDate}
2417     \maketitle
2418 }

```

5.18 Math extras

For l3fp.

```
2419 \AtBeginDocument
2420 {
2421   \clist_map_inline:nn
2422     {
2423       asin, acos, atan, acot,
2424       asinh, acosh, atanh, acoth, round, floor, ceil
2425     }
2426     { \exp_args:Nc \DeclareMathOperator{#1}{#1} }
2427 }
```

`\nan`

```
2428 \NewDocumentCommand { \nan } { } { \text { \texttt { nan } } }
```

(End of definition for \nan. This function is documented on page ??.)

```
2429 </cfg>
```

5.19 Makeindex configuration

```
2430 <*docist>
```

The makeindex style `l3doc.ist` is used in place of the usual `gind.ist` to ensure that I is used in the sequence I J K not I II II, which would be the default makeindex behaviour.

Will: Do we need this?

Frank: at the moment we do not distribute or generate this file. `gind.ist` is used instead.

```
2431 actual '='
2432 quote '!'
2433 level '>'
2434 preamble
2435 "\n \\\begin{theindex} \n \\\makeatletter\scan@allowedfalse\n"
2436 postamble
2437 "\n\n \\\end{theindex}\n"
2438 item_x1 "\\efill \n \\\subitem "
2439 item_x2 "\\efill \n \\\subsubitem "
2440 delim_0 "\\pfill "
2441 delim_1 "\\pfill "
```

```

2442 delim_2    "\\pfill "
2443 % The next lines will produce some warnings when
2444 % running Makeindex as they try to cover two different
2445 % versions of the program:
2446 lethead_prefix "\\bfseries\\hfil "
2447 lethead_suffix "\\hfil}\\nopagebreak\n"
2448 lethead_flag    1
2449 heading_prefix "\\bfseries\\hfil "
2450 heading_suffix  "\\hfil}\\nopagebreak\n"
2451 headings_flag   1
2452
2453 % and just for source3:
2454 % Remove R so I is treated in sequence I J K not I II III
2455 page_precedence "rnaA"

(End of definition for .)

2456 </docist>

```

索引

斜体数字指向相应条目描述的页面，下划线数字指向定义的代码行，其它的都指向使用条目的页面。

Symbols		782, 783, 1890, 1891, 1892, 1893,
\"	486, 491	1894, 1895, 1896, 1897, 1898, 1899
\#	597	_ 432, 665, 1563, 2238
\%	1895	\ 487, 492
\&	1716, 1729	
\,	1194, 1200, 1575, 1577	A
\-	2373	\A 962
\/	724	\actualchar 2186, 2188, 2266
\:	433	\addcontentsline 2099, 2375
\<	1025, 1026, 1908	\addpenalty 469
\=	1731	\AddToHook 1678, 1679
\>	1726, 1909	\addtolength 455, 456, 457
\\	453, 1115, 1132, 1194, 1200, 1359, 1726, 1897, 2113, 2379, 2408	\addvspace 470
\{	531, 1898	\advance 476
\}	531, 1899	\allowbreak 1005
_	605, 610, 619, 628, 1437, 1882, 1896	\alphalph 1859
\^	135, 780,	\AlsoImplementation 6, <u>1734</u> , 2390
		\arabic 439, 597

<code>\Arg</code>	7, 32, 530	1259, 1263, 1264, 1265, 1271, 1272,	
<code>arguments (env.)</code>	10, 593	1277, 1284, 1288, 1478, 2081, 2279	
<code>\arrayrulecolor</code>	1216, 1228	<code>\c_false_bool</code> 216, 863, 866, 1398, 2128	
<code>\AtBeginDocument</code>	484, 2419	<code>\c_true_bool</code>	215, 1388
<code>\AtEndDocument</code>	489, 1995, 2065, 2118, 2384	<code>\bottomrule</code>	1101
<code>\author</code>	2410	box commands:	
B		<code>\box_clear:N</code>	1328
<code>\baselineskip</code> ...	608, 617, 626, 1336, 1377	<code>\box_dp:N</code>	1004
<code>\begin</code> ..	1096, 1217, 1219, 1232, 1613, 1623	<code>\box_gclear:N</code>	1012
<code>\begingroup</code>	472, 1614, 1624, 1683, 1700, 1724	<code>\box_if_empty:N</code>	1009, 1206
<code>\bfseries</code>	475, 1869	<code>\box_new:N</code>	14, 61, 62
<code>\bigskip</code>	999, 1694, 1719, 1733	<code>\box_use_drop:N</code>	1054, 1234
bool commands:		<code>\box_wd:N</code>	1040, 1141
<code>\bool_gset_false:N</code>	371, 544, 546, 1735	<code>\l_tmpa_box</code>	1138, 1141, 1146, 1152
<code>\bool_gset_true:N</code> 41, 365, 366, 370, 540, 542, 1737	<code>\bslash</code>	2125, 2260, 2266
<code>\bool_if:N</code> ...	131, 169, 261, 278, 419, 549, 553, 556, 560, 563, 568, 648, 657, 662, 674, 676, 773, 805, 816, 859, 865, 879, 907, 1055, 1108, 1111, 1127, 1128, 1148, 1154, 1161, 1173, 1178, 1211, 1230, 1405, 1407, 1413, 1448, 1471, 1549, 1550, 1598, 1630, 1640, 1740, 2012, 2163, 2385	C	
<code>\bool_lazy_all:nTF</code>	1537	<code>\c</code>	666
<code>\bool_lazy_and:nnTF</code>	1187, 1947	<code>\catcode</code>	1359
<code>\bool_lazy_any:nTF</code>	1415	<code>\changes</code>	60
<code>\bool_lazy_any_p:n</code>	1586	<code>\char</code>	531, 1610
<code>\bool_lazy_or:nnTF</code>	264	char commands:	
<code>\bool_new:N</code>	6, 15, 16, 19, 23, 24, 25, 26, 27, 28, 29, 30, 31, 35, 36, 37, 38, 39, 40, 48, 55, 67, 68, 69, 70, 71, 77, 2080	<code>\char_generate:nn</code>	2198
<code>\bool_set:Nn</code>	1041	<code>\char_set_active_eq:NN</code> ...	1025, 2197
<code>\bool_set_false:N</code> ..	550, 642, 922, 927, 936, 1015, 1016, 1017, 1018, 1019, 1020, 1021, 1266, 1278, 1283, 1315, 1316, 1317, 1318, 1319, 1320, 1321, 1322, 1323, 1324, 2087, 2280	<code>\char_set_catcode:nn</code>	706
<code>\bool_set_true:N</code> ..	7, 72, 73, 557, 643, 916, 921, 928, 933, 934, 935, 941, 942, 952, 1022, 1249, 1252, 1256,	<code>\char_set_catcode_active:N</code> 135, 1026, 1891, 1892, 1908, 1909
		<code>\char_set_catcode_letter:N</code> 431, 432, 433
		<code>\char_set_catcode_other:N</code> 1890, 1893, 1894
		<code>\char_set_lccode:nn</code> 1895, 1896, 1897, 1898, 1899
		<code>check (option)</code>	5
		<code>checktest (option)</code>	5
		<code>\ClassError</code>	2060
		<code>\clearpage</code>	1773, 1784, 1818
		clist commands:	
		<code>\clist_clear:N</code>	1023
		<code>\clist_count:N</code>	14
