

# lipsum

Access to 150 paragraphs of Lorem Ipsum dummy text<sup>ab</sup>

Patrick Happel<sup>c</sup>

September 20, 2021

## Abstract

`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 `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>.

---

<sup>a</sup>Version: 2.7

<sup>b</sup>Since March 2021, Phelype Oleinik maintains this package ([phelype.oleinik@latex-project.org](mailto:phelype.oleinik@latex-project.org)).

<sup>c</sup>patrick.happel@rub.de

## 1 Introduction

To load the package, write

```
\usepackage{lipsum}
```

`\lipsum` in the preamble of your document. Probably the most important macro provided by this package is `\lipsum`, which typesets the *Lorem ipsum* paragraphs. The first optional argument allows to specify the range of the paragraphs. For example, `\lipsum[4-57]` typesets the paragraphs 4 to 57 and accordingly, `\lipsum[23]` typesets the 23<sup>rd</sup> paragraph. Using `\lipsum` without its optional argument typesets the paragraphs 1–7 of *Lorem ipsum*...

As of version 2.0, `\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 `\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 `\usepackage[text=lipsum-cs]{lipsum}` or use `\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 `lipsum` with `\usepackage[auto-lang=false]{lipsum}` or, as above, use `\setlipsum{auto-lang=false}`.

Finally, as demonstrated above, a new macro `\setlipsum` was added to change package options anywhere in the document, so you may change, for example, the dummy text printed by `\lipsum`

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

```
\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{%
  (\themycnt)\stepcounter{mycnt}}
\lipsumexp
\end{document}
```

- The internal macros `\lips@i`, `\lips@ii`, `\lips@iii`, ..., `\lips@c1` 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, `lipsum` provides a simple interface to define other texts to be used as output of the `\lipsum`-family of commands. This was heavily inspired by an issue raised by *svenper* on github<sup>1</sup>. 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 *Lorem ipsum* text.

---

<sup>1</sup><https://github.com/patta42/lipsum/issues/13>

## 2 Usage

`lipsum` was intended to quickly provide a way to fill a page or two to analyze the page layout<sup>2</sup>. 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 `\unpackedlipsum` commands, and can be set at load-time with `\usepackage[option]{lipsum}`, or later in the document by using `\setlipsum{option}`.

<hr/> <code>nopar</code> <hr/>	= <i>&lt;boolean&gt;</i>	(default: <code>false</code> )
	Changes the initial default separator between each paragraph of <code>\lipsum</code> from <code>\par</code> to <code>\space</code> , and the other way around for <code>\lipsum*</code> .	
<hr/> <code>text</code> <hr/>	= <i>&lt;name&gt;</i>	(default: <code>lipsum</code> )
	Selects the dummy text <i>&lt;name&gt;</i> that is used by <code>\lipsum</code> and <code>\unpackedlipsum</code> (see section 4).	
<hr/> <code>language</code> <hr/>	= <i>&lt;lang&gt;</i>	(default: <code>latin</code> )
	Sets the language to be used by <code>\lipsum</code> to typeset the currently active dummy text (see section 3.2). Changing the dummy text with the <code>text</code> option will also change the current <code>language</code> .	
<hr/> <code>auto-lang</code> <hr/>	= <i>&lt;boolean&gt;</i>	(default: <code>true</code> )
	Turns on/off automatic language switching. This changed since version 2.3, in which this option (didn't exist thus) was <code>false</code> by default. See section 3.2 for more details.	
<hr/> <code>default-range</code> <hr/>	= <i>&lt;p<sub>i</sub>-p<sub>f</sub>&gt;</i>	(default: <code>1-7</code> )
	Sets the default range of paragraphs produced by <code>\lipsum</code> when no optional argument is provided. The value to <code>default-range</code> obeys the <i>&lt;range&gt;</i> syntax described in section 3.1. If no value is given to <code>default-range</code> (that is, <code>\setlipsum{default-range}</code> ), then the default is reset to <code>1-7</code> .	

