

\textbf{Abstract}

\textit{lipsum} is a \LaTeX\ package that produces dummy text to be used in test documents or examples. The paragraphs are taken with permission from https://www.lipsum.com/, thanks to James Wilson for this work. Furthermore, the following people contributed to \textit{lipsum} by suggesting improvements, correcting bugs or finding typos in the documentation: Florent Chervet, Ulrike Fischer, Vincent Belaïche, Enrico Gregorio, Frank Mittelbach, Karl Hagen. Please, file bug reports, typos in the documentation or feature requests as an issue on https://github.com/PhelypeOleinik/lipsum/issues.

1 Introduction

To load the package, write

\begin{verbatim}
\usepackage{lipsum}
\end{verbatim}

in the preamble of your document. Probably the most important macro provided by this package is $\textit{\textbackslash lipsum}$, which typesets the \textit{Lorem ipsum} paragraphs. The first optional argument allows to specify the range of the paragraphs. For example, $\textit{\textbackslash lipsum}[4-57]$ typesets the paragraphs 4 to 57 and accordingly, $\textit{\textbackslash lipsum}[23]$ typesets the 23$^{\text{rd}}$ paragraph. Using $\textit{\textbackslash lipsum}$ without its optional argument typesets the paragraphs 1–7 of \textit{Lorem ipsum}...

As of version 2.0, $\textit{\textbackslash lipsum}$ has a second optional argument which allows selecting a range of sentences from the paragraphs. To get the sentences four to eight from paragraphs three to nine, use $\textit{\textbackslash lipsum}[3-9][4-8]$. The sentences are counted from the first sentence of the first selected paragraph. In the previous example, sentence number 1 is the first sentence of paragraph number 3.

1.1 Foreword to Version 2.4

Version 2.4 received another almost complete rewrite focussing on the internal structure of the package, and some minor fixes (see the CHANGELOG for more details).

The package now ships with a new dummy text, in pseudo-Czech, provided by Ondřej Macek. To select this text, load the package with $\textit{\usepackage[text=lipsum-cs]{lipsum}}$ or use $\textit{\setlipsum[text=lipsum-cs]}$.

The dummy texts now have a language metadata which is used to select the proper hyphenation patterns to the dummy text. For compatibility with old documents you can load \textit{lipsum} with $\textit{\usepackage[auto-lang=false]{lipsum}}$ or, as above, use $\textit{\setlipsum[auto-lang=false]}$.

Finally, as demonstrated above, a new macro $\textit{\setlipsum}$ was added to change package options anywhere in the document, so you may change, for example, the dummy text printed by $\textit{\textbackslash lipsum}$.
on-the-fly by using \texttt{\setlipsum[text=(name)]} (see section 4) for a list of available texts). In general, a key-val syntax was added which will eventually replace the command-based syntax for package settings. For the time being, both versions are available.

1.2 Foreword to Version 2.0

Version 2.0 of \texttt{lipsum} is a complete (well, nearly complete) rewrite of the code in expl3 syntax. I have never used expl3 before and thus the code might be too complicated, might use wrong or badly chosen data types or weird function names. I am happy to receive comments on this.

Due the complete rewrite, some internals have changed which might impact older documents. Since, however, I guess that \texttt{lipsum} is not used for documents with true, important, content, I think potentially breaking up old documents is not a big issue here. The changes are:

- The package option \texttt{nopar} now uses a \texttt{\space} as terminator, instead of \texttt{\relax}.
- The commands \texttt{\UnpackLipsum} and \texttt{\UnpackLipsum*} are no longer available. The effect of \texttt{\UnpackLipsum} now is default for \texttt{\unpacklipsum} (or \texttt{\unpacklipsum*}, depending on the package option). The effect of \texttt{\UnpackLipsum*} can be mimicked by using \texttt{\LipsumProtect{⟨command⟩}}, as in the following example:

  \begin{verbatim}
\documentclass{article}
\usepackage{lipsum,xcolor}
\newcommand\foo{}
\SetLipsumParListItemEnd{\LipsumProtect{\foo}}

\begin{document}
\renewcommand\foo{\color{.!75!red}}
  \lipsumexp
\newcounter{mycnt}\setcounter{mycnt}{1}
\renewcommand\foo{\stepcounter{mycnt}}
\lipsumexp
\end{document}
  \end{verbatim}

- The internal macros \texttt{\lips@i}, \texttt{\lips@ii}, \texttt{\lips@iii}, ..., \texttt{\lips@c} are no longer available.
- All other internal macros (with one exception) are no longer available, too.

1.3 Foreword to version 2.2

As of version 2.2, \texttt{lipsum} provides a simple interface to define other texts to be used as output of the \texttt{\lipsum}-family of commands. This was heavily inspired by an issue raised by \texttt{svenper} on github\textsuperscript{1}. However, the implementation of this interface might not match the needs of everyone who wants to provide a dummy text in another language. Comments and suggestions on this are very welcome.

Please note that the documentation still only refers to the \textit{Lorem ipsum} text.

\textsuperscript{1}https://github.com/patta42/lipsum/issues/13
Usage

lipsum was intended to quickly provide a way to fill a page or two to analyze the page layout. While it has grown in the meanwhile and now provides some more advanced features, it still is only intended to quickly provide text. If you want more features, look at the blindtext-package.