		<code>\clist_count:n</code>	14
		<code>\clist_gset:Nn</code>	2355
		<code>\clist_if_in:NnTF</code>	2339, 2342

\clist_map_function:NN	1770	__codedoc_date_compare:nNn	293
\clist_map_function:nN	2021, 2040	__codedoc_date_compare:nNnTF	293, 971
\clist_map_inline:Nn	1163, 1790	__codedoc_date_compare_aux:nnnNnnn	293, 298, 307
\clist_map_inline:nn	1763, 2421	__codedoc_date_compare_aux:w	293, 294, 295
\clist_new:N	3, 76, 2354	__codedoc_date_compare_p:nNn	293
\clist_put_right:Nn	1765	__codedoc_date_set:Nn	958, 958, 970
\clist_set:Nn	951	__codedoc_date_set_past:Nn	46, 944, 945, 958, 968
\closeout	1820	\l__codedoc_date_updated_tl	78, 945, 1189, 1197, 1200
\cls	8, 535	\l__codedoc_descr_coffin	11, 1004, 1011, 1046, 1067, 1070, 1080
\cmd	5, 7, 16, 36, 80, 505, 525	__codedoc_detect_internals:N	129, 129, 784, 1917, 1926
codedoc internal commands:		__codedoc_detect_internals_-	aux:N 129, 132, 136
\c__codedoc_active_minus_tl	2196	\l__codedoc_detect_internals_-	bool 6, 131
\l__codedoc_allow_indexing_bool	657, 674, 2080, 2081, 2087	\l__codedoc_detect_internals_cs_-	tl 9, 149, 154
__codedoc_base_form_aux:nnN	234, 258, 286, 811	\l__codedoc_detect_internals_tl	6, 138, 139, 140, 141, 142, 144, 146, 147, 149, 150, 151, 152, 155
__codedoc_base_form_aux:nnnnN	274, 276	\l__codedoc_doc_def_tl	1940, 1953, 1962, 1983
__codedoc_base_form_signature_-	do:nnn 271, 271	\l__codedoc_doc_undef_tl	1941, 1954, 1966, 1986
\g__codedoc_base_name_tl	74, 841, 842, 848, 854	__codedoc_ensuremath_sb:n	38, 699, 708, 712
\g__codedoc_checkfunc_bool	35, 373, 1948	\g__codedoc_finale_tl	1734
\g__codedoc_checktest_bool	35, 374, 1539, 2012	__codedoc_fn_footnote:nn	1671, 1678, 1679
__codedoc_cmd:nn	31, 506, 508, 511, 640, 640, 698	__codedoc_fn_restore:n	1662, 1676
\l__codedoc_cmd_index_tl	64, 635, 644, 678, 681	__codedoc_fn_store:	1657, 1674
\l__codedoc_cmd_module_tl	64, 636, 645, 684, 687	\g__codedoc_fnmark_prop	1656, 1659, 1664
\l__codedoc_cmd_noindex_bool	36, 64, 637, 642, 676	\g__codedoc_func_iow	1939, 1951, 1991, 2000
\l__codedoc_cmd_replace_bool	64, 638, 643, 648		
\l__codedoc_cmd_tl	64, 647, 651, 652, 653, 659, 661, 667, 670, 671, 680, 683		
\g__codedoc_cs_break_bool	35, 378, 662		
\l__codedoc_date_added_tl	78, 944, 1188, 1191, 1194		

__codedoc_function:nnw	__codedoc_hdindex_aux:nn
..... 49 , 565 , 573 , 979 , 979 2196 , 2214 , 2217
__codedoc_function_assemble: ..	__codedoc_hdindex_aux:w
..... 994 , 1051 , 1051 2196 , 2225 , 2229
__codedoc_function_descr_-	__codedoc_if_almost_str:n .. 86 , 94
start:w	__codedoc_if_almost_str:nTF ...
989 , 1044 , 1044 86 , 230 , 659 , 2233
__codedoc_function_descr_stop:	__codedoc_if_detect_internals_-
..... 993 , 1044 , 1049	ok:N
__codedoc_function_end: 161
..... 49 , 570 , 574 , 979 , 991	__codedoc_if_detect_internals_-
__codedoc_function_extra_-	ok:NTF
labels: 129 , 147
1098 , 1159	__codedoc_if_macro_internal:n . 877
__codedoc_function_index:n	__codedoc_if_macro_internal:nTF
..... 1105 , 1107 , 1118 , 1123 877 , 1466
__codedoc_function_init:	__codedoc_if_macro_internal_-
..... 982 , 1007 , 1007	aux:w
__codedoc_function_label:nN 877 , 884 , 891
..... 1109 , 1171 , 1184	__codedoc_if_macro_internal_p:n
\l__codedoc_function_label_clist 877 , 1417 , 1545 , 1588
..... 76 , 951 , 1023 , 1163	\l__codedoc_in_function_bool ...
__codedoc_function_reset: 15 , 1022 , 1230
..... 988 , 1031 , 1031	\l__codedoc_in_implementation_-
__codedoc_function_typeset: ...	bool 33 , 69 , 550 , 557 , 563 , 568
..... 987 , 1035 , 1035	__codedoc_index_codeline_hc:nn
__codedoc_function_typeset_- 1849 , 2175 , 2179
start:	\l__codedoc_index_escaped_key_tl
981 , 997 , 997 2145 , 2171 , 2172 , 2187
__codedoc_function_typeset_-	\l__codedoc_index_escaped_macro_-
stop:	tl
995 , 997 , 1001 2144 , 2173 , 2191
\l__codedoc_functions_coffin ...	\l__codedoc_index_internal_bool
..... 13 , 47 , 15 , 51 , 694 , 2141 , 2279 , 2280
49 , 51 , 11 , 1014 , 1038 , 1040 , 1063 , 1084	\l__codedoc_index_key_tl
__codedoc_functions_typeout:nN 15 , 86 , 87 , 51 ,
..... 1978 , 1984 , 1987 , 1996	690 , 1428 , 2137 , 2275 , 2278 , 2282 ,
__codedoc_get_function_name:n .	2284 , 2287 , 2290 , 2297 , 2298 , 2309 ,
..... 202 , 206 , 206 , 1133 , 1392	2314 , 2317 , 2324 , 2326 , 2337 , 2350
__codedoc_get_function_signature:n	\l__codedoc_index_macro_tl
..... 204 , 206 , 208 15 , 86 , 51 , 691 , 2138 , 2274 , 2276
__codedoc_get_hyper_target:nN .	\l__codedoc_index_module_tl
762 , 762 , 769 , 1165 , 1175 , 1445 , 1601 15 , 86 , 87 , 51 ,
__codedoc_gprop_name:n ... 328 , 328	686 , 692 , 2135 , 2139 , 2277 , 2291 ,
__codedoc_hdindex:nn	2304 , 2305 , 2317 , 2328 , 2329 , 2334 ,
..... 2196 , 2203 , 2205 , 2206	2338 , 2339 , 2341 , 2342 , 2344 , 2350

__codedoc_index_page_hc:nn	\l__codedoc_macro_do_not_index_-
1841, 2180, 2182	tl 15, 51 , 1296,
__codedoc_input:n	1356, 1358, 1360, 1362, 1363, 1365
1754, 1754, 1766, 1770	\l__codedoc_macro_documented_tl
\c__codedoc_iow_mid_rule_tl 59	23, 1294, 1326, 1345, 1349
\c__codedoc_iow_midrule_tl	__codedoc_macro_dump:
57, 1944, 2002, 2004	1309, 1368 , 1368
\c__codedoc_iow_rule_tl 57	__codedoc_macro_end:
\c__codedoc_iow_separator_tl . . .	569, 577, 1528 , 1528
1943, 1950, 1992	__codedoc_macro_end_check_-
\g__codedoc_kernel_bool	tested: 1531, 1535 , 1535
35, 169, 375, 376	__codedoc_macro_end_style:n . . .