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

<hr/> <code>\lipsum</code> <hr/>	<code>\lipsum<sup>*</sup>[<i>&lt;par range&gt;</i>][<i>&lt;sentence range&gt;</i>]</code>
	<code>\lipsum</code> outputs the <i>&lt;par range&gt;</i> from the currently active dummy text. If <i>&lt;par range&gt;</i> is not given or is empty, the <code>default-range</code> (initially <code>1-7</code> ) is output. If a <i>&lt;sentence range&gt;</i> 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 <i>&lt;sup&gt;*</i> version is used, a different set of separators is inserted around the paragraphs or sentences.
	<code>\lipsum</code> 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.

<sup>2</sup><https://groups.google.com/d/topic/de.comp.text.tex/oPeL0jkrLfk>

---

<code>\unpacklipsum</code>	<code>\unpacklipsum*</code> [ <i>par range</i> ] [ <i>sentence range</i> ]
<code>\lipsumexp</code>	...

---

`\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.

---

<code>\setlipsum</code>	<code>\setlipsum</code> { <i>key-val list</i> }
-------------------------	---

---

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 compatibility, and should normally not be needed in user documents.

---

<code>\SetLipsumText</code>	<code>\SetLipsumDefault</code> { <i>name</i> }
-----------------------------	--

---

Loads the dummy text *name* (see section 4). This command does the same as option `text`, but it is kept for backwards compatibility.

---

<code>\SetLipsumDefault</code>	<code>\SetLipsumDefault</code> { <i>range</i> }
--------------------------------	---

---

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 `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 *proper range*  $\langle n_i \rangle - \langle n_f \rangle$ . In a proper range, if  $\langle n_i \rangle$  is blank, it is taken to be the start of the possible range, and in the same way, if  $\langle n_f \rangle$  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:

