

The L^AT_EX dtxdescribe Package

v1.08 — 2024/01/08

© 2016–2024 Brian Dunn

<https://github.com/bdtd/dtxdescribe>

Describe additional object types in dtx or tex source files.

Abstract

The `doc` package includes tools for describing macros and environments in L^AT_EX source `.dtx` format. The `dtxdescribe` package adds additional tools for describing booleans, lengths, counters, hooks, sockets, plugs, keys, packages, classes, options, files, commands, arguments, and other objects. `dtxdescribe` also works with the regular document classes, for those who do not wish to use the `ltxdoc` class and `.dtx` files.

Each described item is given a margin tag similar to `\DescribeEnv`, and is listed in the index by itself and also by type of object displayed in parentheses (length, filename, etc). Each item may be sorted further by an optional category displayed in brackets, such as `[category_name]`.

The `dtxexample` environment is provided for typesetting example code and its results. Contents are displayed verbatim along with a caption and cross-referencing. They are then `\input` and executed, and the result is shown.

Environments are also provided for displaying verbatim or formatted source code, user-interface displays, and sidebars with titles.

Macros are provided for formatting the names of inline L^AT_EX objects such as packages and booleans, as well as program and file names, file types, internet objects, the names of certain programs, a number of logos, and inline dashes and slashes.

`dtxdescribe` works with the `ltxdoc` class, but also works with the standard classes as well, except that the `macro` and `environment` environments are not supported. Either `makeidx` or `splitidx` may be loaded by the user. `makeidx` will be used by default.

`dtxdescribe` works with pdf_TE_X, X_TE_X, and Lua_TE_X, and perhaps other engines as well.

See change in setup of `hyperref` and `cleveref`: Section 2 on page 7.

Contents

1	Introduction	6
2	Using dtxdescribe	7
3	The macros, and the dtxexample environment	9
3.1	Macros and environments	9
3.2	Arguments	9
3.3	Booleans, lengths, counters, hooks, sockets, plugs, keys	10
3.4	Packages, classes, options	10
3.5	Files, programs, commands	11
3.6	Other source objects	11
3.7	In a description environment	12
3.8	Defaults	12
3.9	Nesting	13
3.10	\margintag, \watchout	13
3.11	dtxexample environment	13
3.12	noindmacro and noindenvironment environments	14
3.13	sourceverb, sourcedisplay, UIDisplay, docsidebar	14
3.14	Formatted objects	15
3.14.1	L ^A T _E X objects	15
3.14.2	Programs and commands	16
3.14.3	File types	16
3.14.4	Internet	16
3.14.5	Specific programs	17
3.14.6	Acronyms, brand names, trademarks	17
3.15	Logos	17
3.16	Dashes and slashes	18

4 Examples	19
5 Usage notes	32
6 Code	33
6.1 Required packages	33
6.2 Warning sign	35
6.3 Special character handling	35
6.4 Patching hypdoc, splitidx, doc	35
6.5 Gobbling comment characters	37
6.6 Vertical spacing	38
6.7 Not ltxdoc: ltxdoc emulation	38
6.8 Not doc: doc emulation	38
6.9 Support macros	39
6.10 doc: Key handling for object categories	43
6.11 doc: Handling \marginpar inside a float	44
6.12 doc: Handling categories and detokenizing names	44
6.13 Not doc: \DescribeMacro and \DescribeEnvironment	53
6.14 New \Describe. . . macros	54
6.15 \DescribeDefault	63
6.16 \ItemDescribeMacro, etc.	63
6.17 \margintag, \watchout	67
6.18 Nesting	67
6.19 The dtxexample environment	68
6.20 noindmacro and noindenvironment	70
6.21 sourcedisplay, UIdisplay, docsidebar	71
6.22 Formatted objects	73
6.22.1 L ^A T _E X objects	73
6.22.2 Programs and commands	74

6.22.3	File types	75
6.22.4	Internet	76
6.22.5	Specific programs	77
6.22.6	Acronyms, brand names, trademarks	77
6.23	Logos	77
6.24	Dashes and slashes	79
7	Compiling dtxdescribe	80
	Change History and Index	81

List of Examples

1	Macros	19
2	Environment	20
3	Second Environment	20
4	Booleans and Counters	21
5	Lengths	21
6	Hooks and Sockets	21
7	Packages, Classes, and Options	22
8	Files, Commands, and Programs	22
9	Keys	23
10	Arguments	24
11	Object	25
12	Other	25
13	Description environments	26
14	Nesting	27
15	dtxexample	28
16	fsourceverb	29

dtxdescribe	5
-------------	---

17	<code>sourcedisplay</code>	29
18	<code>UIdisplay</code>	30
19	<code>docsidebar</code>	31

List of Figures

1	A Figure	28
---	----------	----

1 Introduction

The `doc` package provides `\DescribeMacro` and `\DescribeEnv` to help document new macros and environments. Each generates a heading in the documentation, to which `\marg`, `\oarg`, and `\parg` may be added to identify arguments to be passed to the new object. Their names are added to the margin, and index entries are added, as well as group of entries for environments.

`dtxdescribe` extends this concept to include a number of additional objects, such as booleans and keys. To help identify what is being described in the margin, small tags are added to the name, such as “Env”, “Bool”, or “Key”. These new objects are also listed in the index with the same tag shown after their names, and also by group. Optional categories may be used to further sort index entries.

Modifications have been made to interact with `hyperref` to provide hyper links for regular index entries as well as the new `\Describe` entries.

Additional macros are provided to generate colored margin tags and warnings, and a new `dtxexample` environment demonstrates code examples.

This documentation and its index show examples of these macros in use.

While the index may appear to be overkill for a small package, keep in mind that it includes a number of fictional entries from the examples. Extensive cross-referencing can be useful for larger works. And, of course, you need not cross-reference everything!

2 Using dtxdescribe

To use dtxdescribe with the ltxdoc class and .dtx files:

```
%<*driver>
\documentclass{ltxdoc}
...
\usepackage{lmodern}
...
\usepackage{dtxdescribe}
...
\usepackage{packagename} % the name of your new package
...

% hyperref now automatically loaded by \pkg{ltxdoc}.

\AtBeginDocument{
  \hypersetup{...}%           If needed.
  \pdfstringdefDisableCommands{ ... }% If needed.
}

\AddToHook{begindocument/before}{% Before .aux file is loaded.
  \usepackage{cleveref}%           If needed.
  \crefname{somename}{name}{names}% If needed.
}

...
%</driver>
```

To use dtxdescribe with the regular classes, such as article, use:

```
\documentclass{article}

\usepackage{dtxdescribe}

\usepackage{hyperref}%           If needed.
  \hypersetup{...}%           If needed.
  \pdfstringdefDisableCommands{ ... }% If needed.

\usepackage{cleveref}%           If needed.
  \crefname{somename}{name}{names}% If needed.


\begin{document}
...
\end{document}
```

Various objects inside the dtx or tex file may be described with new macros such as `\DescribeBoolean`, `\DescribeLength`, `\DescribeCounter`, similar to the already-familiar `\DescribeMacro` and `\DescribeEnv`.

Optional “categories” may be assigned to the objects being described, including the new versions of `\DescribeMacro` and `\DescribeEnv`. These categories are printed in the margin tag and index entry for each item, and also generate additional index entries sorted by category. This is especially useful for key/value sets, where several sets may appear in the same document.

inside a float The margin tag is not printed if the `\Describe` macros are used inside a float such as a table, but the index entries are still made.

`\margintag{text}` `\margintag{text}` may be used to place a colored tag in the margin to summarize paragraph contents or draw attention to an index destination.

 `\watchout[optional text]` `\watchout[optional text]` may be used to place a red warning sign in the margin, along with optional text.

The `dtxexample` environment may be used to typeset and execute small pieces of \LaTeX code as examples of its use. Optional cross-referencing notes may be used to refer to any example float being generated. `\listofdtxexamplefloats` prints the list of examples.

3 The macros, and the dtxexample environment

3.1 Macros and environments

macro (*env*) These are only provided by the ltxdoc class and doc package to document a .dtx file, where comments are used by *docstrip* to disable these environments in the resulting .sty file. When using the regular document classes, the macro and environment environments would localize any definitions, and \DescribeMacro and \DescribeEnv should be used instead.

⚠ .dtx only

\DescribeMacro [*<category>*] {*<\name>*}

The preexisting macro from the doc package is redefined to create hyperlinked index entries, and include an optional category. A margin tag is created and an index entry is made. When the optional category is used, it is displayed along with the margin tag, and is used to group an index entry by macro name and another index entry by category. An example would be to describe the float creation and caption setup for a new category of float, such as the dtxexample float and the example “photograph” float both found in the index for this document. See example 1 on page 19 for examples.

\DescribeEnv [*<category>*] {*<environment name>*}

The preexisting macro from the doc package is redefined to create hyperlinked index entries, include an optional category, and also to place an ‘Env’ tag in front of the name in the margin. See example 2 on page 20.

3.2 Arguments

The \Describe. . . macros may be followed by \marg, \oarg, and \parg to describe arguments passed to the macros.

\marg {*<text>*}

Shows a mandatory argument for a macro or environment.

The results looks like {*<mandatory>*}.

\oarg {*<text>*}

Shows an optional argument for a macro or environment.

The results looks like [*<optional>*].

\parg {*<text>*}

Used for “picture” arguments, such as coordinates.

The result looks like (*<coordinate>*).

`\DescribeArgument` [*category*] {*argument*}

May be used to describe actions taken when given certain macro arguments. These will be given an ‘Arg’ margin tag and will appear in the index. The category may be used to categorize arguments by their macro or environment name. See example 10 on page 24.

3.3 Booleans, lengths, counters, hooks, sockets, plugs, keys

See example 4 on page 21.

`\DescribeBoolean` [*category*] {*name*}

Describes a boolean. Given a ‘Bool’ tag in the margin and index.

`\DescribeLength` [*category*] {*name*}

Describes a length. Given a ‘Len’ tag in the margin and index.

`\DescribeCounter` [*category*] {*name*}

Describes a counter. Given a ‘Ctr’ tag in the margin and index.

`\DescribeHook` [*category*] {*name*}

Describes a hook. Given a ‘Hook’ tag in the margin and index. The category may be used to categorize hooks by package. The hook name may or may not have a backslash. Example:

```
\DescribeHook{\hookname}
\DescribeHook[LaTeX]{para/begin}
```

`\DescribeSocket` [*category*] {*name*}

`\DescribePlug` [*category*] {*name*}

Describes a socket or plug. Given a ‘Socket’ or ‘Plug’ tag in the margin and index. The category may be used to categorize sockets and plugs by package. Example:

```
\DescribeSocket{socketname}
\DescribePlug{plugname}
```

`\DescribeKey` [*category*] {*name*}

Describes a key. Given a ‘Key’ tag in the margin and index. The category may be used to categorize keys by their key/value group. See example 9 on page 23.

3.4 Packages, classes, options

`\DescribePackage` [*category*] {*name*}

Describes a package. Given a ‘Pkg’ tag in the margin and index.

`\DescribeClass` [*<category>*] {*<name>*}

Describes a L^AT_EX class. Given a ‘Cls’ tag in the margin and index.

`\DescribeOption` [*<category>*] {*<name>*}

Describes a L^AT_EX package or class option. Given an ‘Opt’ tag in the margin and index.

3.5 Files, programs, commands

`\DescribeFile` [*<category>*] {*<name>*}

Describes an operating-system file. Given a ‘File’ tag in the margin and index. The filename may have underscores.

`\DescribeProgram` [*<category>*] {*<name>*}

Describes an operating-system program. Given a ‘Prog’ tag in the margin and index. The program name may have underscores.

`\DescribeCommand` [*<category>*] {*<name>*}

Describes an operating-system command. Given a ‘Cmd’ tag in the margin and index. The command name may have underscores.

3.6 Other source objects

`\DescribeObject` [*<category>*] {*<name>*}

Describes an arbitrary programming object, such as a color definition or caption setup. A margin tag and index entry are created with `\ttfamily` type. When a category is used, it is added to the margin tag, appended to the index entry, and a second index entry is created grouped by category. If a macro name is to be described, use `\DescribeMacro` instead. See example 11 on page 25.

`\DescribeOther` [*<category>*] {*<name>*}

Describes an arbitrary non-programming object, such as a license agreement or credits. A margin tag and index entry are created in roman type. When a category is used, it is added to the margin tag, appended to the index entry, and a second index entry is created grouped by category. See example 12 on page 25.

3.7 In a description environment

To describe an object using a description environment, use the following. See example 13 on page 26.

<code>\ItemDescribeMacro</code>	<code>[<category>]{<\name>}</code> A description.
<code>\ItemDescribeEnv</code>	<code>[<category>]{<name>}</code> A description.
<code>\ItemDescribeArgument</code>	<code>[<category>]{<argument>}</code> A description.
<code>\ItemDescribeBoolean</code>	<code>[<category>]{<name>}</code> A description.
<code>\ItemDescribeLength</code>	<code>[<category>]{<\name>}</code> A description.
<code>\ItemDescribeCounter</code>	<code>[<category>]{<name>}</code> A description.
<code>\ItemDescribeHook</code>	<code>[<category>]{<name>}</code> A description.
<code>\ItemDescribeSocket</code>	<code>[<category>]{<name>}</code> A description.
<code>\ItemDescribePlug</code>	<code>[<category>]{<name>}</code> A description.
<code>\ItemDescribeKey</code>	<code>[<category>]{<name>}</code> A description.
<code>\ItemDescribePackage</code>	<code>[<category>]{<package_name>}</code> With underscores.
<code>\ItemDescribeClass</code>	<code>[<category>]{<class_name>}</code> With underscores.
<code>\ItemDescribeOption</code>	<code>[<category>]{<name>}</code> A description.
<code>\ItemDescribeFile</code>	<code>[<category>]{<file_name>}</code> With underscores.
<code>\ItemDescribeProgram</code>	<code>[<category>]{<program_name>}</code> With underscores.
<code>\ItemDescribeCommand</code>	<code>[<category>]{<command_name>}</code> With underscores.
<code>\ItemDescribeObject</code>	<code>[<category>]{<name>}</code> A description.
<code>\ItemDescribeOther</code>	<code>[<category>]{<name>}</code> A description.

3.8 Defaults

<code>\DescribeDefault</code>	<code>{<value>}</code>
Default: <code>value</code>	Shows the default value of a <code>\Describe. . .</code> item, such as displayed here. Place this macro immediately after the <code>\Describe. . .</code> macro and any arguments, but before the text description.
<code>\DescribeDefaultcolor</code>	The color of the margin tag used to show the default value. This is used by <code>\textcolor</code> to create the margin tag.
Default: <code>green!50!black</code>	

3.9 Nesting

`\shownesting` * [*<fraction of \linewidth>*] {*<container name>*} {*<contents>*}

It may be useful to show which objects contain which other objects. `\shownesting` shows a box enclosing a name for the container, and the container's contents. `\shownesting` be nested, showing boxes inside other boxes, which displays how each environment and macro is fit together inside each other.

The optional argument is the fraction of `\linewidth` to use for the box, from [0] to [1]. The default is [1]. Each `\shownesting` starts its own paragraph, unless the star `*` is used, in which case the `\shownesting*` appears inline with previous text. To place two `\shownesting` boxes side-by-side, use optional arguments to specify less than full `\linewidth` for each box, and use `\shownesting*` for the second box to place it inline.

See example 14 on page 27 for an example.

3.10 `\margintag`, `\watchout`

`\margintag` {*<text>*}

Creates a colored margin tag. May be used to identify the topic of a paragraph or the destination of an arbitrary index entry.

`\margintag{example}`

`\margintagcolor` The color of the `\margintag`.

Default: `blue!70!black`

`\watchout` [*<text>*]

Creates a red margin tag with a warning sign and optional text. May be used to warn the reader of special instructions, etc. Without the optional text the warning sign is displayed by itself.

 `\watchout{example}`

`\watchoutcolor` The color of the `\watchout`.

Default: `red!50!black`

3.11 `dtxexample` environment

`dtxexample (env.)` * [*<Notes/cross-references>*] {*<caption & label>*}

The `dtxexample` environment is useful for demonstrating a piece of \LaTeX code. The example is a simulated float with its own caption and optional label, along with optional notes and/or cross-referencing commands. The contents of the `dtxexample` environment are printed verbatim, then loaded and executed as \LaTeX code, showing the results just below the printed code. In the case of float commands, the floats are generated as expected somewhere nearby, and should be given their own labels. References to the float's labels may be placed in the optional argument to the `dtxexample` environment, and will be printed below the code.