2.1 Package Options

lipsum outputs a range of paragraphs taken from the Lorem ipsum... dummy text. The package options control mainly the behaviour of the \lipsum and \unpacklipsum commands, and can be set at load-time with \usepackage\{option\}\{lipsum\}, or later in the document by using \setlipsum\{option\}.

no\par = \{boolean\} \hspace{1cm} \text{(default: false)}
Changes the initial default separator between each paragraph of \lipsum from \par to \space, and the other way around for \lipsum*.

text = \{name\} \hspace{1cm} \text{(default: lipsum)}
Selects the dummy text \{name\} that is used by \lipsum and \unpacklipsum (see section 4).

language = \{lang\} \hspace{1cm} \text{(default: latin)}
Sets the language to be used by \lipsum to typeset the currently active dummy text (see section 3.2). Changing the dummy text with the text option will also change the current language.

auto-lang = \{boolean\} \hspace{1cm} \text{(default: true)}
Turns on/off automatic language switching. This changed since version 2.3, in which this option (didn’t exist thus) was false by default. See section 3.2 for more details.

default-range = \{p1-pf\} \hspace{1cm} \text{(default: 1-7)}
Sets the default range of paragraphs produced by \lipsum when no optional argument is provided. The value to default-range obeys the \{range\} syntax described in section 3.1. If no value is given to default-range (that is, \setlipsum\{default-range\}), then the default is reset to 1-7.

Besides these options, there are still ones that can be passed to the package to influence the paragraph and sentence separators and other such things. These options are detailed in section 3.3.

2.2 User Commands

\lipsum \lipsum\{\{par range\}\{sentence range\}\}
\lipsum outputs the \{par range\} from the currently active dummy text. If \{par range\} is not given or is empty, the default-range (initially 1-7) is output. If a \{sentence range\} is given, the selected paragraphs are split into sentences, numbered starting from 1, and the specified range of sentences is taken out from those paragraphs. If the \{*\} version is used, a different set of separators is inserted around the paragraphs or sentences.

\lipsum changes the active language to that of the dummy text for typesetting, so the proper hyphenation patterns are used. See section 3.2. Section 3.1 explains the syntax of ranges, and section 3.3 explains the separators added around the pieces of text.

\text{\footnotesize \hspace{1cm} \text{2https://groups.google.com/d/topic/de.comp.text.tex/oPeLDjkrlfk}}
\unpacklipsum \lipsumexp
\lipsumexp
\unpacklipsum selects the paragraphs and/or sentences exactly as described for \lipsum, but instead of outputting them, it saves the selected text in the \lipsumexp macro. Additionally, \unpacklipsum ... \lipsumexp is not completely equivalent to \lipsum because it doesn’t change languages as \lipsum does.

\setlipsum \setlipsum{\{key-val list\}}
Applies the \{key-val list\} of options to the package. The options are described in section 2.1 and in section 3.3.

2.3 Other commands
These commands exist for necessity or backwards comatibility, and should normally not be needed in user documents.

\SetLipsumText \SetLipsumDefault{\{name\}}
Loads the dummy text \{name\} (see section 4). This command does the same as option text, but it is kept for backwards compatibility.

\SetLipsumDefault \SetLipsumDefault{\{range\}}
Sets the default range for \lipsum and \unpacklipsum. This command does the same as option default-range, but it is kept for backwards compatibility.

3 General remarks on behaviour
Here are some topics that are general considerations about the behaviour of \lipsum and its commands. These are technicalities that most end users don’t care too much about, unless you are trying to do something beyond the usual “print me some dummy text”.

3.1 Syntax of paragraph and sentence ranges
A \{range\} argument can either be blank, a single integer, or a proper integer range. If the \{range\} argument is blank, the commands behave as if the argument was not given at all. For example, \lipsum[] behaves exactly like \lipsum and outputs the default paragraph range. Note that \lipsum[] [2-5] does not behave as \lipsum[2-5], but behaves as \lipsum[1-7] [2-5] (assuming \text{default=range=1-7}), because the default value is then taken for the first argument. If the \{range\} argument is an integer, then only a single paragraph/sentence is selected.

If the argument contains a - (ASCII 45), it is interpreted as a \textit{proper} range \(n_{i} - n_{f}\). In a proper range, if \(n_{i}\) is blank, it is taken to be the start of the possible range, and in the same way, if \(n_{f}\) is empty it is taken to be the end of the possible range. That is, \lipsum[-9] is the same as \lipsum[1-9], and \lipsum[5-] is the same (assuming the standard 150-paragraph dummy text) as \lipsum[5-150], and similarly, \lipsum[-] is the same as \lipsum[1-150].

Only one - is allowed in a range, so if more than one - is given, an error is raised and no paragraphs/sentences are output. No paragraphs or sentences will be output also in case one of the
ranges is reversed, so \lipsum[2-1] returns no paragraphs, as does \lipsum()[2-1] output no sentences, for example. Note that “returning no paragraphs/sentences” is not “the output is empty”: that is mostly true, except that the -before and -after separators are still output (see section 3.3).

Finally, if a range spans more paragraphs or sentences than what the dummy text actually provides, the range is truncated so that it fits the available text. If the range in the argument does not intersect with the range provided by the dummy text, no paragraphs or sentences are output.