__codedoc_key_drop_underscores:	1533, 1554 , 1554
2300, 2307, 2310, 2316, 2333	__codedoc_macro_end_wrap_item:n
__codedoc_key_func:	1559, 1559, 1570
87, 88, 2285, 2312 , 2312	__codedoc_macro_exclude_index:
__codedoc_key_get:n	1306, 1354 , 1354
683, 2133, 2272 , 2272	\l__codedoc_macro_EXP_bool
__codedoc_key_get_base:nN	23, 921, 927,
86, 228 , 228, 2274	935, 1018, 1127, 1265, 1277, 1283, 1320
__codedoc_key_get_base_TF:nN . .	__codedoc_macro_index:nN 1406, 1410
232, 238	\l__codedoc_macro_index_box
__codedoc_key_get_module: . . . 2321	61, 1328, 1376, 1422, 1423
__codedoc_key_pop:	__codedoc_macro_init:
.. 2283, 2295, 2310, 2315, 2331, 2332	1301, 1312 , 1312
__codedoc_key_trim_module:n . . .	\l__codedoc_macro_int
18, 2300 , 2300, 2318, 2319, 2351	61, 1334, 1336, 1440
__codedoc_key_var:	\l__codedoc_macro_internal_bool
87, 88, 2288, 2321 , 2321	44, 23, 879, 1249, 1252, 1316
\g__codedoc_lmodern_bool 35 , 377, 419	\l__codedoc_macro_nodoc_bool . . .
\l__codedoc_long_name_bool	23, 947, 1253, 1419, 1590
13, 49 , 16 , 1041, 1055, 1211	\l__codedoc_macro_noTF_bool
__codedoc_lseq_name:n	23, 865, 941, 1017, 1272, 1319
14, 41 , 328 , 329, 822	\l__codedoc_macro_pTF_bool
__codedoc_macro:nnw	23, 859, 933, 1016, 1264, 1318, 1549
564, 576, 1299 , 1299	__codedoc_macro_reset:
\l__codedoc_macro_argument_tl . .	1310, 1339 , 1339
80, 975, 983, 1302	\l__codedoc_macro_rEXP_bool
\l__codedoc_macro_box	23, 922, 928,
61, 61 , 1329, 1379, 1433, 1435	936, 1019, 1128, 1266, 1278, 1284, 1321
\l__codedoc_macro_deprecated_-	__codedoc_macro_save_names: . . .
bool 23 , 946, 1246, 1315, 1418, 1589	1307, 1343 , 1343

__codedoc_macro_save_names_-	__codedoc_names_parse_one_-
aux:n 1346, 1352	aux:nnNn 800, 803
__codedoc_macro_single:nNN	\l__codedoc_names_seq
..... 39, 56, 1388, 1398, 1401, 1401 49, 794, 985, 1304, 1517, 1544
\l__codedoc_macro_tested_bool ..	__codedoc_names_typeset:
..... 19, 1288, 1323, 1478, 1541 833, 833, 1099, 1308
\l__codedoc_macro_TF_bool	__codedoc_names_typeset_auxi:n
..... 23, 816, 867, 907, 916, 934, 42, 833, 836, 838
942, 1015, 1259, 1263, 1271, 1317, 1550	__codedoc_names_typeset_auxii:n
__codedoc_macro_typeset_- 42, 848, 853, 857, 857, 869
block:nN 874, 1386, 1386	__codedoc_names_typeset_-
__codedoc_macro_typeset_one:nN	block:nN 861, 866, 867, 870, 870, 876
..... 1404, 1431, 1431	\l__codedoc_names_verb_bool
__codedoc_macro_typeset_- 48, 773, 955, 1021, 1292, 1324
variant_list:nN .. 1391, 1395, 1400	\l__codedoc_nested_macro_int ...
\l__codedoc_macro_var_bool 43, 57, 18, 872, 1314, 1532
..... 23, 1256, 1322, 1540, 1598	\g__codedoc_nested_names_seq ...
__codedoc_macroname_prefix:n 50, 1348, 1353, 1569,
..... 1460, 1464, 1469	1583, 1584, 1593, 1596, 1602, 1606
__codedoc_macroname_suffix:N ..	\l__codedoc_no_label_bool
..... 1461, 1470 76, 952, 1020, 1161, 1173
__codedoc_meta:n 516, 699, 699	\g__codedoc_non_modules_clist ..
__codedoc_meta_original:n 2339, 2342, 2354
..... 699, 710, 714	\g__codedoc_not_tested_seq
\g__codedoc_missing_tests_prop 19, 1546, 2037, 2053
..... 63, 19, 1516, 2015, 2034	__codedoc_old_hdclindex:nnn ...
\g__codedoc_module_name_tl .. 56, 2196, 2201, 2205
113, 116, 156, 163, 166, 1756, 1933	__codedoc_old_hdpindex:nn
__codedoc_names_block_base_- 2196, 2200, 2203
map:N 892, 892, 1346	__codedoc_oldlist:nn .. 458, 458, 460
\l__codedoc_names_block_tl 41, 42,	\l__codedoc_output_coffin
44, 49, 58, 46, 792, 825, 827, 835, 894 10, 1058, 1062,
__codedoc_names_get_seq:nN	1066, 1069, 1075, 1079, 1083, 1086
..... 770, 770, 985, 1304	\l__codedoc_override_module_tl ..
__codedoc_names_parse: 13, 23, 956, 1024,
..... 790, 790, 986, 1305	1033, 1293, 1325, 1341, 2134, 2135
__codedoc_names_parse_aux:Nnn ..	__codedoc_pdfstring_cmd:w
..... 822, 823 517, 518, 525
__codedoc_names_parse_aux:nnn ..	__codedoc_pdfstring_cs:w
..... 808, 810, 818, 821 517, 520, 526, 527
__codedoc_names_parse_one:n ...	__codedoc_pdfstring_meta:w
..... 790, 795, 797 517, 521, 528

__codedoc_pdfstring_newline:w 689, 2123, 2136, 2144 , 2146, 2168
..... 449 , 451, 453	
__codedoc_predicate_from_base:n 2144 , 2173, 2230
..... 200, 200, 862	
__codedoc_print_documented: 206 , 214, 219
..... 1559 , 1579, 1581	
__codedoc_print_end_definition:	auxii:w 206 , 221, 222
..... 1533 , 1559 , 1566	
__codedoc_print_macroname:nN ..	__codedoc_split_function_do:nn
..... 1436 , 1442 , 1442	206 , 207, 209, 212, 227, 234, 273, 799
__codedoc_print_testfile:n	__codedoc_str_if_begin:nn 101, 109
..... 1327 , 1476 , 1476	__codedoc_str_if_begin:nnTF ...