The unstarred version places the code inside a minipage, forbidding a page break in the middle of the code listing. The starred version does not use a minipage. This is required when the code is too large to fit on a single page.


`listofdtxexamplefloats` Prints the list of examples.

See example 15 for a demonstration of how `dtxexample` works.

The examples may be customized by redefining the following, perhaps for another language:

`\dtxexamplecodename` The text name of the code section.
 Default: `Code:`
`\dtxexampleresultname` The text name of the result section.
 Default: `Result:`

3.12 `noindmacro` and `noindenvironment` environments

 **.dtx only** These are like macro and environment, but not indexed. These only make sense if using the `ltxdoc` class and `doc` package to document a .dtx file, where comments are used by *docstrip* to disable these environments in the resulting .sty file. When using the regular document classes, `noindmacro` and `noindenvironment` environments should not be used, as they would localize any definitions. `\DescribeMacro` and `\DescribeEnv` should be used instead.

`noindmacro (env)` `{\name}` To document macros which should not be included in the index.

`noindenvironment (env)` `{\name}` To document environments which should not be included in the index.

Replace

```
\begin{macro}{\macroname} \oarg{optional} \marg{mandatory}
...
\end{macro}
```

with

```
\begin{noindmacro}{\macroname} \oarg{optional} \marg{mandatory}
...
\end{noindmacro}
```

and similarly for `noindenvironment`.

3.13 `sourceverb`, `sourcedisplay`, `UIDisplay`, `docsidebar`

`sourceverb (env)` `[\key/values]` Display source code verbatim. Uses optional `fancyvrb` keys. Includes `gobble=2` to absorb the leading % and space character of a dtx file source format. Because this is a verbatim environment, it *cannot* be used inside a macro.
 Default: `gobble=2`,
 `tabsize=4`,
 `xleftmargin=2em`

<code>fsourceverb (env)</code>	<code>[⟨key/values⟩]</code>	Display source code verbatim inside a frame. A label may be included using the <code>label</code> key. Because this is a verbatim environment, it <i>cannot</i> be used inside a macro. See example 16 on page 29.
<code>Default: gobble=2,</code> <code>tabsize=4,</code> <code>xleftmargin=2em, frame=lines</code>		
<code>sourcedisplay (env)</code>		Display source code with manual formatting. This is not a verbatim environment. <code>\textcolor</code> , <code>\textbf</code> , and <code>\emph</code> may be used to highlight text. Macros must be escaped with <code>\cs</code> , characters such as <code>{</code> must be produced with <code>\{</code> , etc. <code>\</code> must be used to force a new line. <code>\fquad</code> , <code>\fqquad</code> , and <code>\fqqqquad</code> may be used to force indenting. Because this is <i>not</i> a verbatim environment, it <i>can</i> be used inside a macro. See example 17 on page 29.
<code>\fquad</code>		Single-level indent inside a <code>sourcedisplay</code> .
<code>\fqquad</code>		Double-level indent inside a <code>sourcedisplay</code> .
<code>\fqqqquad</code>		Triple-level indent inside a <code>sourcedisplay</code> .
<code>UIdisplay (env)</code>		Displays a user interface, such as a dialog box entry or a menu selection. See example 18 on page 30. Also see the <code>\UI</code> macro..
<code>\userentry</code>	<code>{⟨text to enter⟩}</code>	Typeset something for the user to enter. Also see the <code>\cmds</code> macro.
<code>\userentryname</code>		Text to tell the user to enter the following item. Change with <code>\renewcommand</code> .
<code>Default: Enter ⇒</code>		
<code>docsidebar (env)</code>	<code>[⟨title⟩]</code>	Creates a sidebar within the document. See example 19 on page 31.

3.14 Formatted objects

Macros to format references to various kinds of objects.

This dtxdescribe package documentation uses `erewhon`, `cabin`, and `inconsolata`, along with `metalogo`, to demonstrate the following font effects.

3.14.1 L^AT_EX objects

<code>\pkg</code>	<code>{⟨packagename⟩}</code>	Prints as <code>packagename</code> . Also for a <code>classname</code> .
<code>\cs</code>	<code>{⟨csname⟩}</code>	Prints as <code>\csname</code> .
<code>\env</code>	<code>{⟨environment⟩}</code>	Prints as <code>environment</code> .
<code>\marg</code>	<code>{⟨argument⟩}</code>	Prints <code>{⟨arg⟩}</code> . Mandatory argument.
<code>\oarg</code>	<code>{⟨argument⟩}</code>	Prints <code>[⟨arg⟩]</code> . Optional argument.
<code>\parg</code>	<code>{⟨argument⟩}</code>	Prints <code>(⟨arg⟩)</code> . Picture-mode argument.
<code>\ctr</code>	<code>{⟨counter⟩}</code>	Prints as <code>counter</code> .

<code>\bool</code>	<code>{\boolean}</code>	Prints as boolean.
<code>\optn</code>	<code>{\option}</code>	Prints as option, for example to a macro, package, class.
<code>\TOC</code>		TOC: Table of contents.
<code>\LOF</code>		LOF: List of figures.
<code>\LOT</code>		LOT: List of tables.

3.14.2 Programs and commands

<code>\progcode</code>	Prints as inline program code: Escape underscores and other special characters such as {, %, \$.	
<code>\prog</code>	Prints as <i>grep</i> , <i>make</i> : A program name. Underscores allowed.	
<code>\filenm</code>	Prints as file_name: Underscores allowed.	
<code>\UI</code>	Prints as General user-interface text . What the user sees on the display. Also see the <code>UIDisplay</code> environment.	
<code>\cmds</code>	Prints as Commands to be entered : What the user enters. Escape underscores and other special characters such as {, %, \$. Also see the <code>\userentry</code> macro.	

3.14.3 File types

<code>\ODT</code>	ODT OpenDocument Format word processing document
<code>\SVG</code>	SVG image format
<code>\PNG</code>	PNG image format
<code>\GIF</code>	GIF image format
<code>\JPG</code>	JPG image format
<code>\EPS</code>	EPS image format
<code>\PDF</code>	PDF image format
<code>\DVI</code>	DVI image format

3.14.4 Internet

<code>\UTF</code>	UTF: Unicode
<code>\URL</code>	URL: Uniform Resource Locator
<code>\element</code>	<code>{\element name}</code> Prints as <element>, an HTML/CSS element

<code>\attribute</code>	<code>{\langle attribute name \rangle}</code>	Prints as attribute, an HTML/CSS attribute. pdfTeX and XeTeX only. Not for LuaTeX.
<code>\attrib</code>	<code>{\langle attribute name \rangle}</code>	Prints as attribute, an HTML/CSS attribute. pdfTeX, XeTeX, or LuaTeX.
<code>\HTML</code>		HTML: Hypertext Markup Language
<code>\HTMLfive</code>		HTML5: Old-style figure if font supports
<code>\CSS</code>		css: Cascading Style Sheet
<code>\CSSthree</code>		css3: Old-style figure if font supports
<code>\EPUB</code>		EPUB: E-book file format

3.14.5 Specific programs

<code>\TikZ</code>		TikZ: Package logo
<code>\CTAN</code>		CTAN: Comprehensive TeX Archive Network
<code>\TDS</code>		TDS: TeX Directory Structure
<code>\MathML</code>		MathML: Mathematical Markup Language
<code>\MathJax</code>		MATHJAX: Math on the web.

3.14.6 Acronyms, brand names, trademarks

<code>\brand</code>	<code>{\langle name \rangle}</code>	BRANDNAME, COMPANY NAME
<code>\acro</code>	<code>{\langle acronym \rangle}</code>	ACRO: Acronym
<code>\supregistered</code>		Superscript trademark symbol®

3.15 Logos

Several additional logos are provided.

Also see the `metalogo` and `metalogox` packages.

<code>\dviTeX</code>	DVI TeX
<code>\dviLaTeX</code>	DVI L ^A TeX
<code>\pdfTeX</code>	PDF TeX
<code>\pdfLaTeX</code>	PDF L ^A TeX

<code>\LuaTeX</code>	LuaTeX
<code>\LuaLaTeX</code>	LuaLaTeX
<code>\XeTeX</code>	X _Ǝ TeX, with reversed E if <code>graphics</code> is loaded.
<code>\XeLaTeX</code>	X _Ǝ LaTeX, with reversed E if <code>graphics</code> is loaded.
<code>\AmS</code>	\mathcal{MS}
<code>\LyX</code>	LyX
<code>\BibTeX</code>	BibTeX
<code>\MakeIndex</code>	<i>MakeIndex</i>
<code>\ConTeXt</code>	ConTeXt
<code>\MiKTeX</code>	MiKTeX

3.16 Dashes and slashes

<code>\thinspace</code>	A breakable thin skip.
<code>\endash</code>	An endash: –
<code>\emdash</code>	An emdash: —
<code>\thinbrspace</code>	A thin space which allows a line break.
<code>\thinthinbrspace</code>	A very thin space which allows a line break.
<code>\Dash</code>	An unbreakable thin space, emdash, and breakable thin space: A—B
<code>\dash</code>	An unbreakable thin space, endash, and breakable thin space: A–B
<code>\Slash</code>	An unbreakable very thin space, a slash, and a breakable very thin space:

Command	Result
<code>A--B</code>	A–B (not breakable)
<code>A \dash B</code>	A–B (only breakable before the B)
<code>A -- B</code>	A–B (breakable before or after the dash)
<code>A---B</code>	A—B (not breakable)
<code>A \Dash B</code>	A—B (only breakable before the B)
<code>A --- B</code>	A—B (breakable before or after the dash)
<code>A/B</code>	A/B (not breakable)
<code>A \Slash B</code>	A/B (only breakable before the B)
<code>A / B</code>	A / B (breakable before or after the slash)
<code>A~/~B</code>	A / B (not breakable)

4 Examples

Example 1: Macros

Code:

```
\DescribeMacro{\mymacro} \oarg{optional} \marg{mandatory}
  A typical macro definition.
```

```
\DescribeMacro[photograph]{\DeclareFloatingPhoto}
Create a photograph float environment.
```

```
\DescribeMacro[c=photograph]{\photocaptionsetup}
Caption settings for a photograph float.
```

```
\DescribeMacro[photograph]{\cphotonameref}
\pkg{cleveref} name for the photograph float.
```

Result:

<code>\mymacro</code>	<code>[<i><optional></i>] [<i><mandatory></i>]</code> A typical macro definition.
<code>\DeclareFloatingPhoto</code>	Create a photograph float environment.
<code>[photograph]</code>	
<code>\photocaptionsetup</code>	Caption settings for a photograph float.
<code>[photograph]</code>	
<code>\cphotonameref [photograph]</code>	<code>cleveref</code> name for the photograph float.

The optional category is used to label and group tags and index entries. See this document's index entries for examples of this “photograph” category and the `dtxexample` category of macros.

The re-defined `\DescribeMacro`, `\DescribeEnv`, and all the following macros create [hyperlinks](#) hyperlinked index entries, along with regular uses of `\index`.

Example 2: Environment

Code:

```
\DescribeEnv{myenvironment} \marg{argument} Short description.
```


Result:

```
myenvironment (env)  {⟨argument⟩} Short description.
```

add'l tags

index groups

The re-defined \DescribeEnv adds an ‘Env’ tag to the margin, and adds “(environment)” to its own index entry. Note that environments and all the other new objects defined by this package each receives two index entries, one by name, and one grouped with others of its kind.

 too much text

Example 2 shows descriptive text on the same line as the \DescribeEnvironment. For macros and environments with many arguments after the name, it may be better to place any additional text in a following paragraph.

Example 3: Second Environment

Code:

```
\DescribeEnv[kindofenvironment]{otherenvironment}
  \oarg{opt args} \parg{coordinates} A description.
```

Result:

```
otherenvironment (env)  [⟨opt args⟩] (⟨coordinates⟩) A description.
[kindofenvironment]
```

The otherenvironment will be indexed by itself and also with myenvironment under the index entry “environments”, and also under the category kindofenvironment.

Example 4: Booleans and Counters

Code:

\DescribeBoolean[examples]{sampleboolean} Some description.

\DescribeCounter[examples]{samplecounter} Some description.

Result:

sampleboolean (*bool*) [examples] Some description.

samplecounter (*Ctr*) [examples] Some description.

Most of the new \Describe_____ macros behave like the new \DescribeEnv, placing a tag in the margin, an index entry by name, and another index entry by group.

Example 5: Lengths

Code:

\DescribeLength[photograph]{\photowidth} Some description.

Result:

\photowidth (*Len*) [photograph] Some description.

Lengths have a leading backslash, but are otherwise described the same as the rest of the objects.

Example 6: Hooks and Sockets

Code:

\DescribeHook{\hookname} A hook with a backslash.

\DescribeHook{para/begin} A hook without a backslash.

\DescribeSocket{socketname} A socket.

\DescribePlug{plugname} A socket plug.

Result:

\hookname (*Hook*) A hook with a backslash.

para/begin (*Hook*) A hook without a backslash.

socketname (*Socket*) A socket.

plugname (*Plug*) A socket plug.

Hooks may or may not have a leading backslash.

Example 7: Packages, Classes, and Options

Code:

```
\DescribePackage[examples]{samplepackage}
  About a \LaTeX\ package.

\DescribeClass[examples]{sample_class}
  About a \LaTeX\ class.

\DescribeOption[examples]{sampleoption}
  About an option for a package or class.
```

Result:

samplepackage (<i>Pkg</i>) [examples]	About a L ^A T _E X package.
sample_class (<i>Cls</i>) [examples]	About a L ^A T _E X class.
sampleoption (<i>Opt</i>) [examples]	About an option for a package or class.

Example 8: Files, Commands, and Programs

Code:

```
\DescribeFile[bigfiles]{really_big_file.txt} Some description.

\DescribeFile[bigfiles]{another_big_file.txt} Some description.

\DescribeFile{lone_file.txt} Some description.

\DescribeCommand{OS_command} An operating-system command.

\DescribeProgram{program_name} An operating-system program.
```

Result:

really_big_file.txt (<i>file</i>)	Some description.
[bigfiles]	
another_big_file.txt (<i>file</i>)	Some description.
[bigfiles]	
lone_file.txt (<i>file</i>)	Some description.
OS_command (<i>Cmd</i>)	An operating-system command.
program_name (<i>Prog</i>)	An operating-system program.

Filenames, program names, and command names may have underscores, such as tested here. A category is used to group “bigfiles” together in the index.

Example 9: Keys

Code:

```
\DescribeKey[groupofkeys]{firstkey} About the first key
    of the |groupofkeys| set.

\DescribeKey[groupofkeys]{secondkey} About the second key
    of |groupofkeys|.

\DescribeKey[examples]{samplekey} About some key of |otherkeys|.

\DescribeKey[examples]{sampletwokey} About another key of |otherkeys|.

\DescribeKey{lonekey} A key without a category.
```

Result:

firstkey (Key) [groupofkeys]	About the first key of the groupofkeys set.
secondkey (Key) [groupofkeys]	About the second key of groupofkeys.
samplekey (Key) [examples]	About some key of otherkeys.
sampletwokey (Key) [examples]	About another key of otherkeys.
lonekey (Key)	A key without a category.

See the index key groups.

Example 10: Arguments

Code:

```
\DescribeArgument[figure]{[H]}
What happens when a figure is [H]ere.
```

```
\DescribeArgument[figure]{[M]}
What happens when a figure is in the [M]argin.
```

```
\DescribeArgument[\mymacro]{bold}
What happens when \cs{mymacro} is given the |bold| argument.
```


Result:

[H] (*Arg*) [figure] What happens when a figure is [H]ere.

[M] (*Arg*) [figure] What happens when a figure is in the [M]argin.

bold (*Arg*) [\mymacro] What happens when \mymacro is given the bold argument.

Arguments behave like keys, and may have an optional category to identify their macro or environment, and group their entries in the index.

 **macro names** Note you may need to use \cs{mymacro} for the macro's name.

Example 11: Object

Code:

```

\DescribeObject[color]{somecolor}
    The color of something.

\DescribeObject[color]{othercolor}
    The other color.

\DescribeObject{randomobject} About some random object.

```

Result:

somecolor [color]	The color of something.
othercolor [color]	The other color.
randomobject	About some random object.

Describes an arbitrary programming object, using \ttfamily text.