### 3.2 Hyphenation patterns

Since version 2.4, the command \lipsum automatically changes the hyphenation patterns when typesetting a dummy text, so that line-breaking looks better (see section 1.1). This feature is on by default, so if you need the old behaviour you have to explicitly disable automatic language switching with \setlipsum{auto-lang=false}.

The language is defined individually for each dummy text (see section 4), but you may change it for the current dummy text by using \setlipsum{language=⟨lang⟩}. If you load another dummy text (for example with the text option), then the option language is also changed according to the dummy text loaded (see section 4).

### 3.3 Paragraph and sentence separators

As may be clear by now, \lipsum has two modes of operation: sentence output, and paragraph output, selected by providing or not providing the second optional argument to \lipsum. In each mode, the dummy text is separated into chunks (paragraphs or sentences), which are counted, and then output accordingly.

When \lipsum (or \unpacklipsum) is used with a single (or no) optional argument, then a range of paragraphs is output, along with some “separators” (in the lack of a better name) between paragraphs, around each paragraph, and before and after the whole output. A schematic (very colorful, because I couldn’t find a better visual) representation of the output is:

```
par-before
par-begin
par-sep
par-end
par-after
sentence-before
sentence-begin
sentence-sep
sentence-end
sentence-after
```

When \lipsum is called, the first thing it outputs is the par-before tokens. These tokens are output unconditionally, regardless of how many (if any) paragraph is output.

Then, before each paragraph in the range, \lipsum outputs the par-begin tokens, and then the actual text of the (paragraph), and then the par-end tokens. These tokens are output conditionally, if the paragraph text is output. If more than one paragraph is output, then the par-sep tokens are inserted between the par-end of one paragraph and the par-begin of the paragraph that follows.

Finally, at the end, the par-after tokens are inserted unconditionally at the end, same as for par-before.

As mentioned before, in case of an error parsing the range, the output will be no paragraphs, but the par-before and par-after tokens are still output.
The explanation above is equally valid for the starred variants. If \lipsum* is used, the par-before* tokens are inserted, and so on. It is also true for sentences (starred or otherwise), replacing par in the option names by sentence, so when you use, for example, \lipsum[][1-9], the sentence-before tokens will be unconditionally inserted, and so on.

Note that, when \lipsum is used in sentence-mode (for example, with \lipsum[1-3][1-9]), only the sentence-... tokens are inserted in the output, regardless of how many paragraphs those sentences were collected from. In the same way, if paragraph-mode is being used, only par-... tokens are inserted.

3.3.1 Deprecated command-based syntax

Older versions of \lipsum (from 2.0 to 2.3) provided 10 CamelCase commands for changing the separators, but the syntax was rather cumbersome to use, so the keyval syntax presented thus far was introduced in the hopes of making things a bit easier. The old commands will still exist for some time in the package, but with a deprecation warning. Changing to the keyval syntax is advised, so here is a correspondence table between the old and new syntaxes:

<table>
<thead>
<tr>
<th>Old command</th>
<th>New key names</th>
</tr>
</thead>
<tbody>
<tr>
<td>\SetLipsumParListStart</td>
<td>par-before</td>
</tr>
<tr>
<td>\SetLipsumParListItemStart</td>
<td>par-begin</td>
</tr>
<tr>
<td>\SetLipsumParListItemSeparator</td>
<td>par-sep</td>
</tr>
<tr>
<td>\SetLipsumParListEnd</td>
<td>par-after</td>
</tr>
<tr>
<td>\SetLipsumSentenceListStart</td>
<td>sentence-before</td>
</tr>
<tr>
<td>\SetLipsumSentenceListItemStart</td>
<td>sentence-begin</td>
</tr>
<tr>
<td>\SetLipsumSentenceListItemSeparator</td>
<td>sentence-sep</td>
</tr>
<tr>
<td>\SetLipsumSentenceListEnd</td>
<td>sentence-end</td>
</tr>
<tr>
<td>\SetLipsumSentenceListEnd</td>
<td>sentence-after</td>
</tr>
</tbody>
</table>

Additionally, the command-based interface provided shortcuts \SetLipsum\langle Thing\rangle\List\langle Item\rangleSurrounders, which are equivalent to just using the commands \SetLipsum\langle Thing\rangle\List\langle Item\rangleStart then \...\End. These don’t provide any functionality, other than requiring a little less typing, so no key-val alternative was implemented. The \...\langle Thing\rangle...Surrounders commands should be replaced by \langle thing\rangle-before and \langle thing\rangle-after, and the \...\langle Thing\rangle...ItemSurrounders commands should be replaced by \langle thing\rangle-begin and \langle thing\rangle-end, as in the correspondence table below:

<table>
<thead>
<tr>
<th>Old command</th>
<th>New key names</th>
</tr>
</thead>
<tbody>
<tr>
<td>\SetLipsumParListSurrounders</td>
<td>par-before</td>
</tr>
<tr>
<td>\SetLipsumParListItemSurrounders</td>
<td>par-after</td>
</tr>
<tr>
<td>\SetLipsumParListItemSurrounders</td>
<td>par-begin</td>
</tr>
<tr>
<td>\SetLipsumSentenceListSurrounders</td>
<td>par-end</td>
</tr>
<tr>
<td>\SetLipsumSentenceListSurrounders</td>
<td>sentence-before</td>
</tr>
<tr>
<td>\SetLipsumSentenceListSurrounders</td>
<td>sentence-after</td>
</tr>
<tr>
<td>\SetLipsumSentenceListSurrounders</td>
<td>sentence-begin</td>
</tr>
<tr>
<td>\SetLipsumSentenceListSurrounders</td>
<td>sentence-end</td>
</tr>
</tbody>
</table>
4  Loading and defining dummy texts

Starting with \texttt{lipsum v2.2}, a simple interface is provided to define and load other texts for the output of \texttt{lipsum} and friends. This interface can, for example, be used to implement dummy texts in different languages without re-coding the logic implemented by \texttt{lipsum}.

\begin{verbatim}
\NewLipsumPar{⟨paragraph⟩}
\end{verbatim}

In order to provide a new text that will be used by \texttt{lipsum}, define the text by using a set of \texttt{\NewLipsumPar{⟨paragraph⟩}} commands in a file with the ending \texttt{.ltd.tex} (\texttt{ltd} means \texttt{lipsum text definition\footnote{To avoid name clashes with files using general languages as names, I chose to introduce the \texttt{.ltd.tex} file ending. I did not find a file with this ending in my \texttt{texmf}-tree, so I guess it is safe.}}) to a location where your \TeX{} system will find it. The \texttt{⟨paragraph⟩}-argument is a single paragraph of the new text. Thus, the first occurrence of \texttt{\NewLipsumPar} defines the first paragraph, the second occurrence the second paragraph and so on.

\begin{verbatim}
\SetLipsumLanguage{⟨lang⟩}
\end{verbatim}

Additionally, tell \texttt{lipsum} the language of the dummy text using \texttt{\SetLipsumLanguage{⟨lang⟩}} somewhere in the \texttt{.ltd.tex} file. To specify the new text as output for \texttt{lipsum} and friends, use \texttt{\setlipsum{text=(name)}}\footnote{https://github.com/PhelypeOleinik/lipsum}, where \texttt{(name)} is the name of the file without the ending \texttt{.ltd.tex}, as given in the table below. When a new dummy text is loaded, the previous one is cleared, and the language is changed as well, according to the table.

\begin{table}
\begin{tabular}{|l|l|l|l|}
\hline
File (.ltd.tex) & Language & Source & Description \\
\hline
\texttt{lipsum} & Latin & James Wilson & Contains the standard \textit{Lorem ipsum} dummy text, obtained from https://www.lipsum.com (default). \\
\texttt{cicero} & Latin & GH user svenper & Contains the speech by Cicero which inspired the \textit{Lorem ipsum}... dummy text. \\
\texttt{lipsum-cs} & Czech & Ondřej Macek & Lipsum dummy text in the Czech language, obtained from Petr Staníček’s website: https://www.wellstyled.com/tools/dummy-cz. \\
\hline
\end{tabular}
\end{table}

4.1 Guidelines on providing new dummy texts

\texttt{\SetLipsumText} more or less just uses an \texttt{\input} or, to be more precise, the \LaTeX{}-variant \texttt{\file_input:n}, to load the \texttt{.ltd.tex} file. This means, that the file is not necessarily loaded in the preamble of the document and thus the contents of the file underlie the respective restrictions.

Should you want a new dummy text, create an issue in the GitHub repository\footnote{https://github.com/PhelypeOleinik/lipsum} with the source for the dummy text.

Should you prefer to distribute the dummy text as a separate package, make sure that the text follows the layout of \texttt{lipsum}’s dummy texts, so that everything works correctly. The dummy text definition file should contain a line with \texttt{\SetLipsumLanguage}, and then as many \texttt{\NewLipsumPar} entries as there are paragraphs in the dummy text. Make sure that the file has the \texttt{.ltd.tex} extension, and everything should work smoothly.
5 \texttt{lipsum} Implementation

5.1 Variables

\texttt{\g__lipsum_par_int} Stores the number of paragraphs in the current text.
\begin{verbatim}
\int_new:N \g__lipsum_par_int
(End definition for \texttt{\g__lipsum_par_int}.)
\end{verbatim}

\texttt{\g__lipsum_language_tl} Stores the language of the dummy text for hyphenation patterns.
\begin{verbatim}
\tl_new:N \g__lipsum_language_tl
(End definition for \texttt{\g__lipsum_language_tl}.)
\end{verbatim}

\texttt{\g__lipsum_default_range_tl} The default range for \texttt{lipsum} paragraphs.
\begin{verbatim}
\tl_new:N \g__lipsum_default_range_tl
(End definition for \texttt{\g__lipsum_default_range_tl}.)
\end{verbatim}

\texttt{\l__lipsum_output_tl} This variables is used to store the token list containing the selected output.
\begin{verbatim}
\tl_new:N \l__lipsum_output_tl
(End definition for \texttt{\l__lipsum_output_tl}.)
\end{verbatim}

\texttt{\g__lipsum_text_str} Holds the current text loaded for the output of \texttt{lipsum} and friends. Used to avoid loading the same text definition if it is already used.
\begin{verbatim}
\str_new:N \g__lipsum_text_str
(End definition for \texttt{\g__lipsum_text_str}.)
\end{verbatim}