__codedoc_print_testfile_aux:n 101 , 163, 166, 171
..... 1476 , 1485, 1489	__codedoc_syntax:w .. 579 , 1203 , 1204
__codedoc_quote_special_char:N	\g__codedoc_syntax_box 49 ,
..... 2172 , 2262, 2264 , 2264	14 , 1009, 1012, 1054, 1206, 1214, 1234
__codedoc_replace_at_at:N	\l__codedoc_syntax_coffin
... 36 , 111 , 111, 652, 785, 1918, 1927	. 13 , 11 , 1013, 1053, 1059, 1076, 1087
__codedoc_replace_at_at_aux:Nn	\l__codedoc_syntax_dim
..... 111 , 115, 119 1203 , 1208, 1219
__codedoc_shorthand_meta:	__codedoc_syntax_end: 581 , 1203 , 1224
..... 1025 , 1028 , 1028	__codedoc_target:
__codedoc_shorthand_meta:w 657 , 1094, 1334, 2066, 2182
..... 1028 , 1029, 1030	__codedoc_test_missing:n
__codedoc_show_functions_- 1512 , 1513 , 1513
defined: 1945, 1995	__codedoc_test_missing_aux:Nnn
__codedoc_show_not_tested: 1515 , 1520, 1527
..... 2010, 2065	\g__codedoc_testfiles_seq
\g__codedoc_show_notes_bool 19 , 1481, 1483
..... 35 , 380, 381, 1630, 1640	__codedoc_tmp:w 97, 98
__codedoc_signature_base_form:n	__codedoc_tmpa:w ... 82 , 82, 210,
..... 179 , 179, 268	226 , 2017, 2034, 2039, 2053, 2302, 2305
__codedoc_signature_base_form_-	\l__codedoc_tmpa_int
aux:n 179 , 180, 181, 195 42 , 1595, 1597, 1599, 2058, 2061
__codedoc_signature_base_form_-	\l__codedoc_tmpa_seq 42 , 140,
aux:w 179 , 196, 198	141 , 142, 897, 898, 899, 1362, 1364,
__codedoc_special_index:nn	1544 , 1548, 1568, 1572, 1574, 1577,
.. 1121 , 1425 , 2108, 2131 , 2131, 2143	2212 , 2213 , 2215 , 2336 , 2338 , 2341
__codedoc_special_index_-	\l__codedoc_tmpa_tl
aux:nnnnn 2144 , 2150, 2169 38 , 40, 44, 42 , 232,
__codedoc_special_index_-	234 , 701, 707, 710, 772, 776, 779,
module:nnnnN	781 , 782, 783, 784, 785, 787, 840,
	841 , 844, 845, 898, 904, 1166, 1167,

1180, 1181, 1450, 1451, 1452, 1457, 1458, 1460, 1522, 1523, 1524, 1525, 1561, 1562, 1564, 1603, 1604, 1664, 1666, 1667, 1912, 1916, 1917, 1918, 1919, 1928, 1931, 1934, 2014, 2028, 2046, 2056, 2060, 2208, 2209, 2212, 2219, 2220, 2221, 2223, 2225, 2227	\g__codedoc_variants_seq 42, 47, 847, 851, 852, 1113, 1135, 1136, 1139, 1389, 1397
__codedoc_tmpb:w	__codedoc_xmacro_code:n 75, 1889, 1905, 1910
. 82, 83, 2021, 2023, 2040, 2041	__codedoc_xmacro_code:w 1889, 1914, 1922, 1936
\l__codedoc_tmpb_tl 44, 42, 650, 651, 653, 705, 707, 852, 854, 899, 905, 906, 1925, 1926, 1927, 1928	\CodedocExplain 604
\l__codedoc_trial_width_dim 13, 16, 1037, 1039, 1042, 1212	\CodedocExplainEXP 604
__codedoc_trim_right:Nn 95, 95, 100, 144, 146, 150, 151, 152	\CodedocExplainREXP 604
__codedoc_typeset_aux:n 729, 758, 1133, 1467	\CodedocExplainTF 604
__codedoc_typeset_dates: 1100, 1185, 1185	\CodelineIndex 2388
\g__codedoc_typeset_documentation_- bool 70, 365, 370, 540, 544, 549, 553	CodelineNo 435
__codedoc_typeset_exp: 610, 729, 729, 1127	\CodelineNumbered 2393
__codedoc_typeset_expandability: 1112, 1124, 1157	coffin commands:
__codedoc_typeset_function_- block:nN 873, 1105, 1105, 1117	\coffin_clear:N 1011, 1013, 1014
__codedoc_typeset_functions: 49, 1038, 1091, 1091	\coffin_join:NnnNnnnn 1057, 1061, 1065, 1074, 1078, 1082
\g__codedoc_typeset_implementation_- bool 70, 366, 371, 542, 546, 556, 560, 1735, 1737, 1740, 1948, 2385	\coffin_new:N 10, 11, 12, 13
__codedoc_typeset_rexp: 619, 729, 736, 1128	\coffin_typeset:Nnnnn 1069, 1086
__codedoc_typeset_TF: 628, 729, 743, 1111, 1148, 1154, 1471	\color 749, 753, 760
__codedoc_typeset_variant_- list:nN 1114, 1130	\columnwidth 1141, 1143
\l__codedoc_undoc_def_tl 1942, 1955, 1974, 1989	\comment 33, 551, 558, 1653
\l__codedoc_variants_prop 74	\ConTeXt 504
	\cs 5, 7, 12, 16, 36, 57, 80, 505, 526
	cs commands:
	\cs_generate_variant:Nn 84, 85, 100, 227, 698, 769, 869, 876, 1117, 1123, 1184, 1400, 1469, 1527, 1670, 2143, 2168
	\cs_gset:Npe 1810
	\cs_gset:Npn 429, 459, 466, 482, 1610, 1782, 1857, 1864, 1875, 2093
	\cs_gset_eq:NN 441, 442, 1874
	\cs_gset_protected:Npn 1826, 1832, 2105, 2119, 2202, 2204
	\cs_if_exist:NTF 393, 731, 738, 745, 753, 1451, 1749, 1877
	\cs_new:Npe 238
	\cs_new:Npn 179, 181, 198, 200, 206, 208, 212, 219, 222, 258, 295, 307, 328, 329, 449, 521, 891, 1185

<code>\cs_new_eq:NN</code> .	82, 83, 458, 2200, 2201	<code>\DeclareDocumentCommand</code>	505,
<code>\cs_new_protected:Npe</code>	119, 2272		507, 509, 515, 1500, 1510, 1511,
<code>\cs_new_protected:Npn</code>			1734, 1736, 1738, 1751, 1761, 1769
.	95, 111, 129, 136, 228,	<code>\DeclareDocumentEnvironment</code>	561, 572, 575
	271, 276, 640, 699, 712, 714, 729,	<code>\DeclareExpandableDocumentCommand</code> .	