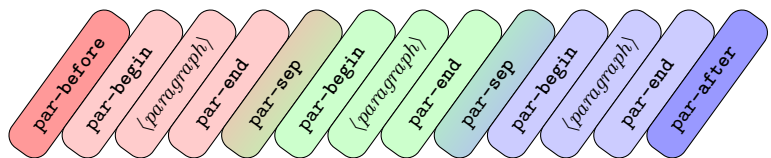
---

```

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:

Old command	New key name
<code>\SetLipsumParListStart</code>	<code>par-before</code>
<code>\SetLipsumParListItemStart</code>	<code>par-begin</code>
<code>\SetLipsumParListItemSeparator</code>	<code>par-sep</code>
<code>\SetLipsumParListItemEnd</code>	<code>par-end</code>
<code>\SetLipsumParListEnd</code>	<code>par-after</code>
<code>\SetLipsumSentenceListStart</code>	<code>sentence-before</code>
<code>\SetLipsumSentenceListItemStart</code>	<code>sentence-begin</code>
<code>\SetLipsumSentenceListItemSeparator</code>	<code>sentence-sep</code>
<code>\SetLipsumSentenceListItemEnd</code>	<code>sentence-end</code>
<code>\SetLipsumSentenceListEnd</code>	<code>sentence-after</code>

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

Old command	New key names
<code>\SetLipsumParListSurrounders</code> -----	<ul style="list-style-type: none"> <li>  <code>par-before</code></li> <li>  <code>par-after</code></li> </ul>
<code>\SetLipsumParListItemSurrounders</code> -----	<ul style="list-style-type: none"> <li>  <code>par-begin</code></li> <li>  <code>par-end</code></li> </ul>
<code>\SetLipsumSentenceListSurrounders</code> -----	<ul style="list-style-type: none"> <li>  <code>sentence-before</code></li> <li>  <code>sentence-after</code></li> </ul>
<code>\SetLipsumSentenceListItemSurrounders</code> -----	<ul style="list-style-type: none"> <li>  <code>sentence-begin</code></li> <li>  <code>sentence-end</code></li> </ul>

## 4 Loading and defining dummy texts

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

---

`\NewLipsumPar` `\NewLipsumPar{<paragraph>}`

In order to provide a new text that will be used by `lipsum`, define the text by using a set of `\NewLipsumPar{<paragraph>}` commands in a file with the ending `.ltd.tex` (`ltd` means *lipsum text definition*<sup>3</sup>) to a location where your `TEX` system will find it. The `<paragraph>`-argument is a single paragraph of the new text. Thus, the first occurrence of `\NewLipsumPar` defines the first paragraph, the second occurrence the second paragraph and so on.

---

`\SetLipsumLanguage` `\SetLipsumLanguage{<lang>}`

Additionally, tell `lipsum` the language of the dummy text using `\SetLipsumLanguage{<lang>}` somewhere in the `.ltd.tex` file.

To specify the new text as output for `\lipsum` and friends, use `\setlipsum{text=<name>}`, where `<name>` is the name of the file without the ending `.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.

File ( <code>.ltd.tex</code> )	Language	Source	Description
<code>lipsum</code>	Latin	James Wilson	Contains the standard <i>Lorem ipsum</i> dummy text, obtained from <a href="https://www.lipsum.com">https://www.lipsum.com</a> (default).
<code>cicero</code>	Latin	GH user <a href="#">svenper</a>	Contains the speech by Cicero which inspired the <i>Lorem ipsum...</i> dummy text.
<code>lipsum-cs</code>	Czech	Ondřej Macek	Lipsum dummy text in the Czech language, obtained from Petr Staníček's website: <a href="https://www.wellstyled.com/tools/dummy-cz">https://www.wellstyled.com/tools/dummy-cz</a> .

### 4.1 Guidelines on providing new dummy texts

`\SetLipsumText` more or less just uses an `\input` or, to be more precise, the `LATEX3`-variant `\file_input:n`, to load the `.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<sup>4</sup> 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 `lipsum`'s dummy texts, so that everything works correctly. The dummy text definition file should contain a line with `\SetLipsumLanguage`, and then as many `\NewLipsumPar` entries as there are paragraphs in the dummy text. Make sure that the file has the `.ltd.tex` extension, and everything should work smoothly.

<sup>3</sup>To avoid name clashes with files using general languages as names, I chose to introduce the `.ltd.tex` file ending. I did not find a file with this ending in my `texmf-tree`, so I guess it is safe.

<sup>4</sup><https://github.com/Phelype01einik/lipsum>

## 5 lipsum Implementation

```
1 \*package
2 \@@=lipsum
```

### 5.1 Variables

`\g__lipsum_par_int` Stores the number of paragraphs in the current text.

```
3 \int_new:N \g__lipsum_par_int
```

*(End definition for \g\_\_lipsum\_par\_int.)*

`\g__lipsum_language_tl` Stores the language of the dummy text for hyphenation patterns.

```
4 \tl_new:N \g__lipsum_language_tl
```

*(End definition for \g\_\_lipsum\_language\_tl.)*

`\g_lipsum_default_range_tl` The default range for lipsum paragraphs.

```
5 \tl_new:N \g_lipsum_default_range_tl
```

*(End definition for \g\_lipsum\_default\_range\_tl.)*

`\l__lipsum_output_tl` This variables is used to store the token list containing the selected output.

```
6 \tl_new:N \l__lipsum_output_tl
```

*(End definition for \l\_\_lipsum\_output\_tl.)*

`\g__lipsum_text_str` Holds the current text loaded for the output of `\lipsum` and friends. Used to avoid loading the same text definition if it is already used.

```
7 \str_new:N \g__lipsum_text_str
```

*(End definition for \g\_\_lipsum\_text\_str.)*

`\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).

```
8 \str_new:N \l__lipsum_sep_set_str
```

*(End definition for \l\_\_lipsum\_sep\_set\_str.)*

`\l__lipsum_autolang_bool` Boolean whether to change hyphenation patterns according to the dummy text language.

```
9 \bool_new:N \l__lipsum_autolang_bool
```

*(End definition for \l\_\_lipsum\_autolang\_bool.)*

`\q__lipsum_mark` Quark and scan mark used throughout the package.