\texttt{\l__lipsum_sep_set_str} Holds the name of the active separator token set. By default it is empty to use the default separator set (empty).
\begin{verbatim}
\str_new:N \l__lipsum_sep_set_str
(End definition for \texttt{\l__lipsum_sep_set_str}.)
\end{verbatim}

\texttt{\l__lipsum_autolang_bool} Boolean whether to change hyphenation patterns according to the dummy text language.
\begin{verbatim}
\bool_new:N \l__lipsum_autolang_bool
(End definition for \texttt{\l__lipsum_autolang_bool}.)
\end{verbatim}

\texttt{\q__lipsum_mark} \texttt{\s__lipsum} Quark and scan mark used throughout the package.
\begin{verbatim}
\quark_new:N \q__lipsum_mark
\scan_new:N \s__lipsum
(End definition for \texttt{\q__lipsum_mark} and \texttt{\s__lipsum}.)
\end{verbatim}

\texttt{\l__lipsum_tmpa_str \l__lipsum_a_int \l__lipsum_b_int} Scratch variables.
\begin{verbatim}
\str_new:N \l__lipsum_tmpa_str
\int_new:N \l__lipsum_a_int
\int_new:N \l__lipsum_b_int
(End definition for \texttt{\l__lipsum_tmpa_str}, \texttt{\l__lipsum_a_int}, and \texttt{\l__lipsum_b_int}.)
\end{verbatim}
Scratch macro.

\cs_new_eq:NN \__lipsum_tmp:w ?
(End definition for \__lipsum_tmp:w.)

These variables store the separators and delimiters added around the paragraphs and sentences, in the starred or nonstarred variants, as well as the generic version for runtime usage.

\clist_map_inline:nn { start, itemstart, itemseparator, itemend, end }
\clist_map_inline:nn { par, sentence }
\clist_map_inline:nn { { }, star, nostar }
\tl_new:c { l__lipsum_##1_#1_####1_tl }
\tl_new:c { l__lipsum_par_#1_parsepar_tl }
\tl_set:Nn \l__lipsum_par_itemseparator_parsepar_tl { ~ }
(End definition for \l__lipsum_<thing>_<place>_<version>_tl.)

5.2 Developer interface

Parser argument that may be a single integer or an integer range separated by a -, and stores them into the integer registers #2 and #3. If a number is blank, zero is used. If only a single number is given, #3 is set equal to #2.

\cs_new_protected:Npn \__lipsum_parse_par_range:nNN #1 #2 #3
\cs_new_protected:Npn \__lipsum_parse_sentence_range:nNN #1 #2 #3
\cs_new_protected:Npn \__lipsum_parse_range_arg:nNNn #1
\cs_new_protected:Npn \__lipsum_parse_range_arg:wnNNn #1 - #2 - #3 

\str_if_eq:nnTF {#3} { - }
{ \__lipsum_int_set:Nnn #5 {#1} { 1 }
\__lipsum_int_set:Nnn #6 {#2} {#7} }
{ \tl_if_empty:nTF {#3} { \__lipsum_int_set:Nnn #5 {#1} { \ERROR } 
\int_set_eq:NN #6 #5 }

\__lipsum_int_set:Nnn #5 {#1} { 1 }
\__lipsum_int_set:Nnn #6 {#2} {#7} 
}
\msg_error:nnn { lipsum } { invalid-range } {#4}
\__lipsum_parse_range_arg:nNNn { 2 - 1 } #5 #6 {#7}

(End definition for \__lipsum_parse_range:nNN and others.)

A shorthand to leave an \undexpanded token list.
\cs_new:Npn \__lipsum_sep_item:nn #1 #2
{ \exp_not:v { l__lipsum_#1_#2_ \l__lipsum_sep_set_str _tl } }

(End definition for \__lipsum_sep_item:nn.)

Expands to the paragraphs between \textit{number} and \textit{number} with the proper delimiters added. Text is returned in \exp_not:n, so this macro can be safely used in an \edef.
\cs_new:Npn \lipsum_get_range:nn #1 #2
{ \__lipsum_sep_item:nn { par } { start }
\use:e
{ \exp_not:N \__lipsum_get_paragraph:ww
\__lipsum_build_list:nn {#1} {#2}
\exp_not:N \q__lipsum_mark ;
\exp_not:N \q__lipsum_mark ; \s__lipsum
}\__lipsum_sep_item:nn { par } { end }
}
\cs_new:Npn \__lipsum_build_list:nn #1 #2
{ \int_step_function:nnN
{ \int_max:nn {#1} { 1 } }
{ \int_min:nn {#2} { \g__lipsum_par_int } }
\__lipsum_build_list_aux:n
}
\cs_new:Npn \__lipsum_build_list_aux:n #1 { #1 ; }
\cs_new:Npn \__lipsum_get_paragraph:ww #1 #2 ;
{ \if_meaning:w \q__lipsum_mark #2 \if_meaning:w \q__lipsum_mark #1 \__lipsum_get_paragraph_end:w \else:
\lipsum_get_paragraph:n {#1}
\fi:
\else:
\lipsum_get_paragraph:n {#1}
\__lipsum_sep_item:nn { par } { itemseparator }
\fi:
\lipsum_get_paragraph:n\ Expandstotheparagraph\ (number)\ withtheproperdelimitersadded.Textisreturnedin\ \exp_not:n,sothismacrocanbesafelyusedinan\ \edef.