	736, 743, 758, 762, 770, 790, 797,	450, 517, 519
	803, 821, 823, 833, 838, 857, 870,	<code>\DeclareFontShape</code>	425
	892, 958, 968, 979, 991, 997, 1001,	<code>\DeclareKeys</code>	360
	1007, 1028, 1031, 1035, 1044, 1049,	<code>\DeclareMathOperator</code>	2426
	1051, 1091, 1105, 1118, 1124, 1130,	<code>\DeclareRobustCommand</code> .	535, 536, 537, 538
	1159, 1171, 1204, 1224, 1299, 1312,	<code>\DeclareUnknownKeyHandler</code>	383
	1339, 1343, 1352, 1354, 1368, 1386,	<code>\def</code>	609, 618, 627, 1678, 1679,
	1395, 1401, 1410, 1431, 1442, 1464,		1686, 1687, 1703, 1713, 1726, 1727
	1470, 1476, 1489, 1513, 1520, 1528,	<code>\DeleteShortVerb</code>	491, 492
	1535, 1554, 1559, 1566, 1581, 1657,	<code>\Describe</code>	3
	1662, 1671, 1754, 1841, 1849, 1910,	<code>\DescribeOption</code>	1608
	1922, 1945, 1996, 2010, 2066, 2131,	<code>\DescribeRoutine</code>	3
	2146, 2169, 2206, 2217, 2229, 2230,	<code>\DescribeVariable</code>	3
	2264, 2295, 2300, 2307, 2312, 2321	<code>\description</code>	1685, 1702
<code>\cs_new_protected_nopar:Npn</code> . . .	1030	<code>\detokenize</code>	1634
<code>\cs_set:Npe</code>	1789	dim commands:	
<code>\cs_set:Npn</code>		<code>\dim_compare:nNnTF</code>	1141
.	97, 1372, 1615, 1625, 1776,	<code>\dim_compare_p:nNn</code>	1042
	1777, 2017, 2023, 2039, 2041, 2302	<code>\dim_new:N</code>	17, 1203
<code>\cs_set_eq:NN</code> . .	453, 525, 526, 527,	<code>\dim_set:Nn</code>	1004, 1039, 1208
	528, 1327, 1798, 1801, 1802, 1803,	<code>\dim_zero:N</code>	460, 1037
	1804, 1805, 1806, 1809, 1814, 1815,	<code>\ding</code>	741
	1816, 1817, 1824, 1861, 1872, 1887	<code>\DisableCrossrefs</code>	2394
<code>\cs_set_nopar:Npe</code>	2073	<code>\DisableDocumentation</code>	6, 539
<code>\cs_set_nopar:Npn</code>	2084	<code>\DisableImplementation</code>	6, 539
<code>\cs_set_protected:Npn</code> .	210, 443, 1903	doc commands:	
<code>cs-break</code> (option)	5	<code>\g_doc_functions_seq</code>	
<code>cs-break-nohyphen</code> (option)	5	4 , 1120, 1956, 1958, 1972
<code>\currentfile</code>	1776, 1777,	<code>\g_doc_macros_seq</code>	
	1807, 1811, 1813, 1877, 1878, 1880	4 , 1421, 1957, 1960, 1970
<code>\CurrentOption</code>	384	<code>\DocInclude</code>	1771
D			
<code>\d</code>	962	<code>\docincludeaux</code>	1774 , 1857
<code>danger</code> (env.)	1609	<code>\DocInput</code>	6, 12 , 1761 , 1813
<code>\date</code>	2416	docinput commands:	
<code>\dbend</code>	1610, 1616, 1626	<code>\g_docinput_clist</code> . .	72 , 3 , 1765, 1770
<code>ddanger</code> (env.)	1609	<code>\DocInputAgain</code>	6, 1769
		documentation (env.)	547

<code>\DoNotIndex</code>	58, 1365, 1405, 1412	<code>\evensidemargin</code>	457
E			
<code>\edef</code>	720	<code>\exp_after:wN</code>	
<code>\else</code>	1706, 1821	98, 104, 105, 214, 884, 2305
<code>\emph</code>	1557, 1634, 1644, 1649	<code>\exp_args</code>	265
<code>\EnableCrossrefs</code>	2389	<code>\exp_args:Nc</code>	822, 2426
<code>\EnableDocumentation</code>	6, 539	<code>\exp_args:Ne</code> 810, 1406, 1604, 2241, 2323	
<code>\EnableImplementation</code>	6, 539	<code>\exp_args:Nee</code>	445
<code>\encapchar</code>	1845, 1853, 2266	<code>\exp_args:NNe</code>	786, 900
<code>\end</code> 1102, 1226, 1227, 1235, 1619, 1628, 1905		<code>\exp_args:NNNo</code>	1427
<code>\endcomment</code>	33, 553, 560, 1653	<code>\exp_args:NNo</code>	115
<code>\enddescription</code>	1689, 1715	<code>\exp_args:NNV</code>	1361
<code>\endenumerate</code>	602	<code>\exp_args:No</code>	226, 683,
<code>\endgraf</code>	586, 1615, 1625	971, 1167, 1181, 2318, 2319, 2324, 2351	
<code>\endgroup</code>		<code>\exp_args:NV</code>	447, 710, 1365
479, 1615, 1625, 1689, 1715, 1727, 1830		<code>\exp_last_unbraced</code>	266
<code>\endinput</code>	1747	<code>\exp_last_unbraced:NNNo</code>	890
<code>\endtabbing</code>	1727	<code>\exp_last_unbraced:NNo</code>	1452
<code>\endtheglossary</code>	1806, 1817	<code>\exp_last_unbraced:NV</code>	1666, 1667
<code>\endtrivlist</code>	1530	<code>\exp_not:N</code>	
<code>\endVerbatim</code>	442	240, 242, 244, 248, 254, 256,
<code>\endverbatim</code>	440 , 1648	1676, 2076, 2274, 2275, 2276, 2277,	
<code>\enquote</code>	631	2278, 2279, 2280, 2281, 2282, 2283,	
<code>\ensuremath</code>	713, 716, 727	2284, 2285, 2287, 2288, 2290, 2291	
<code>\enumerate</code>	595	<code>\exp_not:n</code>	
<code>\env</code>	8, 535	..	21, 97, 199, 260, 267, 285, 681,
environments:		1177, 1447, 1676, 2020, 2077, 2084,	
arguments	10, 593	2088, 2150, 2210, 2224, 2227, 2303	
danger	1609	<code>\ExplFileDate</code>	1882, 2416
ddanger	1609	<code>\ExplFileVersion</code>	1883
documentation	547	<code>\ExplMakeTitle</code>	27, 2404
function	8, 572	F	
implementation	547	<code>\fi</code>	480, 1710, 1796, 1823
macro	9, 572	fi commands:	
NOTE	1640	<code>\fi:</code>	1794
syntax	8, 578	<code>\file</code>	8, 535
texnote	9, 584	file commands:	
variable	8, 10, 561	<code>\g_file_curr_name_str</code>	1518
<code>\epTeX</code>	494	<code>\file_if_exist:nTF</code>	390
<code>\errorstopmode</code>	2059	<code>\file_input:n</code>	392
<code>\eTeX</code>	494	<code>\filekey</code>	1810, 1811, 1861, 1869
<code>\eupTeX</code>	494	<code>\filename</code>	1881

<code>\filesep</code>	1837, 1840, 1854, 1860	<code>hide-notes</code> (option)	5
<code>\Finale</code>	1734	<code>\hologo</code>	494, 495, 497, 498, 499, 500, 501, 504