```
\s__lipsum 10 \quark_new:N \q__lipsum_mark
```

```
11 \scan_new:N \s__lipsum
```

*(End definition for \q\_\_lipsum\_mark and \s\_\_lipsum.)*

`\l__lipsum_tmpa_str` Scratch variables.

```
\l__lipsum_a_int 12 \str_new:N \l__lipsum_tmpa_str
```

```
\l__lipsum_b_int 13 \int_new:N \l__lipsum_a_int
```

```
14 \int_new:N \l__lipsum_b_int
```

*(End definition for \l\_\_lipsum\_tmpa\_str, \l\_\_lipsum\_a\_int, and \l\_\_lipsum\_b\_int.)*



`\_lipsum_tmp:w` Scratch macro.

```
15 \cs_new_eq:NN \_lipsum_tmp:w ?
```

(End definition for `\_lipsum_tmp:w`.)

`\_lipsum_<thing>_<place>_<version>_tl` 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.

```
16 \clist_map_inline:nn { start, itemstart, itemseparator, itemend, end }
17 {
18   \clist_map_inline:nn { par, sentence }
19   {
20     \clist_map_inline:nn { { }, star, nostar }
21     { \tl_new:c { l_lipsum_##1_#1_###1_tl } }
22   }
23   \tl_new:c { l_lipsum_par_#1_parsepar_tl }
24 }
25 \tl_set:Nn \l_lipsum_par_itemseparator_parsepar_tl { ~ }
```

(End definition for `\_lipsum_<thing>_<place>_<version>_tl`.)

## 5.2 Developer interface

`\_lipsum_parse_par_range:nNN` Parses an 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.

`\_lipsum_parse_par_range:eNN`

`\_lipsum_parse_sentence_range:nNN`

`\_lipsum_parse_sentence_range:eNN`

`\_lipsum_parse_range_arg:nNNn`

`\_lipsum_parse_range_arg:wnNNn`

`\_lipsum_int_set:Nnn`

```
26 \cs_new_protected:Npn \_lipsum_parse_par_range:nNN #1 #2 #3
```

```
27 {
```

```
28   \tl_if_blank:nTF {#1}
```

```
29   { \exp_args:NV \_lipsum_parse_range_arg:nNNn \g_lipsum_default_range_tl }
```

```
30   { \_lipsum_parse_range_arg:nNNn {#1} }
```

```
31   #2 #3 { \g_lipsum_par_int }
```

```
32 }
```

```
33 \cs_new_protected:Npn \_lipsum_parse_sentence_range:nNN #1 #2 #3
```

```
34 { \_lipsum_parse_range_arg:nNNn {#1} #2 #3 { \c_max_int } }
```

```
35 \cs_new_protected:Npn \_lipsum_parse_range_arg:nNNn #1
```

```
36 {
```

```
37   \exp_last_unbraced:No \_lipsum_parse_range_arg:wnNNn
```

```
38   \tl_to_str:n { #1 - - } \s_lipsum {#1}
```

```
39 }
```

```
40 \cs_new_protected:Npn \_lipsum_parse_range_arg:wnNNn
```

```
41 #1 - #2 - #3 \s_lipsum #4 #5#6 #7
```

```
42 {
```

```
43   \str_if_eq:nnTF {#3} { - }
```

```
44   {
```

```
45     \_lipsum_int_set:Nnn #5 {#1} { 1 }
```

```
46     \_lipsum_int_set:Nnn #6 {#2} {#7}
```

```
47   }
```

```
48   {
```

```
49     \tl_if_empty:nTF {#3}
```

```
50     {
```

```
51       \_lipsum_int_set:Nnn #5 {#1} { \ERROR }
```

```
52       \int_set_eq:NN #6 #5
```

```

53     }
54     {
55         \msg_error:nnn { lipsum } { invalid-range } {#4}
56         \__lipsum_parse_range_arg:nNNn { 2 - 1 } #5 #6 {#7}
57     }
58 }
59 }
60 \cs_new_protected:Npn \__lipsum_int_set:Nnn #1 #2 #3
61 { \int_set:Nn #1 { \tl_if_blank:nT {#2} {#3} #2 } }
62 \cs_generate_variant:Nn \__lipsum_parse_par_range:nNN { e }
63 \cs_generate_variant:Nn \__lipsum_parse_sentence_range:nNN { e }

```

(End definition for `\__lipsum_parse_par_range:nNN` and others.)

`\__lipsum_sep_item:nn` A shorthand to leave an (`\undexpanded`) token list.