\__lipsum_unexpanded_par:n\ Expands to the paragraph \langle number \rangle wrapped in \exp_not:n. If \langle number \rangle is out of range, it expands to nothing.

\lipsum_get_sentences:nnn\ \lipsum_get_sentences:nnV\ \__lipsum_get_sentences:nnnw\ \__lipsum_get_sentences_end:w\ Expands to the sentences numbered between \langle number_1 \rangle and \langle number_2 \rangle, inclusive, contained in the \langle text \rangle, and adding the proper separators.
5.3 User- and developer-level commands

\LipsumPar

Macro to typeset a single paragraph of \textit{Lorem ipsum}... Was not officially available in version prior to 2.0.

\#1: Number of the paragraph to typeset.

Implemented as follows:

\begin{verbatim}
\NewDocumentCommand \LipsumPar { m }
{ \__lipsum_deprecated:n \__lipsum_unexpanded_par:n \par }
\end{verbatim}

(End definition for \LipsumPar.)

5.4 Tokens surrounding the \textit{Lorem ipsum}... content

\__lipsum_element_set:nnn

A general macro for setting starred/non-starred versions of several elements used between chunks of dummy text. Arguments are:

\#1: Element name;
\#2: Boolean true or false if the \* variant was used;
\#3: Value to set the element to.

\begin{verbatim}
\cs_new_protected:Npn \__lipsum_element_set:nnn #1 #2 #3
{ \tl_set:cn { l__lipsum_ #1 _ \IfBooleanF {#2} { no } star _tl } {#3} }
\end{verbatim}

(End definition for \__lipsum_element_set:nnn.)
The following macros are considered to be user-level commands and thus all lower-case.

\lipsum #1: Range-like string that specifies the number of the paragraphs taken from Lorem ipsum... If omitted, the value set by \SetLipsumDefault is used, which defaults to 1-7.

#2: Sentences to be typeset from the range selected by \{paragraph range\}. If sentences outside the number of sentences in \{paragraph range\} are specified, only existing sentences are typeset.

The difference between \lipsum and \lipsum* is the token(s) that are inserted after each paragraph (only if called without the second optional argument).

\lipsum and \unpacklipsum have the same interface and do almost the same thing, so both are implemented using a common macro \_\lipsum_do:nnn that does the heavy-lifting, and at the end executes the code in #4.
This command does the same as \lipsum, but instead of typesetting the paragraphs or sentences, it stores the expanded content in the \lipsumexp token list. The tokens between items of the list, set, for example, by using the package option space or by using the \SetLipsum...List commands, are x-expanded.

\NewDocumentCommand {\unpacklipsum} { s O { \g_lipsum_default_range_tl } o } { \__lipsum_do:nnnn {#1} {#2} {#3} { \tl_gset_eq:NN \lipsumexp ##1 } }

This is the main macro for \lipsum and \unpacklipsum. It parses the paragraph range, sets the sentence/paragraph separators, then acts accordingly if a sentence range was provided.

\cs_new_protected:Npn \__lipsum_do:nnnn #1 #2 #3 #4
{ \cs_set_protected:Npn \__lipsum_do:N ##1 {#4}
\__lipsum_parse_par_range:eNN {#2} \l__lipsum_a_int \l__lipsum_b_int
\str_set_eq:NN \l__lipsum_tmpa_str \l__lipsum_sep_set_str
\str_set:Nx \l__lipsum_sep_set_str { \IfBooleanF {#1} { no } star }
\bool_lazy_or:nnTF
{ \tl_if_novalue_p:n {#3} } { \tl_if_blank_p:n {#3} }
{ \tl_set:Nx \l__lipsum_output_tl { \lipsum_get_range:nn { \l__lipsum_a_int } { \l__lipsum_b_int } } }
\str_set_eq:NN \l__lipsum_sep_set_str \l__lipsum_tmpa_str
\__lipsum_do:N \l__lipsum_output_tl
}

\__lipsum_set_hyphens: Selects the hyphenation patterns for the language of the dummy text, using \hyphenrules if that's defined. If \hyphenrules doesn't exist try setting hyphenation with \__lipsum_set_hyphens_raw:. Each \__lipsum_set_hyphens_(method): function appropriately redefines \__lipsum_restore_hyphens: to reset the hyphenation patterns.

\cs_new_protected:Npn \__lipsum_set_hyphens:
\bool_if:NTF \l__lipsum_autolang_bool
{ \use:n } { \use_none:n }
{
\cs_if_exist:NTF \hyphenrules
{
\cs_if_exist:cTF { ver@polyglossia.sty }
{ \__lipsum_set_hyphens_polyglossia: }
{ \__lipsum_set_hyphens_babel: }
}
{ \__lipsum_set_hyphens_raw: }
}
}{
\__lipsum_restore_hyphens:
}
(End definition for \__lipsum_set_hyphens: and \__lipsum_restore_hyphens:.)