<code>\font</code>	721, 722, 1609, 2373	hook commands:	
<code>\fontfamily</code>	1474	<code>\hook_gput_next_code:nn</code> ..	1670, 1675
<code>\fontseries</code>	1474	<code>\href</code>	2413
<code>\foo</code>	7	<code>\hskip</code>	477, 1616, 1626, 1707, 1709
<code>\footnote</code>	1634, 1678, 1679	<code>\hspace</code>	1096
<code>\footnotemark</code>	1673	<code>\hss</code>	478
<code>\footnotesize</code>	1491, 1557, 1868	<code>\hyperlink</code>	732, 739, 746
<code>\footnotetext</code>	1676	<code>\hyperref</code>	1452
full (option)	5	<code>\hypertarget</code>	608, 617, 626
function (env.)	8, 572	<code>\hyphenchar</code>	721, 722, 2373
<code>\fvset</code>	440		

G

<code>\GetFileInfo</code>	1880
<code>\GlossaryParms</code>	2373
<code>\GlossaryPrologue</code>	2370
group commands:	
<code>\group_begin:</code>	134, 423, 896, 1889, 1907, 2069, 2075, 2086, 2196
<code>\group_end:</code>	160, 426, 901, 1902, 1938, 2071, 2078, 2089, 2199

H

<code>\hangafter</code>	1614, 1624
<code>\hangindent</code>	1614, 1616, 1624, 1626
<code>\hbox</code>	1436, 1616, 1626, 1708
hbox commands:	
<code>\hbox:n</code>	1331
<code>\hbox_gset:Nw</code>	1214
<code>\hbox_gset_end:</code>	1229
<code>\hbox_set:Nn</code>	1138
<code>\hbox_set:Nw</code>	1422
<code>\hbox_set_end:</code>	1427
<code>\hbox_unpack_drop:N</code>	1146, 1152, 1376, 1423
hcoffin commands:	
<code>\hcoffin_set:Nn</code>	1038, 1053
<code>\hdclindex</code>	2196
<code>\hdpindex</code>	2196
<code>\hfil</code>	478
<code>\hfill</code>	1616, 1626, 1708, 1885

I

if commands:	
<code>\if_meaning:w</code>	1792
<code>\IfFileExists</code>	1775
<code>\ifnum</code>	468
<code>\ifx</code>	1706
<code>\ignorespaces</code>	1047, 1508, 1616, 1626
<code>\ignorespacesafterend</code>	582
<code>\immediate</code>	1785, 1799, 1800, 1820, 1834, 1851
implementation (env.)	547
<code>\include</code>	1779
<code>\indexentry</code>	1829, 1836, 1845, 1853
<code>\IndexPrologue</code>	2095
<code>\IniTeX</code>	494
<code>\input</code>	1758
<code>\InstanceKey</code>	1726
<code>\InstanceSemantics</code>	1727
int commands:	
<code>\int_compare:nNnTF</code>	88, 309, 311, 313, 317, 322, 872, 1334, 1532, 1597, 1599, 1778
<code>\int_compare:nTF</code> ...	1135, 1455, 1572
<code>\int_eval:n</code>	1336
<code>\int_gdecr:N</code>	1426
<code>\int_gincr:N</code>	1424
<code>\int_incr:N</code>	1314, 1440
<code>\int_new:N</code>	18, 44, 45, 63, 81
<code>\int_set:Nn</code>	1595, 2058, 2061
<code>\int_use:N</code>	1095, 1660, 1676, 1837, 1854

iow commands:		M
\iow_char:N	780, 782, 783	macro (env.) 9, 572
\iow_close:N	1991	\MacroFont 1437, 1456
\iow_indent:n	2114, 2380	\MacroLongFont 1456, 1472
\iow_new:N	1939	\MacroTopsep 1370
\iow_newline: . . .	1950, 1963, 1967, 1975, 1980, 2002, 2003, 2019, 2025, 2030, 2032, 2043, 2048, 2049, 2051	\makebox 751
\iow_now:Nn	2000	\makelabel 1372
\iow_open:Nn	1951	\MakePercentComment 1759
\iow_term:n	78, 1950, 1992	\MakePercentIgnore 1757
\item	1384, 1613, 1623, 1686, 1705	\MakePrivateLetters 429, 1359, 2092
\itemindent	462	\MakeShortVerb 486, 487
\itshape	750, 754	\maketitle 2417
		\manual 1609, 1610
K		\marg 7, 32, 530
\kern	500, 501, 754, 1626	\marginparsep 1064, 1068, 1085
kernel (option)	5	\marginparwidth 455, 1042, 1068, 1212
kernel internal commands:		\markboth 2098, 2374
__kernel_tl_set:Nn		\MaybeStop 71, 1734
. 98, 233, 240, 661, 680, 686, 764, 772, 1363, 2232, 2250, 2275, 2297, 2304		\mbox 655, 733, 740, 747
keys commands:		\medskipamount 1068, 1081
\keys_define:nn	633, 911, 1238	\meta 7, 515, 528, 531, 533, 534, 1030
\keys_set:nn	646, 984, 1303	\midrule 1190
\kill	1731	mode commands:
		\mode_if_math:TF 655, 717, 1029
L		\mode_leave_vertical: 1233, 2068
\label	1167, 1181	msg commands:
\langle	716	\msg_error:nn 1010, 1207
\language	723	\msg_error:nnn 817, 964
\LaTeX	2401, 2412	\msg_error:nnnn 974
\LaTeXe	2157	\msg_info:nn 394, 2118, 2384
\leavevmode	475	\msg_new:nn 338, 340, 342, 347, 352, 354, 388, 2111, 2377
\leftskip	476, 477	\msg_new:nnnn 330
\levelchar	2186, 2266	\msg_warning:nnnn 1243
\list	458	\msg_warning:nnnnn 153
\listparindent	460	\multicolumn 1193, 1199
\llap	1374, 1436	
lm-default (option)	5	N
\LoadClass	397	\n 2435, 2437, 2438, 2439, 2447, 2450
\Lua	494	\nan 2428
\LuaTeX	494	\NB 5, 8, 1630
		\newcommand 530
		\NewDescribeEnvironment 4

<code>\NewDocElement</code>	1608	<code>\package_function_two:n</code>	8
<code>\NewDocumentCommand</code>		<code>\pageref</code>	1604
.....	539, 541, 543, 545, 604, 606,	<code>\par</code>	478,
.....	615, 624, 1632, 1638, 1771, 2404, 2428	999, 1003, 1484, 1498, 1502, 1508,
<code>\NewDocumentEnvironment</code>	1557, 1615, 1619, 1625, 1628, 1644,
..	547, 554, 578, 584, 593, 1642, 1653	1649, 1686, 1694, 1711, 1719, 1733
<code>\newenvironment</code>		<code>\parbox</code>	1143, 1866
.....	1611, 1621, 1680, 1696, 1720	<code>\parfillskip</code>	474, 1709
<code>\NewMacroEnvironment</code>	4	<code>\parg</code>	7, 530
<code>\nobreak</code>	478, 1556, 1709	<code>\parindent</code>	461, 473
<code>\noindent</code>		<code>\parskip</code>	463
.....	999, 1047, 1556, 1613, 1623, 1644, 1649	<code>\part</code>	1807, 2097, 2372
<code>\nolinebreak</code>	1139	<code>\partname</code>	464
<code>\nolinkurl</code>	535	<code>\PassOptionsToClass</code>	384, 387
<code>\normalfont</code>	1103	<code>\PassOptionsToPackage</code>	379