```

64 \cs_new:Npn \__lipsum_sep_item:nn #1 #2
65 { \exp_not:v { l__lipsum_#1_#2_ \l__lipsum_sep_set_str_tl } }

```

(End definition for `\__lipsum_sep_item:nn`.)

`\lipsum_get_range:nn` Expands to the paragraphs between  $\langle number_1 \rangle$  and  $\langle number_2 \rangle$  with the proper delimiters added. Text is returned in `\exp_not:n`, so this macro can be safely used in an `\edef`.

```

\__lipsum_build_list:nn
\__lipsum_build_list_aux:n
\__lipsum_get_paragraph:ww
  \__lipsum_get_paragraph_end:w
66 \cs_new:Npn \lipsum_get_range:nn #1 #2
67 {
68     \__lipsum_sep_item:nn { par } { start }
69     \use:e
70     {
71         \exp_not:N \__lipsum_get_paragraph:ww
72         \__lipsum_build_list:nn {#1} {#2}
73         \exp_not:N \q__lipsum_mark ;
74         \exp_not:N \q__lipsum_mark ; \s__lipsum
75     }
76     \__lipsum_sep_item:nn { par } { end }
77 }
78 \cs_new:Npn \__lipsum_build_list:nn #1 #2
79 {
80     \int_step_function:nnN
81     { \int_max:nn {#1} { 1 } }
82     { \int_min:nn {#2} { \g__lipsum_par_int } }
83     \__lipsum_build_list_aux:n
84 }
85 \cs_new:Npn \__lipsum_build_list_aux:n #1 { #1 ; }
86 \cs_new:Npn \__lipsum_get_paragraph:ww #1 ; #2 ;
87 {
88     \if_meaning:w \q__lipsum_mark #2
89     \if_meaning:w \q__lipsum_mark #1
90     \__lipsum_get_paragraph_end:w
91     \else:
92     \lipsum_get_paragraph:n {#1}
93     \fi:
94     \else:
95     \lipsum_get_paragraph:n {#1}
96     \__lipsum_sep_item:nn { par } { itemseparator }
97     \fi:

```

```

98   \__lipsum_get_paragraph:ww #2 ;
99   }
100  \cs_new:Npn \__lipsum_get_paragraph_end:w #1 \s__lipsum { \fi: \fi: }

```

(End definition for \lipsum\_get\_range:nn and others.)

`\lipsum_get_paragraph:n` Expands to the paragraph  $\langle number \rangle$  with the proper delimiters added. Text is returned in `\exp_not:n`, so this macro can be safely used in an `\edef`.

```

101  \cs_new:Npn \lipsum_get_paragraph:n #1
102  {
103    \__lipsum_sep_item:nn { par } { itemstart }
104    \__lipsum_unexpanded_par:n {#1}
105    \__lipsum_sep_item:nn { par } { itemend }
106  }

```

(End definition for \lipsum\_get\_paragraph:n.)

`\__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.