\__lipsum_set_hyphens_babel: babel makes things pretty simple. We just check if \l@⟨lang⟩ is defined, and if so, use \hyphenrules to set it, and once more to reset in \__lipsum_restore_hyphens:. \hyphenrules is actually an environment, but in babel its \end part does nothing, and its effect can be undone by just using another \hyphenrules on top of it.

If the language is not defined, the language either doesn’t exist at all, or we are using Lua\TeX. Both cases are handled by \__lipsum_lang_not_available:.

\__lipsum_set_hyphens_polyglossia: polyglossia less friendly. We also check if the language is loaded (looking at \l@⟨lang⟩@loaded), and if it is, load it with the \hyphenrules environment. Here we can’t use the command form, as the \end part is not a no-op. This also means that an extra group is added around the dummy text, which causes issue #1\(^5\) when used with \texttt{wrapfig}, for example. But not too much we can do about that for now.

In case the language is not loaded, fall back to \__lipsum_set_hyphens_raw: for a final attempt before giving up.

\__lipsum_set_hyphens_polyglossia:
\cs_new_protected:Npn \__lipsum_set_hyphens_polyglossia:
{\cs_if_exist:cTF { \l@ \g__lipsum_language_tl @loaded }
{\exp_args:NnV \begin{hyphenrules} \g__lipsum_language_tl
\cs_set_protected:Npx \__lipsum_restore_hyphens:
{\exp_not:N \hyphenrules \languagename }
}
{ \__lipsum_lang_not_available: }
}
(End definition for \__lipsum_set_hyphens_babel:.)

\__lipsum_set_hyphens_polyglossia:
\cs_new_protected:Npn \__lipsum_set_hyphens_polyglossia:
{\cs_if_exist:cTF { \l@ \g__lipsum_language_tl @loaded }
{\exp_args:NnV \begin{hyphenrules} \g__lipsum_language_tl
\cs_set_protected:Npn \__lipsum_restore_hyphens:
{\end{hyphenrules} }
5https://github.com/PhelypeOleinik/lipsum/issues/1
If nothing else is available, try setting the language using \language{number}. This is always available, except with LuaTeX, which loads languages on-the-fly.

\cs_new_protected:Npn \_lipsum_set_hyphens_raw:
\{ \cs_if_exist:cTF { l\g__lipsum_language_tl } 
\{ \use:x
\{ \language \use:c { l\g__lipsum_language_tl } \cs_set_protected:Npn \_lipsum_restore_hyphens: 
\{ \language \int_eval:n { \language } \scan_stop: \}
\}
\{ \_lipsum_lang_not_available: \}
\}

(End definition for \_lipsum_set_hyphens_polyglossia:)

\_lipsum_lang_not_available:
If the requested language is for some reason unavailable, warn the user, then fall back to the current language.

\cs_new_protected:Npn \_lipsum_lang_not_available:
\{ \msg_warning:nnx { lipsum } { missing-language } 
\{ \g__lipsum_language_tl \tl_gset_eq:NN \g__lipsum_language_tl \languagename \}
\}

(End definition for \_lipsum_lang_not_available:)

\NewLipsumPar
Developer-Level macro to add a paragraph to the dummy text used by \lipsum and related commands. To specify a new dummy text, see section 4.

\cs_new_protected:Npn \NewLipsumPar #1
\{ \int_gincr:N \g__lipsum_par_int 
\tl_gclear_new:c { \g__lipsum_par_int_tl } \tl_gset:c { \g__lipsum_par_int_tl } {#1} \}

(End definition for \NewLipsumPar. This function is documented on page 7.)

\SetLipsumText
Used to select and load the text output by \lipsum and friends. See the section on loading and defining new outputs for \lipsum (section 4). It first checks whether the requested text is already loaded, and if not, it loads the corresponding lipsum text definition file, and clears remaining paragraphs from the previous text, in case their lengths differ.

\NewDocumentCommand \SetLipsumText { m }
\{ \str_if_eq:VN \g__lipsum_text_str {#1} \}

16
\input{ltd}
\SetLipsumLanguage

This macro sets the language for hyphenation patterns of the dummy text. When a new
lipsum text is read, this is reset.
\NewDocumentCommand \SetLipsumLanguage { m }
{ \tl_gset:Nn \g__lipsum_language_tl {#1} }

\SetLipsumLanguage

5.5 Package options and defaults

These are some auxiliaries for the package options and for setting up the default behaviour.
\cs_new_protected:Npn \__lipsum_delim_restore:nnn #1 #2 #3
{ \keys_set:nn { lipsum } { #1-before = , #1-begin = , #1-end = , #1-after = ,
#1-before* = , #1-begin* = , #1-end* = , #1-after* = ,
#1-sep = {#2}, #1-sep* = {#3} }
\cs_new_protected:Nn \__lipsum_restore_sentence_list:
{ \__lipsum_delim_restore:nnn { sentence } { ~ } { ~ } }
\cs_new_eq:NN \__lipsum_restore_par_list: ? \cs_new_protected:Npn \LipsumRestoreParList
{ \__lipsum_deprecated:n { LipsumRestoreParList } \__lipsum_restore_par_list: }
\cs_new_protected:Nn \__lipsum_restore_sentence_list:
\cs_new_protected:Nn \__lipsum_restore_all:
\keys_define:nn { lipsum }
{ #1-before = , #1-begin = , #1-end = , #1-after = ,
#1-before* = , #1-begin* = , #1-end* = , #1-after* = ,
#1-sep = {#2}, #1-sep* = {#3} }
\NewDocumentCommand \setlipsum { +m }
{ \keys_set:nn { lipsum } {#1} }
\keys_define:nn { lipsum }