<code>\normalsize</code>	1103	<code>\pdfstringdefDisableCommands</code> ..	452, 523
NOTE (env.)	1640	<code>\pdfstringnewline</code>	449
<code>\NOTE</code>	5, 8	<code>\pdfTeX</code>	494
O			
<code>\oarg</code>	7, 530	<code>\penalty</code>	1707, 1708
<code>\obeylines</code>	1222	<code>\phantom</code>	754
<code>\obeyspaces</code>	1221	<code>\pkg</code>	8, 535, 2161, 2408
<code>\oddsidemargin</code>	456	<code>\prevdepth</code>	1004
<code>\OnlyDescription</code>	1734, 2395	prg commands:	
<code>onlydoc</code> (option)	5	<code>\prg_break:</code>	242, 244, 248, 254
<code>\OnlyDocumentation</code>	6	<code>\prg_break_point:</code>	256
<code>\openout</code>	1799	<code>\prg_generate_conditional_-</code>	
options:		variant:Nnn	94, 109
check	5	<code>\prg_new_conditional:Npnn</code> ..	293, 877
checktest	5	<code>\prg_new_protected_conditional:Npnn</code>	
cs-break	5	86, 101, 161
cs-break-nohyphen	5	<code>\prg_return_false:</code>	
full	5	91, 107, 173, 175, 314, 318, 323, 887
hide-notes	5	<code>\prg_return_true:</code>	92, 106,
kernel	5	164, 167, 172, 314, 318, 323, 880, 887
lm-default	5	<code>\PrintChanges</code>	1803, 1804, 1815, 2383
onlydoc	5	<code>\PrintIndex</code>	1801, 1802, 1814, 2117
show-notes	5	<code>\ProcessKeyOptions</code>	396
P			
package commands:		prop commands:	
<code>\package_function_one:N</code>	8	<code>\prop_get:NnN</code>	1664
		<code>\prop_get:NnNTF</code>	1522
		<code>\prop_gput:Nnn</code>	1659
		<code>\prop_if_empty:NTF</code>	2015

<code>\prop_map_function:NN</code>	2033	seq commands:	
<code>\prop_new:N</code>	20, 75, 1656	<code>\seq_clear:N</code>	775
<code>\prop_put:Nnn</code>	1525	<code>\seq_clear_new:N</code>	828
<code>\providecommand</code>	494,	<code>\seq_count:N</code>	1135, 1572, 1596
	495, 496, 497, 498, 499, 500, 501,	<code>\seq_gclear:N</code>	847, 1606
	502, 503, 504, 532, 533, 534, 1472, 1655	<code>\seq_get:NN</code>	844, 899
<code>\pTeX</code>	494	<code>\seq_get_left:NN</code>	2338
Q			
quark commands:		<code>\seq_get_right:NN</code>	2341
<code>\q_mark</code> 215, 216, 220, 221, 223, 294, 296		<code>\seq_gpop:NN</code>	852
<code>\q_no_value</code>	33,	<code>\seq_gput_right:Nn</code>	
	644, 645, 1024, 1033, 1297, 1325, 1341	.. 1120, 1348, 1353, 1421, 1483, 1546	
<code>\quark_if_no_value:NTF</code> 678, 684, 2134		<code>\seq_gremove_duplicates:N</code> 1956, 1957	
<code>\quark_if_recursion_tail_stop_-</code>		<code>\seq_gset_eq:NN</code>	851
do:nn	1930	<code>\seq_gset_filter:Nnn</code>	1583
<code>\q_recursion_stop</code>	1914	<code>\seq_if_empty:NTF</code>	
<code>\q_recursion_tail</code>	1914, 1929	.. 1113, 1389, 1593, 2037	
<code>\q_stop</code>	97, 98, 180, 198, 217,	<code>\seq_if_in:NnTF</code>	1481, 1960, 1972
	220, 221, 223, 294, 296, 326, 2302, 2305	<code>\seq_item:Nn</code>	1577, 1602
<code>\quotechar</code>	2266, 2269	<code>\seq_map_function:NN</code>	56, 793, 2052
R		<code>\seq_map_inline:Nn</code>	1397, 1958, 1970
<code>\raggedbottom</code>	418	<code>\seq_map_variable:Nnn</code>	142
<code>\raggedright</code>	1145, 1220, 1869	<code>\seq_new:N</code>	4, 5, 21, 22, 47, 49, 50
<code>\raisebox</code>	608, 617, 626	<code>\seq_pop:NN</code>	840, 898
<code>\rangle</code>	727	<code>\seq_pop_left:NN</code>	141
<code>\RecordChanges</code>	2387	<code>\seq_put_right:Nn</code>	776, 829, 831
regex commands:		<code>\seq_set_eq:NN</code>	897
<code>\regex_replace_all:nnN</code>	664	<code>\seq_set_filter:Nnn</code>	1544
<code>\regex_replace_once:nnNTF</code>	961	<code>\seq_set_from_clist:Nn</code>	58, 786, 1361
<code>\relax</code>	1709,	<code>\seq_set_map:Nnn</code>	1568, 2213
	1773, 1800, 1802, 1804, 1809, 1874	<code>\seq_set_split:Nnn</code>	
<code>\RenewCommandCopy</code>	1750	.. 84, 84, 140, 2212, 2336	
<code>\RequirePackage</code>	398, 399, 421, 422, 428	<code>\seq_use:Nn</code>	
<code>\rightskip</code>	473	58, 1136, 1139, 1364, 1517, 1548, 2215	
S		<code>\seq_use:Nnnn</code>	65, 1574
<code>\sb</code>	713	<code>\setcounter</code>	435, 1667, 2094
scan commands:		<code>\SetKeys</code>	385
<code>\scan_stop:</code>	14, 41,	<code>\setlength</code>	454, 461, 462, 463, 471
	104, 105, 808, 818, 845, 905, 1609, 2197	<code>\sffamily</code>	439
<code>\scriptsize</code>	1194, 1200	show-notes (option)	5
		<code>\small</code>	588, 1093, 1215, 1474
		<code>\space</code>	1779, 2020, 2026, 2044
		<code>\SpecialIndex</code>	2105

<code>\star</code>	734	<code>\@addtoreset</code>	436, 465
<code>\StopEventually</code>	4, 6, 1734	<code>\@auxout</code> 609, 618, 627, 1778, 1798, 1824	
str commands:		<code>\@beginparpenalty</code> ..	1684, 1701, 1725
<code>\c_backslash_str</code>	18, 446,	<code>\@bsphack</code>	1742, 2107
508, 513, 681, 766, 2246, 2259, 2282		<code>\@currenvir</code>	2174
<code>\c_percent_str</code>	780	<code>\@docinclude</code>	1780, 1782
<code>\str_case:nn</code>	2326	<code>\@dottedtocline</code>	483
<code>\str_case:nnTF</code>	183, 2177	<code>\@eha</code>	1779
<code>\str_count:n</code>	1455	<code>\@esphack</code>	1745, 1830, 2109
<code>\str_head:N</code>	2326	<code>\@evenfoot</code>	1872, 1887
<code>\str_if_eq:nnTF</code> .	845, 905, 2154, 2174	<code>\@indexfile</code>	1828, 1834, 1843, 1851
<code>\str_if_eq_p:nn</code>	265, 266	<code>\@input</code>	1785
<code>\str_tail:n</code>	2324	<code>\@latexerr</code>	362, 1779