Example 12: Other

Code:

```

\DescribeOther{license agreement}
The following is the fictional license agreement:

\DescribeOther{Before myenvironment}
    Actions to be done \cs{BeforeBeginEnvironment}.

\DescribeOther[othercategory]{Other Item} About the other item.

\DescribeOther[othercategory]{Additional Item} About the add'l item.

```

Result:

license agreement	The following is the fictional license agreement:
Before myenvironment	Actions to be done \BeforeBeginEnvironment.
Other Item [othercategory]	About the other item.
Additional Item [othercategory]	About the add'l item.

Describes an arbitrary non-programming object, using roman text.

Example 13: Description environments

Code:

```
\begin{description}
\ItemDescribeMacro[descexamples]{\macroname} Describe the macro.
\ItemDescribeBoolean[descexamples]{booleanname} Describe the boolean.
\ItemDescribeLength[descexamples]{\lengthname} Describe the length.
\ItemDescribeKey[descexamples]{keyname} Describe the key.
\ItemDescribePackage[descexamples]{package_name} Describe the package.
\ItemDescribeClass[descexamples]{class_name} Describe the class.
\ItemDescribeFile[descexamples]{file_name} Describe the file.
\ItemDescribeProgram[descexamples]{program_name} Describe the program.
\ItemDescribeCommand[descexamples]{command_name} Describe the command.
\end{description}
```

Result:

\macroname [descexamples]	\macroname: Describe the macro.
booleanname (<i>bool</i>) [descexamples]	booleanname: Describe the boolean.
\lengthname (<i>Len</i>) [descexamples]	\lengthname: Describe the length.
keyname (<i>Key</i>) [descexamples]	keyname: Describe the key.
package_name (<i>Pkg</i>) [descexamples]	package_name: Describe the package.
class_name (<i>Cls</i>) [descexamples]	class_name: Describe the class.
file_name (<i>file</i>) [descexamples]	file_name: Describe the file.
program_name (<i>Prog</i>) [descexamples]	program_name: Describe the program.
command_name (<i>Cmd</i>) [descexamples]	command_name: Describe the command.

Uses a description environment to describe objects.

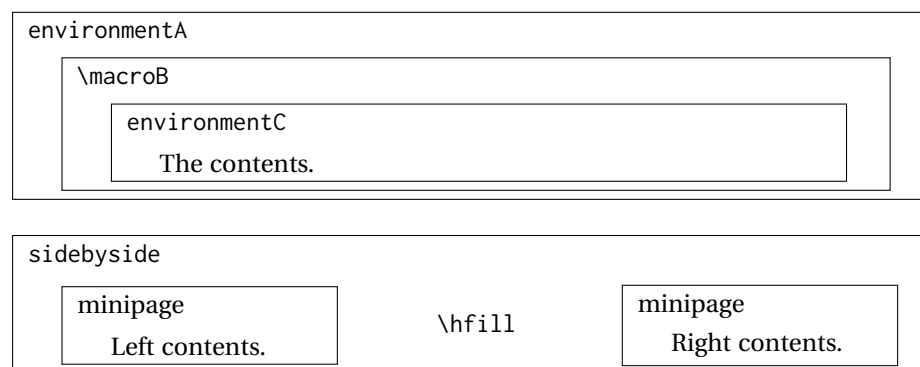
Example 14: Nesting

Code:

```
\shownesting{\env{environmentA}}{
  \shownesting{\cs{macroB}}{
    \shownesting{\env{environmentC}}{
      The contents.
    }
  }
}

\shownesting{\env{sidebyside}}{
  \shownesting[.35]{minipage}{
    Left contents.
  }
  \hfill \cs{hfill} \hfill
  \shownesting* [.35]{minipage}{
    Right contents.
  }
}
```

Result:



Note the use of the optional arguments to select less than full `\linewidth`, and the starred form for the second box to place it inline with the `\hfill` text.

Contents of the figure.

Figure 1: A Figure

Example 15: dtxexample*Code:*

```
\begin{figure}
  \centering\fbbox{Contents of the figure.}
  \caption{A Figure}\label{fig:afigure}
\end{figure}
```

*Result:**See fig. 1*

Example 15, typeset above, was created with the following code:

```
\begin{dtxexample}[See \cref{fig:afigure}]
  {\env{dtxexample}\label{ex:dtxexample}}
\begin{figure}
  \centering\fbbox{Contents of the figure.}
  \caption{A Figure}\label{fig:afigure}
\end{figure}
\end{dtxexample}
```

When the example was created:

1. The “float” of type `dtxexamplefloat` was created, with the caption “dtxexample” and the label `ex:dtxexample`, which points to example 15.
2. The code was displayed verbatim.
3. The code was written to the file `dtxexample_cut.tex`.
4. The code was `\input` from `dtxexample_cut.tex`.
5. Executing the code created the figure with caption “A Figure” and label `fig:afigure`, which points to fig. 1.
6. The cross-reference to the figure was shown on the optional display line by the optional argument to `dtxexample`.
7. The starred form of `dtxexample` was used to create the closing rule below the code, since a float was being generated and nothing followed the code inline. An unstarred version would have created an extra rule.

Example 16: fsourceverb*Code:*

```
% \begin{fsourceverb}[label=An fsourceverb example]
% \newcommand{fdosomething}[1][whattodo]{
%   doing #1
% }
% \end{fsourceverb}
```

Result:

An fsourceverb example

```
\newcommand{fdosomething}[1][whattodo]{
  doing #1
}
```

(The leading % characters would be present in the dtx source.)

Example 17: sourcedisplay*Code:*

```
\begin{sourcedisplay}
\cs{newcommand}\{dosomething\}[1][\textcolor{red}{whattodo}]{\{\}
\quad \textcolor{blue}{doing \textcolor{red}{\#1}}\}
\}
\end{sourcedisplay}
```

Result:

```
\newcommand{dosomething}[1][whattodo]{
  doing #1
}
```

Example 18: UIdisplay

Code:

```
Select:
\begin{UIdisplay}
  \textsf{Preferences $\to$ Plugins $\to$ Files $\to$ HTML}
\end{UIdisplay}
For the field
\begin{UIdisplay}
Title heading:
\end{UIdisplay}
\userentry{H1}
```

Result:

Select:

Preferences → Plugins → Files → HTML

For the field

Title heading:

Enter ⇒ **H1**

Example 19: docsidebar

Code:

Main text.

More main text.

```
\begin{docsidebar}[A title]
An aside, which may help explain something
incidental to the main text.
\end{docsidebar}
```

Additional main text.

Result:

Main text.

More main text.

A title

An aside, which may help explain something incidental to the main text.

Additional main text.

5 Usage notes

Placement of `\Describe` macros: Typically L^AT_EX macro and environment definitions are enclosed in macro and environment environments at their place in the source code. `\DescribeMacro` and `\DescribeEnv` would be used elsewhere in the manual to describe how to use the code. `\DescribeBoolean` and such might be at their place in the source code, unless they are worthy of discussion for the end-user, in which case they should be in the “User’s Manual” section of the document.¹ It may be useful to use `\DeclareBoolean` and friends both at the code location and also in the User’s Manual section.

Extra spaces: When placing multiple uses of `\Describe`, `\index`, `\margintag`, and `\watchout` macros together, care must be taken to avoid extra space in the printed text where these macros occur. A trailing percent character may be used to avoid the extra space:

```
text text text% <-- avoids extra space
\margintag{A comment.}
\index{An entry}
\index{Another entry}
more inline text
```

Unwanted vertical space: Other environments nested inside a docsidebar may produce excessive vertical space. It may be required to insert

```
\vspace*{-\baselineskip}
```

`\margintag` placement: To have the margin tag appear next to the first line of a paragraph, place the `\margintag` or `\watchout` somewhere after the first few words in the paragraph. The `\margintag` may be on its own line, and the rest of the paragraph may follow on the next line. If too many words are printed before the `\margintag`, the words may wrap to the next line before the tag occurs.

Margin tag overlap: To keep margin tags in proper alignment, use a new paragraph or multiple lines between `\margintag`, `\watchout`, or `\Declare` macros

missing tags **`\Describe` inside floats:** When these macros are used inside a float, the margin tag is suppressed (there is no margin in a float), but the index entries are still created.

¹Future versions may include `\DeclareBoolean` for use at the point where the boolean is defined, creating an index entry with a code line number, and `\DescribeBoolean` with a page number index entry for the related discussion in the User’s Manual portion of the document.

6 Code

6.1 Required packages

`makeidx (Pkg)` One of several index programs must be provided. One of several index programs must be provided.

```
1 \AtBeginDocument{
2   \IfPackageLoadedTF{makeidx}{}{
3   \IfPackageLoadedTF{splitidx}{}{
4     \RequirePackage{makeidx}
5     \makeindex
6   }}
7 }
```

`etoolbox (Pkg)` v2.6 or later for `\BeforeBeginEnvironment`, `\AfterEndEnvironment`

```
8 \RequirePackage{etoolbox}[2011/01/03]%
```

`xparse (Pkg)` Used for the examples.

```
9 \RequirePackage{xparse}
```

`calc (Pkg)` Used for `\shownesting`.

```
10 \RequirePackage{calc}
```

`xcolor (Pkg)` Used for the examples.

```
11 \RequirePackage{xcolor}
12 \definecolor{myurlcolor}{rgb}{0,0,.7}
13 \definecolor{mylinkcolor}{rgb}{.7,0,0}
```

`caption (Pkg)` Used for the examples.

```
14 \RequirePackage{caption}
```

`newfloat (Pkg)` Used for the examples.

```
15 \RequirePackage{newfloat}
```

`fancyvrb (Pkg)` Used for the examples.

```
16 \RequirePackage{fancyvrb}
```

`xstring (Pkg)` Used for `\StrSubstitute` for `\DescribeFile`.

```
17 \RequirePackage{xstring}
```

`hyperref (Pkg)` If `hyperref` is loaded, disable some macros in PDF bookmarks:
[PDF bookmarks](#)

```

18 \AtBeginDocument{
19   \IfPackageLoadedTF{hyperref}{
20     \pdfstringdefDisableCommands{%
21       \def\quad{ }%
22       \def\{ { }%
23       \def\pkg#1{#1}%
24       \def\ctr#1{#1}%
25       \def\bool#1{#1}%
26       \def\optn#1{#1}%
27       \def\env#1{#1}%
28       \def\cs#1{\textbackslash#1}%
29       \def\,{ }%
30       \def\LuaLaTeX{LuaLaTeX}%
31       \def\XeLaTeX{XeLaTeX}%
32       \def\TeX{TeX}%
33       \def\LaTeX{LaTeX}%
34       \def\LaTeXe{LaTeX2e}%
35       \def\LuaTeX{LuaTeX}%
36       \def\LuaLaTeX{LuaLaTeX}%
37       \def\XeTeX{XeTeX}%
38       \def\AmS{AMS}%
39       \def\Dash{ --- }%
40       \def\dash{ -- }%
41       \def\Slash{/}%
42       \def\prog#1{\detokenize{#1}}%
43       \def\progcode#1{#1}%
44       \def\filenm#1{\detokenize{#1}}%
45       \def\brand#1{#1}%
46       \def\acro#1{#1}%
47       \def\ODT{ODT}%
48       \def\SVG{SVG}%
49       \def\PNG{PNG}%
50       \def\GIF{GIF}%
51       \def\JPG{JPG}%
52       \def\EPS{EPS}%
53       \def\PDF{PDF}%
54       \def\DVI{DVI}%
55       \def\UTF{UTF}%
56       \def\URL{URL}%
57       \def\element#1{#1}%
58       \def\attribute#1{#1}%
59       \def\attrib#1{#1}%
60       \def\HTML{HTML}%
61       \def\HTMLfive{HTML5}%
62       \def\CSS{CSS}%
63       \def\CSSthree{CSS3}%
64       \def\EPUB{EPUB}%
65       \def\TOC{TOC}%
66       \def\LOF{LOF}%
67       \def\LOT{LOT}%
68     }
69   }% yes hyperref

```

If `hyperref` is not loaded, emulate `\hyperpage` here.

```


70   {% no hyperref
71       \newcommand*{\hyperpage}[1]{#1}
72   }
73 }
```

`pict2e` (*Pkg*)

```

74 \RequirePackage{pict2e}
75 \setlength{\unitlength}{1pt}
```

6.2 Warning sign

`\warningsign` Prints an exclamation point inside a triangle. Displays as: .

Creates a warning sign without relying on the presence of the fourier font. During copy/paste, this shows up as a simple exclamation point.

```

76 \newcommand*{\warningsign}{%
77 \begin{picture}(10,9)
78 \put(4,1){\scriptsize!}
79 \put(0,0){\line(500,866){5}}
80 \put(10,0){\line(-500,866){5}}
81 \put(0,0){\line(1,0){10}}
82 \end{picture}
83 }
```

6.3 Special character handling

The literal backslash character:

```

84 \begingroup
85 \catcode'\|=0
86 \catcode'\=12
87 |gdef|DTXD@backslash{\}
88 |endgroup
```

6.4 Patching hypdoc, splitidx, doc

If `hyperref` is disabled (by `lwarp`) then define the missing `\hdclindex`.

```

89 \IfPackageLoadedTF{doc}{
90 \IfPackageLoadedTF{hypdoc}{
91 \AddToHook{begindocument/before}[doc/hyperref]{%
92 \ifdoc@hyperref
93 \else
94 \def\hdclindex#1#2{%
```

```

95   \ifx\@nil#2\@nil\else\csname #2\expandafter\endcsname\fi%
96   }%
97   \fi
98 }
99 }{}% hypdoc loaded
100 }{}% doc loaded

```

Several changes for when hypdoc and splitidx are used together:

```

101 \AtBeginDocument{
102 \IfPackageLoadedTF{doc}{
103 \IfPackageLoadedTF{hypdoc}{
104 \IfPackageLoadedTF{splitidx}{

```

splitidx is modified to add |hdpindex{} to work with hypdoc.

```

105 \renewcommand*{\@wrsindex}[2][{}]{%
106   \ifx\relax#1\relax
107     \if@splitidx
108       \@wrsindex[idx]{#2}%
109     \else
110       \def\@tempa{#2}%
111       \if@verbindindex\@onelevel@sanitize\@tempa\fi
112       \@wrindex{\@tempa}%
113     \fi
114   \else
115 %
116 %   \def\@tempa{#2}%
117   \def\@tempa{#2\encapchar hdpindex{}}%      dtxdescribe
118 %
119   \csname index@#1@hook\endcsname
120   \expandafter\ifx\csname @wrsindex\endcsname\relax
121     \@@@wrsindex{#1}{\@tempa}{\thepage}}%
122   \else
123     \def\@tempb{\@wrsindex{#1}}%
124     \expandafter\@tempb\@tempa||\%
125   \fi
126   \endgroup
127   \@esphack
128   \fi
129 }

```

hypdoc guesses the toc level for the PDF bookmarks, but its algorithm seems to fail at \StopEventually for split indices.

“Paragraph ended before \HD@guesstoclevel was complete.”

Its guess is fixed to level 1 until \PrintChanges or \printindex.

```

130 \def\HD@guesstoclevel#1{1}

```

Patch doc’s \PrintChanges to reset the toc guess to top level.

```
131 \preto\PrintChanges{\def\HD@guesstoclevel#1{0}}
```

hypdoc adds PDF bookmarks to letter groups in the index. hypdoc must take into account multiple indexes, otherwise, followup indices will have duplicate bookmarks.

Increment the index number at the start of each index, and append the index number to the PDF bookmark.