```

107  \cs_new:Npn \__lipsum_unexpanded_par:n #1
108  {
109    \bool_lazy_and:nnT
110      { \int_compare_p:nNn { 0 } < {#1} }
111      { \int_compare_p:nNn {#1} < { \g__lipsum_par_int + 1 } }
112    { \exp_not:v { g__lipsum_par_#1_tl } }
113  }

```

(End definition for \\_\_lipsum\_unexpanded\_par:n.)

`\lipsum_get_sentences:nnn` 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.

`\lipsum_get_sentences:nnV`  
`\__lipsum_get_sentences:nnnw`  
`\__lipsum_get_sentences_end:w`

```

114  \cs_new:Npn \lipsum_get_sentences:nnn #1 #2 #3
115  {
116    \__lipsum_sep_item:nn { sentence } { start }
117    \exp_args:Ne \use_ii_i:nn { { \int_max:nn {#1} { 1 } } }
118    { \__lipsum_get_sentences:nnnw { 1 } } {#2}
119    #3 ~ \q__lipsum_mark .~ \s__lipsum
120    \__lipsum_sep_item:nn { sentence } { end }
121  }
122  \cs_new:Npn \__lipsum_get_sentences:nnnw #1 #2 #3 #4 .~
123  {
124    \int_compare:nNnT {#1} > {#3} { \__lipsum_get_sentences_end:w }
125    \use:nn { \if_meaning:w \q__lipsum_mark } #4
126    \exp_after:wN \__lipsum_get_sentences_end:w
127    \else:
128      \int_compare:nNnF {#1} < {#2}
129      {
130        \int_compare:nNnF {#1} = {#2}
131        { \__lipsum_sep_item:nn { sentence } { itemseparator } }
132        \__lipsum_sep_item:nn { sentence } { itemstart }
133        \exp_not:n { #4 . }
134        \__lipsum_sep_item:nn { sentence } { itemend }
135      }
136    \fi:

```

```

137     \exp_args:Nf \__lipsum_get_sentences:nnw { \int_eval:n { #1 + 1 } }
138         {#2} {#3}
139     }
140 \cs_new:Npn \__lipsum_get_sentences_end:w #1 \s__lipsum { }
141 \cs_generate_variant:Nn \lipsum_get_sentences:nnn { nnV }

```

(End definition for \lipsum\_get\_sentences:nnn, \\_\_lipsum\_get\_sentences:nnw, and \\_\_lipsum\_get\_sentences\_end:w.)

### 5.3 User- and developer-level commands

`\LipsumPar` Macro to typeset a single paragraph of *Lorem ipsum...* Was not officially available in version prior to 2.0.

**#1** : Number of the paragraph to typeset.

Implemented as follows:

```

142 \NewDocumentCommand \LipsumPar { m }
143 {
144     \__lipsum_deprecated:n { LipsumPar }
145     \__lipsum_unexpanded_par:n {#1} \par
146 }

```

(End definition for \LipsumPar.)

### 5.4 Tokens surrounding the *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.

```

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

```

(End definition for \\_\_lipsum\_element\_set:nnn.)

`\__lipsum_deprecated:n` Warns about deprecated commands and destroys itself.

```

149 \cs_new_protected:Npn \__lipsum_deprecated:n #1
150 {
151     \msg_warning:nnn { lipsum } { cmd-deprecated } {#1}
152     \cs_gset_eq:NN \__lipsum_deprecated:n \use_none:n
153 }

```

(End definition for \\_\_lipsum\_deprecated:n.)

`\SetLipsumParListStart` A dirty loop to quickly define the old command-based user-interface.

```

\SetLipsumParListEnd
\SetLipsumParListSurrounders
\SetLipsumParListItemSeparator
\SetLipsumParListItemStart
\SetLipsumParListItemEnd
\SetLipsumParListItemSurrounders
\SetLipsumSentenceListStart
\SetLipsumSentenceListEnd
\SetLipsumSentenceListSurrounders
\SetLipsumSentenceListItemSeparator
\SetLipsumSentenceListItemStart
\SetLipsumSentenceListItemEnd
\SetLipsumSentenceListItemSurrounders

```

```

154 \cs_set_protected:Npn \__lipsum_tmp:w #1 #2 #3 #4
155 {
156     \str_set:Nx \l__lipsum_tmpa_str
157         { #2 \tl_if_empty:nTF {#4} {#3} { start } }
158     \use:e
159     {
160         \NewDocumentCommand \exp_not:c { SetLipsum #1 List #2 #3 }
161             { s +m \tl_if_empty:nF {#4} { +m } }

```