<code>\c_underscore_str</code>	139, 765, 1563	<code>\@ltxdoc@PrintChanges</code>	1803, 1815
<code>\string</code>	609, 618, 627, 1779, 1785,	<code>\@ltxdoc@PrintIndex</code>	1801, 1814
1829, 1836, 1845, 1853, 2157, 2161		<code>\@ltxdoc@endtheglossary</code> ..	1806, 1817
<code>\strut</code>	1333, 1444, 1868	<code>\@ltxdoc@theglossary</code>	1805, 1816
<code>\subsection</code>	1682, 1698	<code>\@mainaux</code>	1785, 1824
<code>\subsubsection</code>		<code>\@nameuse</code>	1822
..	1690, 1699, 1716, 1722, 1723, 1729	<code>\@oddfoot</code>	1864, 1872, 1875, 1887
syntax (env.)	8, 578	<code>\@partaux</code> 1778, 1798, 1799, 1800, 1820	
sys commands:		<code>\@partlist</code>	1790
<code>\c_sys_day_int</code>	972	<code>\@plus</code>	470
<code>\c_sys_jobname_str</code> .	1951, 2115, 2381	<code>\@pnumwidth</code>	473, 474, 478
<code>\c_sys_month_int</code>	972	<code>\@secpenalty</code>	469
<code>\c_sys_year_int</code>	972	<code>\@starttoc</code>	80, 2084, 2088
T			
<code>\tabbing</code>	1730	<code>\@tempa</code>	1792
<code>\tabcolsep</code>	1096	<code>\@tempb</code>	1789, 1792
<code>\Team</code>	2399	<code>\@tempdima</code>	471, 476
<code>\TemplateArgument</code>	1686	<code>\@tempswafalse</code>	1788
<code>\TemplateKey</code>	1703, 1706	<code>\@tempswattrue</code>	1786, 1793
<code>\TemplateSemantics</code>	1687, 1713	<code>\@wrindex</code>	73, 1826
<code>\testfile</code>	1327, 1510	<code>\@writeckpt</code>	1819
<code>\TestFiles</code>	10	<code>\@xobeysp</code>	36, 670
<code>\TestFiles</code>	10, 1500	<code>_</code>	65
<code>\TestMissing</code>	10	<code>\bottomrule</code>	42, 51
<code>\TestMissing</code>	10, 1511	<code>\c@CodelineNo</code> 81, 1424, 1426, 1837, 1854	
<code>\TeX</code>	588, 612, 2157	<code>\c@footnote</code>	1660, 1676
TeX and L ^A T _E X 2 _ε commands:		<code>\c@HD@hypercount</code> ...	1095, 1853, 1854
<code>\@</code>	36, 431, 672, 1578	<code>\c@tocdepth</code>	468
<code>\@M</code>	1684, 1701, 1725	<code>\check@checksum</code>	1743
		<code>\Codedoc@explTF</code>	627, 745, 753

<code>\Codedoc@expstar</code>	609, 731	<code>\trivlist</code>	60
<code>\Codedoc@rexpstar</code>	618, 738	<code>\verb</code>	83
<code>\codeline@wrindex</code>	73, 1832	<code>\verbatim@font</code>	36, 83, 658, 1934, 2190
<code>\DocInput</code>	72	<code>\xmacro@code</code>	75, 1889
<code>\DoNotIndex</code>	29	<code>\z@</code>	468, 473
<code>\endtrivlist</code>	60	tex commands:	
<code>\expanded@notin</code>	445	<code>\tex_interactionmode:D</code> ...	2058, 2061
<code>\g@addto@macro</code>	2092	<code>\tex_lowercase:D</code>	1900
<code>\hb@xt@</code>	478	<code>texnote (env.)</code>	9, 584
<code>\HD@savedestfalse</code>	2070, 2076	<code>\text</code>	1934, 2428
<code>\HD@target</code>	2070	<code>\textbackslash</code>	520
<code>\HD@org@theCodelineNo</code>	438	<code>\textbf</code>	588, 1726
<code>\Hy@footnote@currentHref</code>	1655, 1660, 1665	<code>\textcolor</code>	439
<code>\Hy@MakeCurrentHref</code>	1095	<code>\textit</code>	1493, 1503
<code>\if@partsw</code>	1787	<code>\textrm</code>	1139, 1140, 1147, 1153
<code>\if@tempwa</code>	1797	<code>\textsf</code>	537, 538, 1496, 1506
<code>\ifnot@excluded</code>	443	<code>\texttt</code>	531, 533, 534, 536, 597, 611, 612, 613, 621, 622, 628, 629, 1564, 1644, 2428
<code>\include</code>	72	<code>\textwidth</code>	454, 1046, 1210, 1866
<code>\index@excludelist</code>	29, 447	<code>\thanks</code>	2401, 2412
<code>\index@prologue</code>	1862	<code>\the</code>	721, 1853, 1854
<code>\init@checksum</code>	1744	<code>\theCodelineNo</code>	437, 2073, 2077
<code>\input</code>	72	<code>\theglossary</code>	1805, 1816
<code>\it@is@a</code>	82, 2119	<code>\theindex</code>	2092
<code>\l@nohyphenation</code>	723	<code>\thepage</code>	1829, 1846, 1885
<code>\l@section</code>	465	<code>\thepart</code>	1811, 1859, 1860, 1878, 1881
<code>\l@subsection</code>	465	<code>\tiny</code>	439
<code>\m@ne</code>	722	<code>\title</code>	2406
<code>\macro@namepart</code>	29, 446	tl commands:	
<code>\meta</code>	38	<code>\c_catcode_active_space_tl</code>	145
<code>\meta@font@select</code>	719	<code>\c_catcode_other_space_tl</code>	843, 1459
<code>\meta@hyphen@restore</code>	720, 725	<code>\c_space_tl</code>	1518, 2190, 2260
<code>\midrule</code>	51	<code>\tl_clear:N</code> .	792, 1326, 1912, 1953, 1954, 1955, 2007, 2014, 2277, 2344
<code>\nfss@text</code>	717	<code>\tl_const:Nn</code>	57, 59, 1943, 1944, 2198, 2399
<code>\p@</code>	470	<code>\tl_count:n</code>	89, 90
<code>\part</code>	4, 81, 90	<code>\tl_gclear:N</code>	1756, 1840
<code>\partname</code>	4	<code>\tl_gput_right:Nn</code> ..	1862, 2117, 2383
<code>\protected@write</code>	1828, 1843	<code>\tl_greplace_all:Nnn</code>	842
<code>\saved@indexname</code>	60, 1428	<code>\tl_gset:Nn</code>	464, 1665, 1743, 1933
<code>\saved@macroname</code>	60, 1403		
<code>\texttt</code>	64, 65		
<code>\toprule</code>	51		

<code>\verb</code>	2239, 2247, 2252		W
<code>\Verbatim</code>	441	<code>\write</code> 609, 618, 627, 1785, 1800, 1834, 1851	
<code>\verbatim</code>	440 , 1645		
<code>\verbatimchar</code>	2093, 2237,		X
	2239 , 2242 , 2246 , 2247 , 2252 , 2253		
<code>\vskip</code>	1336 , 1728	<code>\XeTeX</code>	494
<code>\vspace</code>	587 , 591		
<code>\vss</code>	1380		Z
<code>\vtop</code>	1377	<code>\Z</code>	962