Also, change the TOC guess to top level from now on, presuming that the indices are at the end.

```
132 \newcounter{DTXD@indexnumber}
133
134 \extendtheindex
135   {%
136     \addtocounter{DTXD@indexnumber}{1}%
137     \def\HD@guesstoclevel##1{0}%
138   }
139   {}
140   {}
141   {}
142
143 \def\HD@@@bfseries\hfil#1\hfil{%
144   \ifx\#1\%
145   \else
146     \raisebox{\baselineskip}[0pt]{%
147       \kern-\HD@margin\relax
148     % \pdfbookmark[\HD@toclevel@subindex]{#1}{HD.#1}}%
149     \pdfbookmark[\HD@toclevel@subindex]{#1}%      dtxdescribe
150     {HD.#1.\arabic{DTXD@indexnumber}}%          dtxdescribe
151     \kern\HD@margin\relax
152   }%
153 \fi
154 \hfil#1\hfil
155 }%

156 }{}% splitidx loaded
157 }{}% hypdoc loaded
158 }{}% doc loaded
159 }% AtBeginDocument
```

6.5 Gobbling comment characters

DTXD@gobble The .dtx format uses leading percent characters for code to be in the documentation only. Other classes do not.

```
160 \IfPackageLoadedTF{doc}{
161   \newcommand*{\DTXD@gobble}{2}
162 }{
163   \newcommand*{\DTXD@gobble}{0}
164 }
```

6.6 Vertical spacing

```

165 \setlength{\marginparsep}{1em}
166 \setlength{\marginparpush}{.7ex}
167
168 \setlength{\parindent}{0em}
169 \setlength{\parskip}{2ex}

```

`\IndexMin` (*Len*) From `ltxdoc`.

```

170 \ifdef{\IndexMin}
171   {\setlength{\IndexMin}{40ex}}
172   {\newlength{\IndexMin}}

```

6.7 Not `ltxdoc`: `ltxdoc` emulation

If the `ltxdoc` class is not used, some of its macros are replicated here.

```

173 \@ifclassloaded{ltxdoc}{}{
174   \def\cmd#1{\cs{\expandafter\cmd@to@cs\string#1}}
175   \def\cmd@to@cs#1#2{\char\number'#2\relax}
176   \DeclareRobustCommand\cs[1]{\texttt{\char'\#1}}
177   \providecommand\marg[1]{%
178     {\ttfamily\char'\{\}\meta{#1}\ttfamily\char'\}}
179   \providecommand\oarg[1]{%
180     {\ttfamily[]\meta{#1}\ttfamily}}
181   \providecommand\parg[1]{%
182     {\ttfamily()\meta{#1}\ttfamily}}
183   \providecommand\url{\texttt}
184 }

```

6.8 Not `doc`: `doc` emulation

If the `doc` class is not used, some of its macros are replicated here.

```

185 \AtBeginDocument{
186   \IfPackageLoadedTF{doc}{}{% not doc
187     \newenvironment*{macro}[1]{%
188       \PackageError{dtxdescribe}
189         {The 'macro' environment is only\MessageBreak
190           available when using the doc package\MessageBreak
191           with a .dtx source file}
192       {This environment only makes sense for .dtx source.}
193     }{}
194     \newenvironment*{environment}[1]{%
195       \PackageError{dtxdescribe}
196         {The 'environment' environment is only\MessageBreak
197           available when using the doc package\MessageBreak
198           with a .dtx source file}
199       {This environment only makes sense for .dtx source.}
200     }{}

```

```

201      \def\MacroFont{\fontencoding\encodingdefault
202                  \fontfamily\ttdefault
203                  \fontseries\mddefault
204                  \fontshape\updefault
205                  \small}%
206      \@ifundefined{actualchar}{\def\actualchar{@}}{}
207      \@ifundefined{quotechar}{\def\quotechar{"}}{}
208      \@ifundefined{levelchar}{\def\levelchar{!}}{}
209      \@ifundefined{encapchar}{\def\encapchar{|}}{}
210      \@ifundefined{verbatimchar}{\def\verbatimchar{+}}{}
211      \setlength\marginparpush{0pt} \setlength\marginparwidth{8pc}
212      \reversemarginpar
213      \DeclareRobustCommand\meta[1]{%
214          \ensuremath\langle
215          \ifmmode \expandafter \nfss@text \fi
216          {%
217              \meta@font@select
218              \edef\meta@hyphen@restore
219                  {\hyphenchar\the\font\the\hyphenchar\font}%
220              \hyphenchar\font\m@ne
221              \language\l@nohyphenation
222              #1\/%
223              \meta@hyphen@restore
224              }\ensuremath\rangle
225      }
226      \def\meta@font@select{\itshape}
227      }% not doc
228 }% AtBeginDocument

```

6.9 Support macros

`\PrintEnvName {<name>}` Prints an environment name.

```

229 \providecommand*\PrintEnvName{}
230 \renewcommand*\PrintEnvName[1]
231 {\strut{\scriptsize{}Env}\quad\MacroFont#1\ }

```

`\DTXD@printtype {<text>}`

Used to print the object category in the margin:

```

232 \newcommand*\DTXD@printtype[1]
233 {\raggedleft\strut{\scriptsize\sffamily#1}\quad\MacroFont}

```

`\usage {<text>}`

Allow hyperlinks in the “usage” index entries:

```

234 \IfPackageLoadedTF{doc}{%}{% not doc package
235

```

```

236 \providecommand{\usage}{}
237 \renewcommand{\usage}[1]{\textit{\hyperpage{#1}}}
238
239 }% not doc package

```

`\DTXD@origwindex` Used to bypass `hyperref` index modifications.

```

240 \let\DTXD@origwindex\@windex

```

`\DTXD@margin tag` $\{\langle category \rangle\} \{\langle name \rangle\} \{\langle margin tag \rangle\}$

Creates the margin tag for the object being described.

The category is used to sub-categorize keys into their key/value groups.

```

241 \newcommand*{\DTXD@margin tag}[3]{%
242   \ifundefined{@capt type}{% not float?
243     \leavevmode%
244     \marginpar{%
245       {%
246         \hbadness=10000%
247         \hfuzz=5em%
248         \DTXD@print type{%
249           #3% margin tag
250           \ifblank{#1}{\{ [ #1 ]}% category
251         }%
252         \texttt{#2}% name
253       }%
254     }% marginpar
255   }\}% not float?
256 }

```

`\DTXD@index` $\{\langle category \rangle\} \{\langle name \rangle\} \{\langle margin tag \rangle\} \{\langle index tag \rangle\} \{\langle main/usage \rangle\}$

Creates the index entries for the object being described, where name has no backslash or underscore.

The category is used to sub-categorize keys into their key/value groups. main prints code lines in the index, and usage prints page numbers.

```

257 \newcommand*{\DTXD@index}[5]{%

```

The `makeindex` program allows each index entry to call a macro by appending a vertical bar and a macro name to each entry. `hyperref` adds a call by `\hyperpage` to each index entry, by appending the phrase `|hyperpage` to the entry in the `.idx` file. The `doc` package uses the same mechanism to distinguish between code line entries (`|main`) and references to the use of a macro (`|usage`). The problem is that `makeindex` can only handle one macro call, but `hyperref` tries to append its `|hyperpage` to the already-existing `|usage` or `|main`.

The solution used for `dtxdescribe` is to allow `hyperref` to modify all regular index entries, but use the original definition of `\@windex` for the `\Describe_____` macros,

before `hyperref` modified it. Then, the `\usage` macro, defined above, manually adds the hyperlink.

Below, `\@bsphack` and `\@esphack` seem to be required for `\@wrindex` to work. `\ignorespaces` is used in addition because `\Declare` and `\index` entries often come in groups.

```
258 \@bsphack%
259 \begingroup%
260 \DTXD@origwrindex{%
```

Index by name:

Write the name, the formatted name, the index tag, and the category:

```
261 #2\actualchar{\protect\ttfamily#2} % name
262 (#4)% index tag
263 \ifblank{#1}{}{[#1]}%
264 \encapchar #5}%
```

Index by tag and category:

Write the tag and category as a group, under which is the name and the formatted name.

```
265 \begingroup%
266 \DTXD@origwrindex{%
267 #4:\levelchar% index tag
268 \ifblank{#1}{}{[#1]:\levelchar}%
269 #2\actualchar{\protect\ttfamily#2}% name
270 \encapchar #5}%
```

Possibly index by category and name:

```
271 \ifblank{#1}{}{% category given
272 \begingroup%
273 \DTXD@origwrindex{%
274 #1\actualchar[#1]:\levelchar% category
275 #2\actualchar{\protect\ttfamily#2} % name
276 (#4)% index tag
277 \encapchar #5}%
278 }% category given
279 % \@esphack%
280 \@esphack%
281 \ignorespaces%
282 }
```

```
\DTXD@margintagindex {<category>} {<name>} {<margin tag>} {<index tag>} {<main/usage>}
```

Creates the margin tag and the index entries. The category is used to sub-categorize keys into their key/value groups.

```
283 \newcommand*{\DTXD@marginindex}[5]{%
284 % \@bsphack%
```

The margin tag and the name:

```
285 \DTXD@margin{#1}{#2}{#3}%
```

The index entries:

```
286 \DTXD@index{#1}{#2}{#3}{#4}{#5}%
287 }
```

`\DTXD@macroname` {*<control sequence>*}

Given a control sequence such as `\name`, prints its name without the backslash.

From: <http://tex.stackexchange.com/questions/42318/removing-a-backslash-from-a-character-sequence>

```
288 \begingroup\lccode'\|='\
289 \lowercase{\endgroup\def\removebs#1{\if#1|\else#1\fi}}
290 \newcommand*{\DTXD@macroname}[1]{\expandafter\removebs\string#1}
```

`\DTXD@verbatimcmd` {*<\name>*}

While printing to the index file, prints the `\name` verbatim. From `\SpecialIndex` in the `doc` package.

```
291 \newcommand*{\DTXD@verbatimcmd}[1]{%
292 \string\verb\quotechar*\verbatimchar\string#1\verbatimchar%
293 }
```

`\DTXD@cmdmarginindex` {*<category>*} {*<name>*} {*<margin tag>*} {*<index tag>*} {*<main/usage>*}

Creates the margin tag and index entries where `name` is a `\macro`.

```
294 \newcommand*{\DTXD@cmdmarginindex}[5]{%
295 \@bsphack%
```

Create a margin tag with the name of the macro:

```
296 \@ifundefined{@capttype}{% not float?
297 \leavevmode%
298 \marginpar{%
299   {%
300     \hbadness=10000%
301     \hfuzz=5em%
302     \DTXD@printtype{%
303       #3% margin tag
304       \ifblank{#1}{\{ [ #1]}% category
305     }%

```

```

306      \cmd{#2}% name
307    }%
308 }% marginpar
309 }{}% not float?

```

Create an index entry sorted by the name without its leading backslash, followed by the macro name with the backslash, and the tag. Prepend with the category if given.

Write [category]:>name=csname (indextag)|usage

```

310 \begingroup%
311 \DTXD@origwindex{%
312 \ifblank{#1}{}{#1\actualchar[#1]:\levelchar}% category
313 \DTXD@macroname{#2}\actualchar\DTXD@verbatimcmd{#2} % name
314 (#4)% index tag
315 \encapchar #5}%

```

Create an index entry grouped by the tag, then printed and sorted by the macro name with the backslash, and the tag.

Write indextag:>[category]:>csname|usage

```

316 \begingroup%
317 \DTXD@origwindex{%
318 #4:\levelchar% index tag
319 \ifblank{#1}{}{[#1]:\levelchar}% category
320 \DTXD@verbatimcmd{#2}% name
321 \encapchar #5}%
322 \@esphack%
323 \ignorespaces%
324 }

```

6.10 doc: Key handling for object categories

If using `doc`, the optional key/value argument may also include an object category. This is supported by assigning any unknown key to be the category.

The category may be given using the key `c`, or as the default action when an unknown key is given.

```

325 \ExplSyntaxOn
326 \IfPackageLoadedTF{doc}{
327
328 \newcommand*{\DTXD@category}{}
329
330 \keys_define:nn {doc}
331 {
332   c .cs_set:Np = \DTXD@category,
333   unknown .code:n = {%
334     \renewcommand*{\DTXD@category}{\l_keys_key_str}%
335   },

```

```

336 }
337
338 }{}% doc loaded
339 \ExplSyntaxOff

```

6.11 doc: Handling \marginpar inside a float

To avoid a floats lost error, do not print margin tags if inside a float.

```

340 \IfPackageLoadedTF{doc}{
341
342 \def\@doc@describe#1#2{%
343     \ifdoc@noprint\else
344         \ifundefined{@capytype}{% not float? dtxdescribe
345             \marginpar{\raggedleft
346                 \strut
347                 \doc@providetarget
348                 \@nameuse{PrintDescribe#1}{#2}%
349                 \ifdefined{DTXD@category}{% dtxdescribe
350                     \space{\footnotesize[\mbox{DTXD@category}]]%
351                 }%
352             }
353         }{}%
354     \fi
355     \ifdoc@noindex\else
356         \@nameuse{Special#1Index}{#2}%
357     \fi
358     \@esphack
359     \endgroup
360     \ignorespaces%
361 }
362
363 }{}% doc loaded

```

6.12 doc: Handling categories and detokenizing names

`\@NewDocElement` File names and such may include underscores or other characters, so patch `doc` to neutralized while processing.

```

364 \IfPackageLoadedTF{doc}{%
365
366 \def\@NewDocElement#1#2#3{%
367     \doc@macrolikefalse
368     \doc@topleveltrue
369     \def\doc@idxtype{#3}%
370     \def\doc@idxgroup{#3s}%
371     \let\doc@printtype@empty
372     \csname keys_set:nn\endcsname{doc}{#1}%
373     \ifx\doc@printtype@empty
374         \@temptokena{}%

```

```

375 \else
376   \@temptokena\expandafter{\expandafter
377     \textnormal\expandafter{\expandafter
378       \space\expandafter
379       (\doc@printtype)}}%
380 \fi
381 \@nameedef{Print#2Name}##1{%
382   {\noexpand\MacroFont
383     \ifdoc@macrolike
384       \noexpand\string##1%           dtxdescribe
385     \else
386       \noexpand\detokenize\expandafter{##1}%   dtxdescribe
387     \fi
388     \the\@temptokena
389   }}%
390 \expandafter\let\csname PrintDescribe#2\expandafter\endcsname
391   \csname Print#2Name\endcsname
392 \edef\doc@expr{%
393   \ifdoc@macrolike
394     \noexpand\doc@createspecialmacrolikeindexes
395   \else
396     \noexpand\doc@createspecialindexes
397   \fi
398   {#2}%
399 }%
400 \expandafter\expandafter\expandafter
401 \doc@expr
402 \expandafter\expandafter\expandafter
403 {\expandafter\doc@idxtype\expandafter}\expandafter
404 {\doc@idxgroup}%
405 \doc@createdescribe{#2}%
406 \ifdoc@macrolike
407   \doc@createenv{TT}{#2}{#3}%
408 \else
409   \doc@createenv{TF}{#2}{#3}%
410 \fi
411 }

```

`\DTXDbreak` Inserts a possible line break point. Used in the index to allow line breaks before verbatim category names.

```
412 \newcommand*{\DTXDbreak}{\space\penalty200}
```

`\DTXD@printobjectname` `{\langle name w/o backslash \rangle}{\langle name \rangle}` Adds the object name to the index in verbatim. These are passed as arguments instead of directly used here because they must have their value when the index is written instead of when `\DTXD@printobjectname` is used when the index is read back.

```

413 \newcommand*{\DTXD@printobjectname}[2]{%
414   #1%
415   \actualchar%
416   \string\verb% %

```

```

417 \quotechar%
418 *%
419 \verbatimchar%
420 #2%
421 \verbatimchar%
422 }

```

`\DTXD@maybecategory` If there is a category, it is added verbatim.

```

423 \newcommand*{\DTXD@maybecategory}{%
424     \ifdefvoid{\DTXD@category}%
425         {}%
426         {

```

`\DTXDbreak` inserts a possible line break here, allowing a break if the following verbatim is too long.

```

427         \string\DTXDbreak

428         [%
429         \string\verb%
430         \noexpand\quotechar%

431         *%
432         \verbatimchar\DTXD@category\verbatimchar%
433     ]}%
434 }

```

`\DTXD@categorylevelname` The simplified name, without backslash.

```

435 \newcommand*{\DTXD@categorylevelname}

```

`\DTXD@maybecategorylevel` `{⟨object_type⟩}` If a category is assigned, index by category.

arg 1: Type of object, shown between parens (), such as macro, boolean, etc.

`\DTXD@category`: The name of the category, printed in brackets [], such as `\macroname`.

`\DTXD@categorylevelname`: The simplified name of the category, such as `macroname`.