\setlipsum

Here are the options available at load-time and to \setlipsum.
\NewDocumentCommand \setlipsum { +m }
{ \keys_set:nn { lipsum } {#1} }
nopar is implemented as a choice key instead of a boolean so we can update the separators using \_\_lipsum_delim_restore:nnn. It’s initially false, and the default is true so that \usepackage[nopar]{lipsum} works as it always did.

\begin{verbatim}
      .choice: ,
nopar / true .code:n =
\{\cs_gset_protected:Nn \_\_lipsum_restore_par_list:
\{ \_\_lipsum_delim_restore:nnn { par } { - } { \par } \}
\},
nopar / false .code:n =
\{\cs_gset_protected:Nn \_\_lipsum_restore_par_list:
\{ \_\_lipsum_delim_restore:nnn { par } { \par } { - } \}
\},
nopar .initial:n = false ,
nopar .default:n = true ,
\end{verbatim}

auto-lang sets \_\_lipsum_autolang_bool. It is initially true, changing the default behaviour from previous versions.

\begin{verbatim}
      auto-lang .bool_set:N = \_\_lipsum_autolang_bool ,
      auto-lang .initial:n = true ,
      auto-lang .default:n = true ,
\end{verbatim}

text just does \SetLipsumText. The initial value is not set here because this chunk of code is executed in expl3 syntax, then the text loads without spaces, so \setlipsum[text=lipsum] is used later.

\begin{verbatim}
      text .code:n = \SetLipsumText{#1} ,
      text .value_required:n = true ,
\end{verbatim}

language sets the language to be used when typesetting.

\begin{verbatim}
      language .tl_gset:N = \g__lipsum_language_tl ,
      language .value_required:n = true ,
\end{verbatim}

default-range does \SetLipsumDefault, initially 1-7, as documented. It’s default is also 1-7 so that the key has two meanings: \setlipsum[default-range=⟨range⟩] sets the range to the given value, while \setlipsum[default-range] sets the range to the “default default range”. Pretty neat :) 

\begin{verbatim}
      default-range .code:n = \SetLipsumDefault{#1} ,
      default-range .initial:n = 1-7 ,
      default-range .default:n = 1-7 ,
    \end{verbatim}

This chunk defines the keys ⟨thing⟩-⟨place⟩[*], where ⟨thing⟩ is par or sentence, ⟨place⟩ is before, begin, sep, end, and after, which totals 10 keys, and another 10 with the * in the name. Each sets a token list called \_\_lipsum⟨⟨thing⟩⟩⟨⟨place⟩⟩[no]star_tl.

\begin{verbatim}
\cs_set_protected:Npn \_\_lipsum_tmp:w #1 #2 #3
\{\keys_define:nn { lipsum }
\begin{verbatim}
  \_1-before #2 .tl_set:c = \_\_lipsum_\_1_start #3star_tl ,
  \_1-begin #2 .tl_set:c = \_\_lipsum_\_1_itemstart #3star_tl ,
  \_1-sep #2 .tl_set:c = \_\_lipsum_\_1_itemseparator #3star_tl ,
  \_1-end #2 .tl_set:c = \_\_lipsum_\_1_itemend #3star_tl ,
  \_1-after #2 .tl_set:c = \_\_lipsum_\_1_end #3star_tl ,
\end{verbatim}
\end{verbatim}
Now turn \ExplSyntaxOff for a while, and load the default \textit{Lorem ipsum}... text, then process the package options, and finally turn \ExplSyntaxOn again. Finally, call \__lipsum_restore_par_list: and \__lipsum_restore_sentence_list: to set the defaults (\__lipsum_restore_par_list: may have been redefined by \nopar).

\ExplSyntaxOff
\setlipsum{text=lipsum}
\ProcessKeysOptions{lipsum}
\ExplSyntaxOn
\__lipsum_restore_par_list:
\__lipsum_restore_sentence_list:

5.6 Messages

Now define the messages used throughout the package.

\msg_new:nnn { lipsum } { invalid-range }
\{ Invalid-number-or-range-\texttt{#1}. \}
\msg_new:nnn { lipsum } { cmd-deprecated }
\{ Command-\texttt{\iow_char:N\#1} deprecated. \textbackslash \textbackslash
See-the-lipsum-documentation-for-help. \}
\msg_new:nnn { lipsum } { missing-language }
\{ Unknown-language-\texttt{\#1}. Hyphenation-patterns-for-
\texttt{\languagename}-will-be-used-instead. \}
\sys_if_engine_luatex:T
\{ \}
\ifcs_if_exist:cTF { ver@polyglossia.sty }
\{ With-polyglossia,-you-have-to-explicitly-load-languages-
with-\texttt{\iow_char:N\setotherlanguage{\#1}}-or-similar. \}
\}
\{ With-LuaTeX,-lipsum-requires-babel-to-get-proper-
hyphenation-(you-can-use-
\texttt{\iow_char:N\usepackage{base}\{babel\}}). \}
\}
\}

{/package}