`\@gtempa`: The name of this particular object.

```

436 \newcommand*{\DTXD@maybecategorylevel}[1]{%
437     \ifdefvoid{\DTXD@categorylevelname}%
438         {}
439         {%

```

Index:

categorylevelname=\verb!*+[category]:+>name=\verb!*+<prefix>name+<break>(type)

```

440      \index{%
441          \DTXD@categorylevelname%
442          \noexpand\actualchar%
443          \string\verb% % to fool emacs highlighting
444          \noexpand\quotechar%
445          *%
446          \noexpand\verbatimchar
447          [\DTXD@category]:%
448          \verbatimchar%
449          \noexpand\levelchar%
450      \noexpand\DTXD@printobjectname{\DTXD@gtempa@noblackslash}{\@gtempa}%
451      \ifblank{#1}{\string\DTXD@break (#1)}%
```

\the\@temptokena is not used here.

```

452      \noexpand\doc@handleencap{usage}
453      }% index
454  }%
455 }
```

\DTXD@findcategorylevelname Given \DTXD@category, create \DTXD@categorylevelname, a detokenized name without backslash.

```

456 \newcommand*{\DTXD@findcategorylevelname}{%
457     \edef\DTXD@categorylevelname{\DTXD@category}%
458     \edef\DTXD@categorylevelname{\detokenize\expandafter{\DTXD@categorylevelname}}%
459     \IfBeginWith{\DTXD@categorylevelname}{\DTXD@backslash}%
460         {\StrGobbleLeft{\DTXD@categorylevelname}{1}[\DTXD@categorylevelname]}
461     {}
462 }
```

\DTXD@gtempa@noblackslash A version without the leading backslash.

```

463 \newcommand*{\DTXD@gtempa@noblackslash}{}
```

\DTXD@findgtempa Detokenize \@gtempa and also find another version without any leading backslash.

```

464 \newcommand*{\DTXD@findgtempa}[1]{%
465     \edef\@gtempa{\detokenize{#1}}%
466     \IfEndWith{\@gtempa}{ }{\StrGobbleRight{\@gtempa}{1}[\@gtempa]}{%
467         \IfBeginWith{\@gtempa}{\DTXD@backslash}%
468             {\StrGobbleLeft{\@gtempa}{1}[\DTXD@gtempa@noblackslash]}%
469         {\edef\DTXD@gtempa@noblackslash{\@gtempa}}%
470 }
```

\doc@createspecialindexes {<1: name>} {<2: index type>} {<3: index group>}

```

471 \def\doc@createspecialindexes#1#2#3{%
472   \@temptokena{\space (#2)}%
473   \@temptokenb{#3:}%

```

Define \SpecialMain<name>Index {<1: name>}

```

474   \@nameedef{SpecialMain#1Index}##1{%

```

Find \@gtempa, the name of the object.

```

475   \noexpand\DTXD@findgtempa{##1}%

```

\@temptokena: Index (type).

\@temptokenb: Index group.

\@gtempa: Detokenized name of the particular object.

\DTXD@gtempa@nobackslash: Detokenized name without backslash.

```

476   \noexpand\@bsphack
477   \noexpand\DTXD@findcategorylevelname%           dtxdescribe

```

```

478   \ifdoc@toplevel
479 %   \noexpand\special@index{##1\noexpand\actualchar
480 %
481 %

```

Index:

name=\verb!*+name+ (type) [category]|main

```

482   \noexpand\special@index{%
483     \noexpand\DTXD@printobjectname%           dtxdescribe
484     {\noexpand\DTXD@gtempa@nobackslash}%
485     {\noexpand\@gtempa}%
486     \ifx\@nil#2\@nil\else \the\@temptokena \fi
487     \noexpand\DTXD@maybecategory%           dtxdescribe
488     \noexpand\encapchar main%
489   }%

```

Maybe add the category index entry:

```

490   \noexpand\DTXD@maybecategorylevel{#2}%       dtxdescribe
491   \fi
492 %
493 %

```

If group is not empty, index:

group:>name=\verb!*+name+ [category]|main

```

494   \ifx\@nil#3\@nil\else

```



```

495 \noexpand\special@index{%
496 \the\@temptokenb\noexpand\levelchar%
497 % ##1\noexpand\actualchar{\string\ttfamily\space##1}%
498 \noexpand\DTXD@printobjectname% dtxdescribe
499 {\noexpand\DTXD@gtempa@nobackslash}%
500 {\noexpand\@gtempa}%
501 \noexpand\DTXD@maybecategory% dtxdescribe
502 \noexpand\encapchar main%
503 }%

```

Maybe add the category index entry:

```

504 \noexpand\DTXD@maybecategorylevel{#2}% dtxdescribe
505 \fi
506 \noexpand\@esphack}%

```

Define \Special<name>Index {(*l: name*)}

```

507 \@nameedef{Special#1Index}##1{%
508 \noexpand\DTXD@findgtempa{##1}%

```

\@temptokena: Index (type).

\@temptokenb: Index group.

\@gtempa: Detokenized name of the particular object.

\DTXD@gtempa@nobackslash: Detokenized name without backslash.

```

509 \noexpand\@bsphack
510 \noexpand\DTXD@findcategorylevelname% dtxdescribe

```

If is a top level object, index:

name=\verb!*+name+ (type) [category]|usage

```

511 \ifdoc@toplevel
512 \noexpand\doc@providetarget

513 % \noexpand\index{##1\noexpand\actualchar{\string\ttfamily\space##1}%
514 \noexpand\index{%
515 \noexpand\DTXD@printobjectname% dtxdescribe
516 {\noexpand\DTXD@gtempa@nobackslash}%
517 {\noexpand\@gtempa}%
518 \ifx\@nil#2\@nil\else \the\@temptokena \fi
519 \noexpand\DTXD@maybecategory% dtxdescribe
520 \noexpand\doc@handleencap{usage}%
521 }%

```

Maybe add the category index entry:

```

522 \noexpand\DTXD@maybecategorylevel{#2}% dtxdescribe

```

```

523 \fi
524 %
525 %

```

If group is not empty, index:

group:>name=\verb!*+name+ [category]|usage

```

526 \ifx\@nil#3\@nil\else
527 \noexpand\index{%
528 \the\@temptokenb\noexpand\levelchar%
529 % ##1\noexpand\actualchar{\string\ttfamily\space##1}
530 \noexpand\DTXD@printobjectname% dtxdescribe
531 {\noexpand\DTXD@egtempa@nobackslash}%
532 {\noexpand\@egtempa}%
533 \noexpand\DTXD@maybecategory% dtxdescribe
534 \noexpand\doc@handleencap{usage}%
535 }%

```

Maybe add the category index entry:

```

536 \noexpand\DTXD@maybecategorylevel{#2}% dtxdescribe
537 \fi
538 %
539 %
540 \noexpand\@esphack}}

```

cc@createspecialmacrolikeindexes {\langle 1: name \rangle} {\langle 2: index type \rangle} {\langle 3: index group \rangle}

```

541 \def\doc@createspecialmacrolikeindexes#1#2#3{%
542 \@temptokena{\space (#2)}%
543 \@temptokenb{#3:}%

```

Define \Code<name>Index {\langle 1: main *or* usage \rangle} {\langle 2: name \rangle}

```

544 \@nameedef{Code#1Index}##1##2{%
545 \noexpand\DTXD@findgtempa{##2}%

```

\@temptokena: Index type.

\@temptokenb: Index group.

\@gtempa: Detokenized name of the particular object.

\DTXD@egtempa@nobackslash: Detokenized name without backslash.

```

546 \noexpand\@bsphack
547 \noexpand\DTXD@findcategorylevelname% dtxdescribe
548 \noexpand\ifdoc@noindex\noexpand\else

```

If a top level object, index:

name=\verb!*+\name+ (type) [category]|<main or usage>

```

549 \ifdoc@toplevel
550 \noexpand\special@index{%
551     \noexpand\DTXD@printobjectname%           dtxdescribe
552     {\noexpand\DTXD@gttempa@nobackslash}%
553     {\noexpand\@gttempa}%
554     \ifx\@nil#2\@nil\else \the\@temptokena \fi
555     \noexpand\DTXD@maybecategory%           dtxdescribe
556     \noexpand\encapchar ##1%
557 }%
```

Maybe add the category index entry:

```

558 \noexpand\DTXD@maybecategorylevel{#2}%       dtxdescribe
559 \fi
```

If group is not empty, index:

group:>name=\verb!*+\name+ [category]|<main or usage>

```

560 \ifx\@nil#3\@nil\else
561 \noexpand\special@index{%
562     \the\@temptokenb\noexpand\levelchar
563     \noexpand\DTXD@printobjectname%           dtxdescribe
564     {\noexpand\DTXD@gttempa@nobackslash}%
565     {\noexpand\@gttempa}%
566     \noexpand\DTXD@maybecategory%           dtxdescribe
567     \noexpand\encapchar ##1%
568 }%
```

Maybe add the category index entry:

```

569 \noexpand\DTXD@maybecategorylevel{#2}%       dtxdescribe
570 \fi
571 \noexpand\fi
572 \noexpand\@esphack}%
```

Define \SpecialMain<name>Index {<I: name>}

```

573 \@nameedef{SpecialMain#1Index}##1{%
574     \expandafter\noexpand\csname Code#1Index\endcsname
575     {main}{##1}}%
```

Define \Special<name>Index {<I: name>}

```

576 \@nameedef{Special#1Index}##1{%
577     \noexpand\DTXD@findgttempa{##1}}%
```

\@temptokena: Index type.

\@temptokenb: Index group.

\@gtempa: Detokenized name of the particular object.

\DTXD@gtempa@nobackslash: Detokenized name without backslash.

```
578 \noexpand\@bsphack
579 \noexpand\DTXD@findcategorylevelname% dtxdescribe

580 \noexpand\ifdoc@noindex\noexpand\else
```

If a top level object, index:

name=\verb!*+\name+ (type) [category]|usage

```
581 \ifdoc@toplevel
582 \noexpand\doc@providetarget
583 \noexpand\index{%
584     \noexpand\DTXD@printobjectname% dtxdescribe
585     {\noexpand\DTXD@gtempa@nobackslash}%
586     {\noexpand\@gtempa}%
587     \ifx\@nil#2\@nil\else \the\@temptokena \fi
588     \noexpand\DTXD@maybecategory% dtxdescribe
589     \noexpand\doc@handleencap{usage}%
590 }%
```

Maybe add the category index entry:

```
591 \noexpand\DTXD@maybecategorylevel{#2}% dtxdescribe
592 \fi
```

If group is not empty, index:

group:>name=\verb!*+\name+ [category]|usage

```
593 \ifx\@nil#3\@nil\else
594 \noexpand\index{%
595     \the\@temptokenb\noexpand\levelchar
596     \noexpand\DTXD@printobjectname% dtxdescribe
597     {\noexpand\DTXD@gtempa@nobackslash}%
598     {\noexpand\@gtempa}%
599     \noexpand\DTXD@maybecategory% dtxdescribe
600     \noexpand\doc@handleencap{usage}%
601 }%
```

Maybe add the category index entry:

```
602 \noexpand\DTXD@maybecategorylevel{#2}% dtxdescribe
603 \fi
604 \noexpand\fi
605 \noexpand\@esphack}}
606
607 }{}% doc loaded
```

6.13 Not doc: \DescribeMacro and \DescribeEnvironment

`\DescribeMacro` [*<category>*] {*<\name>*}

Redefined to allow hyperlinked index entries and an optional category:

```
608 \IfPackageLoadedTF{doc}{\not doc}
609
610 \providecommand*\DescribeMacro{}
611 \renewcommand*\DescribeMacro[2][\not doc]{%
612 \@bsphack%
```

Create the margin tag with the macro's name:

```
613 \@ifundefined{@capttype}{\not float}
614 \leavevmode%
615 \marginpar{%
616   {%
617     \hbadness=10000%
618     \hfuzz=5em%
619     \raggedleft%
620     \ifblank{#1}{\scriptsize\textsf{[#1]}}}% category
621     \cmd{#2}% name
622   }%
623 }% marginpar
624 }{\not float}
```

Write the index sorted by the name without the backslash, followed by the actual name with the backslash. Append the category if given.

Write `name=csname>[category]|usage`

```
625 \begingroup%
626 \DTXD@origwindex{%
627   \DTXD@macroname{#2}\actualchar\DTXD@verbatimcmd{#2}% name
628   \ifblank{#1}{\levelchar{#1}}}% category
629   \encapchar usage%
630 }%
```

Only if a category was given:

```
631 \ifblank{#1}%
632 {}% no category
633 {% category given
634 % Again, and prepend the category:
635 %
636 % Write category=[category]:>name=csname\verb+|usage+
637 %   \begin{macrocode}
638   \begingroup%
639   \DTXD@origwindex{%
640     #1\actualchar{#1}:\levelchar%
641     \DTXD@macroname{#2}\actualchar\DTXD@verbatimcmd{#2}%
```

```

642 \encapchar usage}%
643 }% category given
644 \@esphack%
645 \ignorespaces%
646 }
647
648 }% not doc

```

`\DescribeEnv` [*category*] [*environment name*]

Redefined to allow hyperlinked index entries:

```

649 \IfPackageLoadedTF{doc}{\}% not doc
650
651 \providecommand*\DescribeEnv{}
652 \renewcommand*\DescribeEnv[2][]
653 {\DTXD@margintagindex{#1}{#2}{Env}{environment}{usage}}
654
655 }% not doc

```

6.14 New `\Describe. . . macros`

`\DTXD@filemarginparindex` [*category*] [*name*] [*margin tag*] [*index tag*] [*main/usage*]

The name may have underscores.

```

656 \IfPackageLoadedTF{doc}{\}% not doc
657 \newcommand*\DTXD@filename{}
658
659 \newcommand*\DTXD@filemarginparindex[5]{%

```

Create a detokenized version of the filename...

```

660 \renewcommand{\DTXD@filename}{\detokenize{#2}}%

```

... then replace any underscores with a detokenized `_`, which will print as an underscore when read back from the index file:

```

661 \StrSubstitute{\DTXD@filename}{\detokenize{_\}}{\detokenize{\_}}[\DTXD@filename]%

```

The original filename is printed in the margin. Any underscore characters have already been disabled by the `\catcode` change.

```

662 \DTXD@margintag{#1}{#2}{#3}%

```

The detokenized and sanitized version is sent to the index file:

```

663 \DTXD@index{#1}{\DTXD@filename}{#3}{#4}{#5}%

```

End the group with the disabled underscore, and clean up the extra space from the `\catcode` command:

```
664 \endgroup%
665 \ignorespaces%
666 }
667 }% not doc
```

`\DescribeMacro` Redefine with new definitions.

`\DescribeEnvironment` Redefine with new definitions.

```
668 \IfPackageLoadedTF{doc}{% doc
669
670 \RenewDocElement[macrolike = true ,
671                 idxtype   = ,
672                 idxgroup  = ,
673                 printtype =
674                 ]{Macro}{macro}
675
676 \RenewDocElement[macrolike = false ,
677                 idxtype   = env. ,
678                 idxgroup  = environments ,
679                 printtype = \textit{env.}
680                 ]{Env}{environment}
681
682 }{}% doc
```

`\DescribeFile` $\{\langle name \rangle\}$

The underscore character is temporarily disabled, then the name is passed directly to `\DTXD@DescribeFile`.

```
683 \IfPackageLoadedTF{doc}{% doc
684
685 \NewDocElement[
686   macrolike=false,
687   toplevel=true,
688   idxtype=file,
689   idxgroup=Files,
690   printtype=\textit{file}
691 ]{File}{file}
692
693 }{}% not doc
694
695 \newcommand*{\DTXD@DescribeFile}[2][{}]{%
696   \DTXD@filemarginparindex{#1}{#2}{File}{file}{usage}%
697 }
698
699 \newcommand*{\DescribeFile}{%
700   \begingroup\catcode'\_ =12 \DTXD@DescribeFile%
701 }
```

```
702
703 }% not doc
```

`\DescribeProgram {⟨name⟩}`

The underscore character is temporarily disabled, then the name is passed directly to `\DTXD@DescribeProgram`.

```
704 \IfPackageLoadedTF{doc}{% doc
705
706 \NewDocElement[
707   macrolike=false,
708   toplevel=true,
709   idxtype=program,
710   idxgroup=Programs,
711   printtype=\textit{Prog}
712 ]{Program}{program}
713
714 }{% not doc
715
716 \newcommand*{\DTXD@DescribeProgram}[2][{}]{%
717   \DTXD@filemarginparindex{#1}{#2}{Prog}{program}{usage}%
718 }
719
720 \newcommand*{\DescribeProgram}{%
721   \begingroup\catcode'\_ =12 \DTXD@DescribeProgram%
722 }
723 }% not doc
724
```

`\DescribeCommand {⟨name⟩}`

The underscore character is temporarily disabled, then the name is passed directly to `\DTXD@DescribeCommand`.

```
725 \IfPackageLoadedTF{doc}{% doc
726
727 \NewDocElement[
728   macrolike=false,
729   toplevel=true,
730   idxtype=command,
731   idxgroup=Commands,
732   printtype=\textit{Cmd}
733 ]{Command}{command}
734
735 }{% not doc
736
737 \newcommand*{\DTXD@DescribeCommand}[2][{}]{%
738   \DTXD@filemarginparindex{#1}{#2}{Cmd}{command}{usage}%
739 }
740
741 \newcommand*{\DescribeCommand}{%
742   \begingroup\catcode'\_ =12 \DTXD@DescribeCommand%
```



```

743 }
744
745 }% not doc

```

`\DescribePackage {⟨name⟩}`

The underscore character is temporarily disabled, then the name is passed directly to `\DTXD@DescribePackage`.

```

746 \IfPackageLoadedTF{doc}{% doc
747
748 \NewDocElement[
749   macrolike=false,
750   toplevel=true,
751   idxtype=package,
752   idxgroup=Packages,
753   printtype=\textit{Pkg}
754 ]{Package}{package}
755
756 }{% not doc
757
758 \newcommand*{\DTXD@DescribePackage}[2][{}]{%
759   \DTXD@filemarginparindex{#1}{#2}{Pkg}{package}{usage}%
760 }
761
762 \newcommand*{\DescribePackage}{%
763   \begingroup\catcode'\_ =12 \DTXD@DescribePackage%
764 }
765
766 }% not doc
767

```

`\DescribeClass {⟨name⟩}`

The underscore character is temporarily disabled, then the name is passed directly to `\DTXD@DescribeClass`.

```

768 \IfPackageLoadedTF{doc}{% doc
769
770 \NewDocElement[
771   macrolike=false,
772   toplevel=true,
773   idxtype=class,
774   idxgroup=Classes,
775   printtype=\textit{Cls}
776 ]{Class}{class}
777
778 }{% not doc
779
780 \newcommand*{\DTXD@DescribeClass}[2][{}]{%
781   \DTXD@filemarginparindex{#1}{#2}{Cls}{class}{usage}%
782 }

```

```

783
784 \newcommand*{\DescribeClass}{%
785     \begingroup\catcode'\_ =12 \DTXD@DescribeClass%
786 }
787
788 }% not doc

```

`\DescribeOption` [*category*] {*name*}

```

789 \IfPackageLoadedTF{doc}{% doc
790
791 \NewDocElement[
792     macrolike=false,
793     toplevel=true,
794     idxtype=option,
795     idxgroup=Options,
796     printtype=\textit{Opt}
797 ]{Option}{option}
798
799 }{% not doc
800
801 \newcommand*{\DescribeOption}[2][]
802 { \DTXD@margintagindex{#1}{#2}{Opt}{option}{usage}}
803
804 }% not doc

```

`\DescribeArgument` [*category*] {*name*}

The category may be used to categorize arguments by their macro or environment name.

```

805 \IfPackageLoadedTF{doc}{% doc
806
807 \NewDocElement[
808     macrolike=false,
809     toplevel=true,
810     idxtype=argument,
811     idxgroup=Arguments,
812     printtype=\textit{Arg}
813 ]{Argument}{argument}
814
815 }{% not doc
816
817 \newcommand*{\DescribeArgument}[2][]
818 { \DTXD@margintagindex{#1}{#2}{Arg}{argument}{usage}}
819
820 }% not doc

```

`\DescribeBoolean` [*category*] {*name*}

```

821 \IfPackageLoadedTF{doc}{% doc
822

```

```

823 \NewDocElement[
824     macrolike=false,
825     toplevel=true,
826     idxtype=boolean,
827     idxgroup=Booleans,
828     printtype=\textit{bool}
829 ]{Boolean}{boolenv}
830
831 }{% not doc
832
833 \newcommand*{\DescribeBoolean}[2][]
834 { \DTXD@margintagindex{#1}{#2}{Bool}{boolean}{usage}}
835
836 }{% not doc

```

`\DescribeLength` [*category*] [*name*]

```

837 \IfPackageLoadedTF{doc}{% doc
838
839 \NewDocElement[
840     macrolike=true,
841     toplevel=true,
842     idxtype=length,
843     idxgroup=Lengths,
844     printtype=\textit{Len}
845 ]{Length}{length}
846
847 }{% not doc
848
849 \newcommand*{\DescribeLength}[2][]
850 { \DTXD@cmdmargintagindex{#1}{#2}{Len}{length}{usage}}
851
852 }{% not doc

```

`\DescribeCounter` [*category*] [*name*]

```

853 \IfPackageLoadedTF{doc}{% doc
854
855 \NewDocElement[
856     macrolike=false,
857     toplevel=true,
858     idxtype=counter,
859     idxgroup=Counters,
860     printtype=\textit{Ctr}
861 ]{Counter}{counter}
862
863 }{% not doc
864
865 \newcommand*{\DescribeCounter}[2][]
866 { \DTXD@margintagindex{#1}{#2}{Ctr}{counter}{usage}}
867
868 }{% not doc

```

`\DescribeHook [<category>] {<name>}`

```

869 \IfPackageLoadedTF{doc}{% doc
870
871 \NewDocElement[
872   macrolike=true,
873   toplevel=true,
874   idxtype=hook,
875   idxgroup=Hooks,
876   printtype=\textit{Hook}
877 ]{Hook}{hook}
878
879 }{% not doc
880
881 \newcommand*{\DescribeHook}[2][]
882   {\DTXD@margin@index{#1}{#2}{Hook}{hook}{usage}}
883
884 }{% not doc

```

`\DescribeSocket [<category>] {<name>}`

```

885 \IfPackageLoadedTF{doc}{% doc
886
887 \NewDocElement[
888   macrolike=false,
889   toplevel=true,
890   idxtype=socket,
891   idxgroup=Sockets,
892   printtype=\textit{Socket}
893 ]{Socket}{socket}
894
895 }{% not doc
896
897 \newcommand*{\DescribeSocket}[2][]
898   {\DTXD@margin@index{#1}{#2}{Socket}{socket}{usage}}
899
900 }{% not doc

```

`\DescribePlug [<category>] {<name>}`

```

901 \IfPackageLoadedTF{doc}{% doc
902
903 \NewDocElement[
904   macrolike=false,
905   toplevel=true,
906   idxtype=plug,
907   idxgroup=Plugs,
908   printtype=\textit{Plug}
909 ]{Plug}{plug}
910
911 }{% not doc
912
913 \newcommand*{\DescribePlug}[2][]

```

```

914     {\DTXD@margintagindex{#1}{#2}{Plug}{plug}{usage}}
915
916 }% not doc

```

`\DescribeKey` [*category*] [*name*]

The category may be used to categorize keys by their kev/value group.

```

917 \IfPackageLoadedTF{doc}{% doc
918
919 \NewDocElement[
920     macrolike=false,
921     toplevel=true,
922     idxtype=key,
923     idxgroup=Keys,
924     printtype=\textit{Key}
925 ]{Key}{key}
926
927 }{% not doc
928
929 \newcommand*{\DescribeKey}[2][]
930 {\DTXD@margintagindex{#1}{#2}{Key}{key}{usage}}
931
932 }% not doc

```

`\DescribeObject` [*category*] [*name*]

May be used to describe an arbitrary piece of code. Creates a margin tag and index entries with `\ttfamily`.

```

933 \IfPackageLoadedTF{doc}{% doc
934
935 \NewDocElement[
936     macrolike=false,
937     toplevel=true,
938     idxtype=object,
939     idxgroup=Objects,
940     printtype=,
941 ]{Object}{object}
942
943 }{% not doc
944
945 \newcommand*{\DescribeObject}[2][]{%
946 \ifundefined{@capttype}{% not float?
947     \@bsphack%
948     \leavevmode%
949     \marginpar{%
950         \hbadness=10000%
951         \hfuzz=5em%
952         \raggedleft%
953         \ifblank{#1}{}{\raggedleft{\scriptsize[#1]} }
954         \texttt{#2}%
955     }%

```

```

956 }{}% not float?
957 \ifblank{#1}%
958 {%
959     \begingroup%
960     \DTXD@origwindex{%
961         #2\actualchar{\protect\ttfamily#2}%
962         \encapchar usage%
963     }%
964 }%
965 {%
966     \begingroup%
967     \DTXD@origwindex{%
968         #2\actualchar{\protect\ttfamily#2} [#1]%
969         \encapchar usage%
970     }%
971     \begingroup%
972     \DTXD@origwindex{%
973         #1\actualchar[#1]:\levelchar#2\actualchar{\protect\ttfamily#2}%
974         \encapchar usage%
975     }%
976 }%
977 \@esphack%
978 \ignorespaces%
979 }
980
981 }{}% not doc

```

`\DescribeOther` [*category*] [*name*]

May be used to describe an arbitrary non-programming object. Creates a margin tag and index entries with roman type.

```

982 \IfPackageLoadedTF{doc}{% doc
983
984 \NewDocElement[
985     macrolike=false,
986     toplevel=true,
987     idxtype=other,
988     idxgroup=Other,
989     printtype=,
990 ]{Other}{other}
991
992 }{}% not doc
993
994 \newcommand*{\DescribeOther}[2][{}]{%
995 \@ifundefined{@capttype}{% not float?
996     \@bsphack%
997     \leavevmode%
998     \marginpar{%
999         \hbadness=10000%
1000         \hfuzz=5em%
1001         \raggedleft%
1002         \ifblank{#1}{}{\raggedleft\scriptsize[#1]} }%

```

```

1003      #2%
1004    }%
1005 }{}% not float?
1006 \ifblank{#1}%
1007 {%
1008   \begingroup%
1009   \DTXD@origwindex{#2\encapchar usage}%
1010 }%
1011 {%
1012   \begingroup%
1013   \DTXD@origwindex{#2 [#1]\encapchar usage}%
1014   \begingroup%
1015   \DTXD@origwindex{#1\actualchar[#1]:\levelchar#2\encapchar usage}%
1016 }%
1017 \@espack%
1018 \ignorespaces%
1019 }
1020
1021 }% not doc

```

6.15 \DescribeDefault

`\DescribeDefaultcolor` The color of the margin tag used to show the default value.

```
1022 \newcommand*{\DescribeDefaultcolor}{green!50!black}
```

`\DescribeDefault` {*<value>*}

Creates a colored margin tag showing the booleandefault value.

```

1023 \newcommand{\DescribeDefault}[1]{%
1024   \margintag{%
1025     \footnotesize%
1026     \textcolor{\DescribeDefaultcolor}{%
1027       Default: \texttt{#1}%
1028     }%
1029   }%
1030 }

```

6.16 \ItemDescribeMacro, etc.

The following are for use inside a description.

`\ItemDescribeMacro` [*<category>*] {*<\name>*}

```

1031 \newcommand{\ItemDescribeMacro}[2][ ]{%
1032   \item[\cmd{#2}:]%
1033   \setlength{\parskip}{1.5ex}%
1034   \DescribeMacro[#1]{#2}%
1035 }

```

`\ItemDescribeEnv [<category>] {<name>}`

```
1036 \newcommand{\ItemDescribeEnv}[2][{}]{%
1037     \item[\env{#2}:]%
1038     \setlength{\parskip}{1.5ex}%
1039     \DescribeEnv[#1]{#2}%
1040 }
```

`\ItemDescribeArgument [<category>] {<argument>}`

```
1041 \newcommand{\ItemDescribeArgument}[2][{}]{%
1042     \item[\texttt{#2}:]%
1043     \setlength{\parskip}{1.5ex}%
1044     \DescribeArgument[#1]{#2}%
1045 }
```

`\ItemDescribeBoolean [<category>] {<name>}`

```
1046 \newcommand{\ItemDescribeBoolean}[2][{}]{%
1047     \item[\texttt{#2}:]%
1048     \setlength{\parskip}{1.5ex}%
1049     \DescribeBoolean[#1]{#2}%
1050 }
```

`\ItemDescribeLength [<category>] {<name>}`

```
1051 \newcommand{\ItemDescribeLength}[2][{}]{%
1052     \item[\cmd{#2}:]%
1053     \setlength{\parskip}{1.5ex}%
1054     \DescribeLength[#1]{#2}%
1055 }
```

`\ItemDescribeCounter [<category>] {<name>}`

```
1056 \newcommand{\ItemDescribeCounter}[2][{}]{%
1057     \item[\texttt{#2}:]%
1058     \setlength{\parskip}{1.5ex}%
1059     \DescribeCounter[#1]{#2}%
1060 }
```

`\ItemDescribeHook [<category>] {<name>}`

```
1061 \newcommand{\ItemDescribeHook}[2][{}]{%
1062     \item[\texttt{#2}:]%
1063     \setlength{\parskip}{1.5ex}%
1064     \DescribeHook[#1]{#2}%
1065 }
```

`\ItemDescribeSocket [<category>] {<name>}`


```

1066 \newcommand{\ItemDescribeSocket}[2][{}]{%
1067   \item[\texttt{#2}:]%
1068   \setlength{\parskip}{1.5ex}%
1069   \DescribeSocket[#1]{#2}%
1070 }

```

`\ItemDescribePlug` [*category*] {*name*}

```

1071 \newcommand{\ItemDescribePlug}[2][{}]{%
1072   \item[\texttt{#2}:]%
1073   \setlength{\parskip}{1.5ex}%
1074   \DescribePlug[#1]{#2}%
1075 }

```

`\ItemDescribeKey` [*category*] {*name*}

```

1076 \newcommand{\ItemDescribeKey}[2][{}]{%
1077   \item[\texttt{#2}:]%
1078   \setlength{\parskip}{1.5ex}%
1079   \DescribeKey[#1]{#2}%
1080 }

```

`\ItemDescribePackage` [*category*] {*name*}

```

1081 \newcommand{\DTXD@ItemDescribePackage}[2][{}]{%
1082   \item[\texttt{#2}:]%
1083   \setlength{\parskip}{1.5ex}%
1084   \DescribePackage[#1]{#2}%
1085   \endgroup%
1086 }
1087
1088 \newcommand{\ItemDescribePackage}{%
1089   \begingroup\catcode'\_ =12 \DTXD@ItemDescribePackage%
1090 }

```

`\ItemDescribeClass` [*category*] {*name*}

```

1091 \newcommand{\DTXD@ItemDescribeClass}[2][{}]{%
1092   \item[\texttt{#2}:]%
1093   \setlength{\parskip}{1.5ex}%
1094   \DescribeClass[#1]{#2}%
1095   \endgroup%
1096 }
1097
1098 \newcommand{\ItemDescribeClass}{%
1099   \begingroup\catcode'\_ =12 \DTXD@ItemDescribeClass%
1100 }

```

`\ItemDescribeOption` [*category*] {*name*}

```

1101 \newcommand{\ItemDescribeOption}[2][]{%
1102   \item[\texttt{#2}:]%
1103   \setlength{\parskip}{1.5ex}%
1104   \DescribeOption[#1]{#2}%
1105 }

```

`\ItemDescribeFile` [*category*] {*name*}

```

1106 \newcommand{\DTXD@ItemDescribeFile}[2][]{%
1107   \item[\texttt{#2}:]%
1108   \setlength{\parskip}{1.5ex}%
1109   \DescribeFile[#1]{#2}%
1110   \endgroup%
1111 }
1112
1113 \newcommand{\ItemDescribeFile}{%
1114   \begingroup\catcode'\_ =12 \DTXD@ItemDescribeFile%
1115 }

```

`\ItemDescribeProgram` [*category*] {*name*}

```

1116 \newcommand{\DTXD@ItemDescribeProgram}[2][]{%
1117   \item[\texttt{#2}:]%
1118   \setlength{\parskip}{1.5ex}%
1119   \DescribeProgram[#1]{#2}%
1120   \endgroup%
1121 }
1122
1123 \newcommand{\ItemDescribeProgram}{%
1124   \begingroup\catcode'\_ =12 \DTXD@ItemDescribeProgram%
1125 }

```

`\ItemDescribeCommand` [*category*] {*name*}

```

1126 \newcommand{\DTXD@ItemDescribeCommand}[2][]{%
1127   \item[\texttt{#2}:]%
1128   \setlength{\parskip}{1.5ex}%
1129   \DescribeCommand[#1]{#2}%
1130   \endgroup%
1131 }
1132
1133 \newcommand{\ItemDescribeCommand}{%
1134   \begingroup\catcode'\_ =12 \DTXD@ItemDescribeCommand%
1135 }

```

`\ItemDescribeObject` [*category*] {*name*}

```

1136 \newcommand{\ItemDescribeObject}[2][]{%
1137   \item[\texttt{#2}:]%
1138   \setlength{\parskip}{1.5ex}%
1139   \DescribeObject[#1]{#2}%
1140 }

```

`\ItemDescribeOther` [*<category>*] {*<name>*}

```
1141 \newcommand{\ItemDescribeOther}[2][]{%
1142     \item[\texttt{#2}:]%
1143     \setlength{\parskip}{1.5ex}%
1144     \DescribeOther[#1]{#2}%
1145 }
```

6.17 `\margintag`, `\watchout`

`\margintagcolor` The color of the `\margintag`.

```
1146 \newcommand*{\margintagcolor}{blue!70!black}
```

`\margintag` {*<text>*}

Prints a colored margin tag.

```
1147 \newcommand{\margintag}[1]{%
1148 \ifundefined{@capttype}{% not float?
1149 \marginpar{\raggedleft\textcolor{\margintagcolor}{#1}}%
1150 \ignorespaces%
1151 }{% not float?
1152 }
```

`\watchoutcolor` The color of the `\watchout`.

```
1153 \newcommand*{\watchoutcolor}{red!50!black}
```

`\watchout` [*<text>*]

Prints a warning sign and optional text.

```
1154 \newcommand{\watchout}[1][]{%
1155 \ifundefined{@capttype}{% not float?
1156     \marginpar{%
1157         \raggedleft%
1158         \textcolor{\watchoutcolor}{\warningsign\normalsize\quad#1}%
1159     }%
1160     \ignorespaces%
1161 }{% not float?
1162 }
```

6.18 Nesting

Shows a box enclosing a label for the container, and the container's contents. May be nested.

`\shownesting` [*$\langle fraction of \linewidth \rangle$*] [*$\langle container \rangle$*] [*$\langle contents \rangle$*]

```

1163 \NewDocumentCommand{\shownesting}{s O{1} m m}{
1164   \IfBooleanF{#1}{
1165     \par\smallskip
1166   }
1167   \fbox{
1168     \begin{minipage}{#2\linewidth-2em}
1169       \hbadness=10000\relax%
1170       #3\par\smallskip
1171       \hspace{1em}
1172       \begin{minipage}{\linewidth-1.5em}
1173         #4
1174       \end{minipage}
1175     \end{minipage}
1176   }
1177 }
```

6.19 The dtxexample environment

Also see example 15 on page 28.

`dtxexample_cut.tex` (*file*) Used to store the `\input` example code.

`DTXD@examplerulecolor` [*color*] The color of the middle rule in the `dtxexample`.

```
1178 \definecolor{DTXD@examplerulecolor}{rgb}{.9,.9,.9}
```

`\dtxexamplecodename` The text name of the code section.

```
1179 \newcommand*{\dtxexamplecodename}{Code:}
```

`\dtxexampleresultname` The text name of the result section.

```
1180 \newcommand*{\dtxexampleresultname}{Result:}
```

`dtxexample` (*env.*) * [*$\langle notes/cross-references \rangle$*] [*$\langle caption \& label \rangle$*]

Reads the code listing as a verbatim input using the `fancybox` package, then displays the code listing as a verbatim output, and also executes the code and displays the result. A title caption is specified, along with optional cross-referencing commands or notes to refer to the results. The unstarred version places the code inside a `minipage`, forbidding a page break in the middle of the code listing. The starred version does not use a `minipage`. This is required when the code is too large to fit on a single page.

```

1181 \NewDocumentEnvironment{dtxexample}{s +O{} m}
1182 {% start dtxexample
```

Copy the environment's contents to the file `dtxexample_cut.tex`:

```
1183 \VerbatimOut[gobble=\DTXD@gobble,tabsize=4]{dtxexample_cut.tex}%
1184 }% start dtxexample
```

When the environment closes:

```
1185 {% end dtxexample
```

Finish the verbatim output:

```
1186 \endVerbatimOut
1187 \par
1188 \addvspace{\bigskipamount}
```

If unstarred, typeset the example in a minipage, else use a float:

```
1189 \IfBooleanTF{#1}%
1190   {% minipage
1191     \minipage{\linewidth}%
1192     \captionsetup{type=dtxexamplefloat}%
1193   }%
1194   {% float
1195     \begin{dtxexamplefloat}%
1196   }%

1197 \hrule\medskip
1198 \caption{#3}
```

Typeset the contents as verbatim:

```
1199 \textcolor{DTXD@examplerulecolor}{\smallskip\hrule}
1200 \smallskip
1201 {\scriptsize\itshape\dtxexamplecodename}
1202 \VerbatimInput[tabsize=4]{dtxexample_cut.tex}
1203 \unskip
1204 \textcolor{DTXD@examplerulecolor}{\hrule}
1205 \smallskip
1206 {\scriptsize\itshape\dtxexampleresultname}
1207
```

Possible add the optional cross-references or notes:

```
1208 \ifstrempy{#2}
1209 {}
1210 {\itshape\small #2}}
```

If unstarred, close the float or `\minipage`.

```
1211 \IfBooleanTF{#1}%
1212   {\endminipage}%
1213   {\end{dtxexamplefloat}}%
1214 } % end dtxexample
```

Outside of the environment's scope, input the example to generate its output and labels:

```
1215 \AfterEndEnvironment{dtxexample}
1216 {%
```

Execute the code:

```
1217 \par\unskip\input{dtxexample_cut.tex}%
```

Closing rule::

```
1218 \medskip\hrule%
1219 }
```

`\DeclareFloatingEnvironment` A new float type for the examples.
[dtxexamplefloat]

```
1220 \DeclareFloatingEnvironment[
1221 fileext=lox,
1222 listname={List of Examples},
1223 name=Example,
1224 placement=hbpb
1225 ]{dtxexamplefloat}
```

`\captionsetup` [dtxexamplefloat] Caption setup for the examples.

```
1226 \captionsetup*[dtxexamplefloat]{
1227 format=hang,
1228 font=bf,
1229 justification=raggedright,
1230 singlelinecheck=false,
1231 skip=0pt,
1232 position=top,
1233 }
```

`\crefname` [dtxexamplefloat] Name for `\cleveref`. `\crefname` here is required for documents not using the doc class:

```
1234 %\AddToHook{begindocument/after}{% Before .aux file is loaded.
1235 \AtBeginDocument{
1236   \ifdef{\crefname}{
1237     \crefname{dtxexamplefloat}{example}{examples}
1238   }{}
1239 }
```

6.20 noindmacro and noindenvironment

Similar to macro and environment, but not indexed.

`noindmacro` (*env*) {*<name>*}

```

1240 \newenvironment{noindmacro}[1]
1241 {
1242     \setlength{\parskip}{\marginparpush}
1243     \leavevmode\par\DTXD@margintag{}\cmd{#1}}{}
1244 }
1245 {\unskip}

```

`noindenvironment (env.)` $\{\langle name \rangle\}$

```

1246 \newenvironment{noindenvironment}[1]
1247 {
1248     \setlength{\parskip}{\marginparpush}
1249     \leavevmode\par\DTXD@margintag{}\#1}{Env}
1250 }
1251 {\unskip}

```

6.21 sourcedisplay, UIdisplay, docsidebar

For use in a `sourcedisplay`:

`\fquad` Forces a quad indent.

```

1252 \newcommand*\fquad{\hspace*{1em}}

```

`\fqquad` Forces a double-quad indent.

```

1253 \newcommand*\fqquad{\hspace*{2em}}

```

`\fqqqquad` Forces a triple-quad indent.

```

1254 \newcommand*\fqqqquad{\hspace*{3em}}

```

`sourceverb (env.)` To typeset a block of source code, verbatim.

```

1255 \DefineVerbatimEnvironment{sourceverb}{Verbatim}
1256     {gobble=\DTXD@gobble,tabsize=4,xleftmargin=2em}
1257 \BeforeBeginEnvironment{sourceverb}{\vspace*{-.5\parskip}}

```

`fsourceverb (env.)` To typeset a framed block of source code, verbatim.

```

1258     \DefineVerbatimEnvironment{fsourceverb}{Verbatim}
1259     {gobble=\DTXD@gobble,tabsize=4,xleftmargin=2em,frame=lines}
1260 \BeforeBeginEnvironment{fsourceverb}{\vspace*{-.5\parskip}}

```

`sourcedisplay (env.)` To typeset a block of source code, allowing direct formatting.

```

1261 \newenvironment{sourcedisplay}
1262 {
1263     \leavevmode
1264     \par
1265     \fqquad\minipage{\linewidth-4em}
1266     \ttfamily
1267 }
1268 {%
1269     \endminipage
1270     \par
1271 }

```

`UIDisplay` (*env*) To typeset a user interface display.

```

1272 \newenvironment{UIDisplay}
1273 {
1274     \leavevmode
1275     \par
1276     \fqquad\minipage{\linewidth-4em}
1277     \sfamily\bfseries
1278 }
1279 {
1280     \endminipage
1281     \par
1282 }

```

`\userentryname` Text to tell the user to enter the following item.

```

1283 \newcommand*{\userentryname}{Enter~$\rightarrow$}

```

`\userentry` {*⟨text to enter⟩*}

Typesets text to be entered by the users.

```

1284 \newcommand{\userentry}[1]{%
1285 \par
1286 \fqquad%
1287 \begin{minipage}{\linewidth-2em}
1288     {\footnotesize \userentryname}\quad\cmds{#1}
1289 \end{minipage}
1290 \par
1291 }

```

`docsidebar` (*env*) To typeset a sidebar in the documentation.

```

1292 \newenvironment{docsidebar}[1][[]
1293 {%
1294     \quote\unskip\medskip
1295     \setlength{\parskip}{1.5ex}%
1296     \ifblank{#1}{}{\textit{#1}\newline}%
1297     \rule[.5\bigskipamount]{\linewidth}{.4pt}%

```



```

1298 \newline%
1299 }
1300 {%
1301 \leavevmode\par
1302 \rule[\bigskipamount]{\linewidth}{.4pt}
1303 \endquote\unskip
1304 }

```

6.22 Formatted objects

Macros to format references to various kinds of objects.

6.22.1 L^AT_EX objects

`\pkg {⟨name⟩}` Also useable for class names.

```

1305 \providerobustcmd*{\pkg}[1]{\mbox{\textsf{#1}}}

```

`\cs {⟨csname⟩}` From `ltxdoc`.

```

1306 \providerobustcmd*{\cs}[1]{\texttt{\char'\#1}}

```

`\env {⟨name⟩}`

```

1307 \providerobustcmd*{\env}[1]{\mbox{\texttt{#1}}}

```

`\marg {⟨argument⟩}` From `ltxdoc`.

```

1308 \providecommand\marg[1]{%
1309 {\ttfamily\char'\{}meta{#1}{\ttfamily\char'\}}}

```

`\oarg [⟨argument⟩]` From `ltxdoc`.

```

1310 \providecommand\oarg[1]{%
1311 {\ttfamily[]meta{#1}{\ttfamily[]}}

```

`\parg (⟨argument⟩)` From `ltxdoc`.

```

1312 \providecommand\parg[1]{%
1313 {\ttfamily{}meta{#1}{\ttfamily{}}}

```

`\ctr {⟨name⟩}`

```

1314 \providerobustcmd*{\ctr}[1]{\mbox{\texttt{#1}}}

```

`\bool` { $\langle name \rangle$ }

1315 \providerobustcmd*{\bool}[1]{\mbox{\texttt{#1}}}

`\optn` { $\langle name \rangle$ }

1316 \providerobustcmd*{\optn}[1]{\mbox{\texttt{#1}}}

`\TOC`

1317 \providerobustcmd*{\TOC}{\acro{TOC}}

`\LOF`

1318 \providerobustcmd*{\LOF}{\acro{LOF}}

`\LOT`

1319 \providerobustcmd*{\LOT}{\acro{LOT}}

6.22.2 Programs and commands

`\cmds` { $\langle commands to print \rangle$ } No processing is provided for special characters.

1320 \providerobustcmd*{\cmds}[1]{\mbox{\textbf{\texttt{#1}}}}

`\progcode` { $\langle code to print \rangle$ } No processing is provided for special characters.

1321 \providerobustcmd*{\progcode}[1]{\mbox{\texttt{#1}}}

`\prog` { $\langle program name \rangle$ } Underscores are allowed.

```
1322 \newcommand*{\DTXD@prog}[1]{%
1323   \mbox{\textsf{\textsl{\detokenize{#1}}}}%
1324   \endgroup%
1325 }
1326
1327 \providerobustcmd*{\prog}{%
1328   \begingroup%
1329   \catcode'\_ =12%
1330   \DTXD@prog%
1331 }
```

`\filenm` { $\langle file name \rangle$ } Underscores are allowed.

1332 \newcommand*{\DTXD@filenm}[1]{%

```
1333 \mbox{\texttt{\detokenize{#1}}}%  
1334 \endgroup%  
1335 }  
1336  
1337 \providerobustcmd*{\filenm}{%  
1338 \beginingroup%  
1339 \catcode'\_ =12%  
1340 \DTXD@filenm%  
1341 }
```

\UI General user-interface text.

```
1342 \providerobustcmd*{\UI}[1]{\textbf{\textsf{#1}}}
```

6.22.3 File types

\ODT

```
1343 \providerobustcmd*{\ODT}{\acro{ODT}}
```

\SVG

```
1344 \providerobustcmd*{\SVG}{\acro{SVG}}
```

\PNG

```
1345 \providerobustcmd*{\PNG}{\acro{PNG}}
```

\GIF

```
1346 \providerobustcmd*{\GIF}{\acro{GIF}}
```

\JPG

```
1347 \providerobustcmd*{\JPG}{\acro{JPG}}
```

\EPS

```
1348 \providerobustcmd*{\EPS}{\acro{EPS}}
```

\PDF

```
1349 \providerobustcmd*{\PDF}{\acro{PDF}}
```

\DVI

```
1350 \providerobustcmd*{\DVI}{\acro{DVI}}
```

6.22.4 Internet

\UTF

1351 \providerobustcmd*{\UTF}{\acro{UTF}}

\URL

1352 \providerobustcmd*{\URL}{\acro{URL}}

\element {<name>}

1353 \providerobustcmd*{\element}[1]{\texttt{<#1>}}

\attribute {<name>}

\attrib {<name>}

Each of these is “provided”, and any prior meaning will be unchanged. In particular, LuaTeX uses \attribute, so its meaning is unchanged if using LuaTeX.

1354 \providerobustcmd*{\attrib}[1]{\mbox{\texttt{#1}}}

1355

1356 \providerobustcmd*{\attribute}[1]{\mbox{\texttt{#1}}}

\HTML

1357 \providerobustcmd*{\HTML}{\acro{HTML}}

\HTMLfive

1358 \providerobustcmd*{\HTMLfive}{\HTML\textsc{5}}

\CSS

1359 \providerobustcmd*{\CSS}{\acro{CSS}}

\CSSthree

1360 \providerobustcmd*{\CSSthree}{\CSS\textsc{3}}

\EPUB

1361 \providerobustcmd*{\EPUB}{\acro{EPUB}}

6.22.5 Specific programs

`\TikZ`

1362 `\providerobustcmd*{\TikZ}{\emph{k}Z}`

`\CTAN`

1363 `\providerobustcmd*{\CTAN}{\acro{CTAN}}`

`\TDS`

1364 `\providerobustcmd*{\TDS}{\acro{TDS}}`

`\MathML`

1365 `\providerobustcmd*{\MathML}{\Math\acro{ML}}`

`\MathJax`

1366 `\providerobustcmd*{\MathJax}{\brand{MathJax}}`

6.22.6 Acronyms, brand names, trademarks

`\brand {<name>}`

1367 `\providerobustcmd*{\brand}[1]{\textsc{#1}}`

`\acro {<acronym>}`

1368 `\providerobustcmd*{\acro}[1]{\textsc{\lowercase{#1}}}`

`\supregistered` Superscript trademark symbol.

1369 `\providerobustcmd*{\supregistered}{\textregistered}`

6.23 Logos

`\dviTeX` DVI T_EX

1370 `\providerobustcmd*{\dviTeX}{\mbox{\DVI\, \TeX}}`

`\dviLaTeX` DVI L^AT_EX

1371 `\providerobustcmd*{\dviLaTeX}{\mbox{\DVI\, \LaTeX}}`

`\pdfTeX` PDF T_EX

```
1372 \providerobustcmd*{\pdfTeX}{\mbox{\PDF\,\TeX}}
```

`\pdfLaTeX` PDF L^AT_EX

```
1373 \providerobustcmd*{\pdfLaTeX}{\mbox{\PDF\,\LaTeX}}
```

`\LuaTeX` LuaT_EX

```
1374 \providerobustcmd*{\LuaTeX}{\mbox{\Lua\TeX}}
```

`\LuaLaTeX` LuaL^AT_EX

```
1375 \providerobustcmd*{\LuaLaTeX}{\mbox{\Lua\LaTeX}}
```

`\XeTeX` X_ET_EX, X_EL^AT_EX

`\XeLaTeX`

```
1376 \providerobustcmd*{\XeTeXrevE}
1377   {\hspace{-.1667em}\raisebox{-.5ex}{E}\hspace{-.125em}}
1378
1379 \AtBeginDocument{
1380 \IfPackageLoadedTF{graphics}{
1381   \renewrobustcmd*{\XeTeXrevE}
1382     {\hspace{-.1667em}\raisebox{-.5ex}{\reflectbox{E}}\hspace{-.125em}}
1383 }{}
1384 }
1385
1386 \providerobustcmd*{\XeTeX}{\mbox{X\XeTeXrevE\TeX}}
1387 \providerobustcmd*{\XeLaTeX}{\mbox{X\XeTeXrevE\LaTeX}}
```

`\AmS` A_MS

```
1388 \providerobustcmd*{\AmS}{%
1389   \leavevmode\hbox{$\mathcal A\kern-.2em\lower.376ex%
1390   \hbox{$\mathcal M$}\kern-.2em\mathcal S$}%
1391 }
```

`\LyX` LyX

```
1392 \providerobustcmd*{\LyX}{\textsf{LyX}}
```

`\BibTeX` B_IB_T_EX

```
1393 \providerobustcmd*{\BibTeX}{\mbox{B\textsc{ib}\TeX}}
```

`\MakeIndex` *MakeIndex*

```
1394 \providerobustcmd*{\MakeIndex}{\prog{MakeIndex}}
```

`\ConTeXt` `ConTeXt`

```
1395 \providerobustcmd*{\ConTeXt}{\mbox{Con\TeX{t}}}
```

`\MiKTeX` `MiKTeX`

```
1396 \providerobustcmd*{\MiKTeX}{\mbox{MiK\TeX{t}}}
```

6.24 Dashes and slashes

`\thinspace` A breakable thin skip.

```
1397 \DeclareRobustCommand{\thinspace}{\hspace 0.16667em\relax}
```

`\endash` An endash: –

```
1398 \def\endash{-}
```

`\emdash` An emdash: —

```
1399 \def\emdash{-}
```

`\thinbrspace` A thin space which allows a line break.

```
1400 \newcommand{\thinbrspace}{%
1401   \hspace{.16667em}\penalty\exhyphenpenalty\hspace{0pt}%
1402 }
```

`\thinthinbrspace` A thin space which allows a line break.

```
1403 \newcommand{\thinthinbrspace}{%
1404   \hspace{.08333em}\penalty\exhyphenpenalty\hspace{0pt}%
1405 }
```

`\Dash` An unbreakable thin space, emdash, and breakable thin space.

```
1406 \newrobustcmd{\Dash}{\unskip\thinspace\emdash\thinbrspace}
```

`\dash` An unbreakable thin space, endash, and breakable thin space.

```
1407 \newrobustcmd{\dash}{\unskip\thinspace\endash\thinbrspace}
```

`\Slash` An unbreakable very thin space, a slash, and a breakable thin space.

```
1408 \newrobustcmd{\Slash}{\unskip\hspace{.08333em}/\thinthinbrspace}
```

7 Compiling dtxdescribe

To compile the dtxdescribe package:

```
Enter ⇒ pdflatex dtxdescribe.ins
```

To compile the dtxdescribe documentation

```
Enter ⇒ pdflatex dtxdescribe.dtx
```

(Several times)

```
Enter ⇒ makeindex -s gglo.ist -o dtxdescribe.gls dtxdescribe.glo
```

```
Enter ⇒ makeindex -s gind.ist dtxdescribe
```

```
Enter ⇒ pdflatex dtxdescribe.dtx
```

(Several times)

Change History and Index

Change History

v0.10		
General: 2016/12/08 Initial ver	1	
v0.11		
General: 2018/03/30	1	
\DTXD@cmdmargintagindex: Index		
tag no longer plural.	43	
\DTXD@index: Index tag no longer		
plural.	41	
\watchout: Changed to \raggedleft.	67	
v1.00		
General: 2019/01/11	1	
Added formatted objects.	73	
Added logos.	77	
Cut file name changed to		
dtxexample_cut.tex	68	
\DescribeClass: Fix: Allow		
underscore.	57	
\DescribeDefault: Added.	63	
\DescribeDefaultcolor: Added.	63	
\DescribeMacro: Sans tag font.	53	
\DescribePackage: Fix: Allow		
underscore.	57	
docsidebar: Added.	72	
\DTXD@cmdmargintagindex: Sans tag		
font.	42	
\DTXD@filemarginparindex: Fix: File		
category.	54	
\DTXD@printtype: Sans tag font.	39	
\dtxexamplecodename: Added.	68	
\dtxexampleresultname: Added.	68	
\fqquad: Added.	71	
\fquad: Added.	71	
\quad: Added.	71	
fsourceverb: Added.	71	
\ItemDescribeArgument: Added.	64	
\ItemDescribeBoolean: Added.	64	
\ItemDescribeClass: Added.	65	
\ItemDescribeCommand: Added.	66	
\ItemDescribeCounter: Added.	64	
\ItemDescribeEnv: Added.	64	
\ItemDescribeFile: Added.	66	
\ItemDescribeKey: Added.	65	
\ItemDescribeLength: Added.	64	
\ItemDescribeMacro: Added.	63	
\ItemDescribeObject: Added.	66	
\ItemDescribeOption: Added.	65	
\ItemDescribeOther: Added.	67	
\ItemDescribePackage: Added.	65	
\ItemDescribeProgram: Added.	66	
\margintag: Uses \margintacolor.	67	
\margintagcolor: Added.	67	
noindentenvironment: Added.	71	
noindentmacro: Added.	70	
sourcedisplay: Added.	71	
sourceverb: Added.	71	
UIDisplay: Added.	72	
\userentry: Added.	72	
\userentryname: Added.	72	
\watchoutcolor: Added.	67	
v1.01		
General: 2019/03/22	1	
Sanitize PDF bookmarks.	33	
\DescribeMacro: Put margin tag		
category in brackets.	53	
\DescribeObject: Put margin tag		
category in brackets.	61	
\DescribeOther: Put margin tag		
category in brackets.	62	
\DTXD@cmdmargintagindex: Put		
margin tag category in brackets.	42	
\DTXD@margintag: Put margin tag		
category in brackets.	40	
sourcedisplay: Reduced width.	71	
UIDisplay: Reduced width.	72	
v1.02		
General: 2019/07/16	1	
Fix if not doc package.	38	
Fix if not hyperref package.	33	
Fix if not ltxdoc class.	33, 38	
\DescribeEnv: Fix if not ltxdoc class.	54	
\DescribeMacro: Fix if not ltxdoc		
class.	53	
\DescribeObject: \raggedleft		
margin par.	61	

Added \ignorespaces.	61	\pdfTeX: Added.	78
\DescribeOther: \raggedleft		\shownesting: Added.	68
margin par.	62	v1.04	
Added \ignorespaces.	62	General: 2022/02/01	1
dtxexample: Fix if not doc package.	68	Corrected copyright date.	1
\PrintEnvName: Fix if not ltxdoc		v1.05	
class.	39	General: 2022/09/08	1
DTXD@gobble: Fix if not doc package.	37	\TikZ: Renamed from \tikz, cap Z.	77
\usage: Fix if not ltxdoc class.	39	v1.06	
v1.03		General: 2022/12/07	1
General: 2022/02/01	1	Fixed for updated doc.	35, 36, 43, 44
No longer requires xifthen.	33	\@NewDocElement: Fixed for updated	
\attrib: For LuaTeX.	76	doc.	44
\cs: Provided.	73	v1.07	
\DescribeHook: Added.	60	General: 2023/01/03	1
\DescribeMacro: Reduce hbox		Improved PDF bookmarks.	36, 37
warnings.	53	\doc@createspecialmacrolikeindexes:	
\DescribeObject: Reduce hbox		Added category names for	
warnings.	61	macros.	50
\DescribeOther: Reduce hbox		\DTXD@findgtempa: Detokenized	
warnings.	62	names.	47
\DTXD@cmdmargintagindex: Reduce		\DTXD@maybecategory: \verb*.	46
hbox warnings.	42	Improved line breaks.	46
\DTXD@margintag: Reduce hbox		\DTXD@maybecategorylevel: \verb*.	46
warnings.	40	Fix occasional crash.	47
\dviLaTeX: Added.	77	\DTXDbreak: Improved line breaks.	45
\dviTeX: Added.	77	v1.08	
\ItemDescribeHook: Added.	64	General: 2024/01/08	1
\marg: Provided.	73	Updated contact info.	1
\MathJax: Added.	77	\DescribePlug: Added.	60
\oarg: Provided.	73	\DescribeSocket: Added.	60
\parg: Provided.	73	\ItemDescribePlug: Added.	65
\pdfLaTeX: Added.	78	\ItemDescribeSocket: Added.	64

Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols	another_big_file.txt (file) [bigfiles]	22
\@NewDocElement	Arguments:	364
[H] (argument) [figure]	[H] [figure]	24
[M] (argument) [figure]	[M] [figure]	24
	bold [\mymacro]	24
A	\attrib	17, 1354
\acro	\attribute	17, 1354
Additional_Item (other) [othercategory]		
.		
\AmS	B	
	Before_myenvironment (other)	25

\crefname	70	Files:	
\DeclareFloatingEnvironment ...	70	another_big_file.txt [bigfiles]	22
\dtxexampleresultname	14, 1180	dtxexample_cut.tex	68
\DVI	16, 1350	file_name [descexamples]	26
\dviLaTeX	17, 1371	lone_file.txt	22
\dviTeX	17, 1370	really_big_file.txt [bigfiles] .	22
E		firstkey (key) [groupofkeys]	23
\element	16, 1353	\fqquad	15, 1254
\emdash	18, 1399	\fquad	15, 1253
\endash	18, 1398	\quad	15, 1252
\env	15, 1307	fsourceverb (env.)	1258
environment (env.)	9	fsourceverb (env.)	15
environments:		G	
docsidebar	1292	\GIF	16, 1346
docsidebar	15	group of objects	20
dtxexample	1181	[groupofkeys]:	
dtxexample	13	firstkey (key)	23
environment	9	secondkey (key)	23
fsourceverb	1258	H	
fsourceverb	15	\hookname (hook)	21
macro	9	Hooks:	
myenvironment	20	\hookname	21
noindentenvironment	1246	para/begin	21
noindentenvironment	14	\HTML	17, 1357
noindmacro	1240	\HTMLfive	17, 1358
noindmacro	14	hyperref (package)	33
otherenvironment	20	I	
sourcedisplay	1261	index	
sourcedisplay	15	by group	20
sourceverb	1255	\IndexMin (length)	38
sourceverb	14	\ItemDescribeArgument	12, 1041
UIdisplay	1272	\ItemDescribeBoolean	12, 1046
UIdisplay	15	\ItemDescribeClass	12, 1091
\EPS	16, 1348	\ItemDescribeCommand	12, 1126
\EPUB	17, 1361	\ItemDescribeCounter	12, 1056
etoolbox (package)	33	\ItemDescribeEnv	12, 1036
[examples]:		\ItemDescribeFile	12, 1106
sample_class (class)	22	\ItemDescribeHook	12, 1061
sampleboolean (boolean)	21	\ItemDescribeKey	12, 1076
samplecounter (counter)	21	\ItemDescribeLength	12, 1051
samplekey (key)	23	\ItemDescribeMacro	12, 1031
sampleoption (option)	22	\ItemDescribeObject	12, 1136
samplepackage (package)	22	\ItemDescribeOption	12, 1101
sampletwokey (key)	23	\ItemDescribeOther	12, 1141
F		\ItemDescribePackage	12, 1081
fancyvrb (package)	33	\ItemDescribePlug	12, 1071
[figure]:		\ItemDescribeProgram	12, 1116
[H] (argument)	24	\ItemDescribeSocket	12, 1066
[M] (argument)	24	J	
file_name (file) [descexamples]	26	\JPG	16, 1347
\filenm	16, 1332		

K		randomobject	25
keyname (key) [descexamples]	26	somecolor [color]	25
Keys:		\ODT	16, 1343
firstkey [groupofkeys]	23	Options:	
keyname [descexamples]	26	sampleoption [examples]	22
lonekey	23	\optn	16, 1316
samplekey [examples]	23	OS_command (command)	22
sampletwokey [examples]	23	Other_Item (other) [othercategory] ..	25
secondkey [groupofkeys]	23	Other:	
L		Additional_Item [othercategory] ..	25
\lengthname (length) [descexamples] .	26	Before_myenvironment	25
Lengths:		license_agreement	25
\IndexMin	38	Other_Item [othercategory]	25
\lengthname [descexamples]	26	[othercategory]:	
\photowidth [photograph]	21	Additional_Item (other)	25
license_agreement (other)	25	Other_Item (other)	25
listofdtexamplefloats	14	othercolor (object) [color]	25
\LOF	16, 1318	otherenvironment (env.)	20
lone_file.txt (file)	22	P	
lonekey (key)	23	package_name (package) [descexamples]	26
\LOT	16, 1319	Packages:	
\LuaLaTeX	18, 1375	calc	33
\LuaTeX	18, 1374	caption	33
\LyX	18, 1392	etoolbox	33
M		fancyvrb	33
macro (env.)	9	hyperref	33
\macroname [descexamples]	26	makeidx	33
makeidx (package)	33	newfloat	33
\MakeIndex	18, 1394	package_name [descexamples] ...	26
\marg	9, 15, 1308	pict2e	35
margin tag missing	32	samplepackage [examples]	22
\margintag	13, 1147	splitidx	33
\margintagcolor	13, 1146	xcolor	33
\MathJax	17, 1366	xparse	33
\MathML	17, 1365	xstring	33
\MiKTeX	18, 1396	para/begin (hook)	21
myenvironment (env.)	20	\parg	9, 15, 1312
[\mymacro]:		\PDF	16, 1349
bold (argument)	24	\pdfLaTeX	17, 1373
\mymacro	19	\pdfTeX	17, 1372
N		\photocaptionsetup [photograph] ...	19
newfloat (package)	33	[photograph]:	
noindenvironment (env.)	1246	\cphotonameref	19
noindenvironment (env.)	14	\DeclareFloatingPhoto	19
noindmacro (env.)	1240	\photocaptionsetup	19
noindmacro (env.)	14	\photowidth (length)	21
O		\photowidth (length) [photograph] ...	21
\oarg	9, 15, 1310	pict2e (package)	35
Objects:		\pkg	15, 1305
DTXD@examplerulecolor [color] ..	68	plugname (plug)	21
othercolor [color]	25	Plugs:	
		plugname	21
		\PNG	16, 1345

\PrintEnvName	229	splitidx (package)	33
\prog	16, 1322	\supregistered	17, 1369
\progcode	16, 1321	\SVG	16, 1344
program_name (program)	22		
program_name (program) [descexamples]		T	
.....	26	\TDS	17, 1364
Programs:		\thinbrspace	18, 1400
program_name	22	\thinspace	18, 1397
program_name [descexamples] ...	26	\thinthinbrspace	18, 1403
		\TikZ	17, 1362
R		\TOC	16, 1317
randomobject (object)	25		
really_big_file.txt (file) [bigfiles]	22	U	
		\UI	16, 1342
S		UIdisplay (env.)	1272
sample_class (class) [examples]	22	UIdisplay (env.)	15
sampleboolean (boolean) [examples] .	21	\URL	16, 1352
samplecounter (counter) [examples] .	21	\usage	234
samplekey (key) [examples]	23	\userentry	15, 1284
sampleoption (option) [examples] ...	22	\userentryname	15, 1283
samplepackage (package) [examples] .	22	\UTF	16, 1351
sampletwokey (key) [examples]	23		
secondkey (key) [groupofkeys]	23	W	
\shownesting	13, 1163	\warningsign	76
\Slash	18, 1408	\watchout	13, 1154
socketname (socket)	21	\watchoutcolor	13, 1153
Sockets:			
socketname	21	X	
somecolor (object) [color]	25	xcolor (package)	33
sourcedisplay (env.)	1261	\XeLaTeX	18, 1376
sourcedisplay (env.)	15	\XeTeX	18, 1376
sourceverb (env.)	1255	xparse (package)	33
sourceverb (env.)	14	xstring (package)	33