```

162     {
163       \__lipsum_deprecated:n { SetLipsum #1 List #2 #3 }
164       \__lipsum_element_set:nnn
165       { \exp_args:Ne \str_lowercase:n { #1\_l__lipsum_tmpa_str } }
166       {##1} {##2}
167       \tl_if_empty:nT {#4} { \use_none:nnnn }
168       \__lipsum_element_set:nnn { \str_lowercase:n { #1_#2 #4 } }
169       {##1} {##3}
170     }
171   }
172 }
173 \clist_map_inline:nn { Par, Sentence }
174 {
175   \clist_map_inline:nn
176   { { Start } { }, { End } { }, { Surrounders } { end } }
177   { \__lipsum_tmp:w {#1} { Item } ##1 \__lipsum_tmp:w {#1} { } ##1 }
178   \__lipsum_tmp:w {#1} { Item } { Separator } { }
179 }

```

(End definition for `\SetLipsumParListStart` and others.)

**`\SetLipsumDefault`** Command to change the default range used by `\lipsum` and friends.

*<range>* Range to be used as default.

Implemented as:

```

180 \NewDocumentCommand \SetLipsumDefault { m }
181 {
182   \__lipsum_parse_par_range:eNN {#1} \l__lipsum_a_int \l__lipsum_b_int
183   \tl_gset:Nx \g_lipsum_default_range_tl
184   { \int_use:N \l__lipsum_a_int - \int_use:N \l__lipsum_b_int }
185 }

```

(End definition for `\SetLipsumDefault`. This function is documented on page 4.)

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:nnnn` that does the heavy-lifting, and at the end executes the code in #4.

```

186 \NewDocumentCommand \lipsum { s O { \g_lipsum_default_range_tl } o }
187 {
188   \__lipsum_do:nnnn {#1} {#2} {#3}
189   {
190     \__lipsum_set_hyphens:
191     \tl_use:N ##1
192     \__lipsum_restore_hyphens:
193   }

```

```
194 }
```

(End definition for `\lipsum`. This function is documented on page 3.)

`\unpacklipsum` 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.

`\lipsumexp`

```
195 \NewDocumentCommand \unpacklipsum { s 0 { \g_lipsum_default_range_tl } o }
196   { \__lipsum_do:nnnn {#1} {#2} {#3} { \tl_gset_eq:NN \lipsumexp ##1 } }
197 \cs_new_eq:NN \lipsumexp \prg_do_nothing:
```

(End definition for `\unpacklipsum` and `\lipsumexp`. These functions are documented on page 4.)

`\__lipsum_do:nnnn` 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.

`\__lipsum_do:N`

```
198 \cs_new_protected:Npn \__lipsum_do:nnnn #1 #2 #3 #4
199   {
200     \cs_set_protected:Npn \__lipsum_do:N ##1 {#4}
201     \__lipsum_parse_par_range:eNN {#2} \l__lipsum_a_int \l__lipsum_b_int
202     \str_set_eq:NN \l__lipsum_tmpa_str \l__lipsum_sep_set_str
203     \str_set:Nx \l__lipsum_sep_set_str { \IfBooleanF {#1} { no } star }
204     \bool_lazy_or:nnTF
205       { \tl_if_novalue_p:n {#3} }
206       { \tl_if_blank_p:n {#3} }
207     {
208       \tl_set:Nx \l__lipsum_output_tl
209         { \lipsum_get_range:nn { \l__lipsum_a_int } { \l__lipsum_b_int } }
210     }
211     {
212       \str_set:Nn \l__lipsum_sep_set_str { parsepar }
213       \tl_set:Nx \l__lipsum_output_tl
214         { \lipsum_get_range:nn { \l__lipsum_a_int } { \l__lipsum_b_int } }
215       \str_set:Nx \l__lipsum_sep_set_str { \IfBooleanF {#1} { no } star }
216       \__lipsum_parse_sentence_range:eNN {#3} \l__lipsum_a_int \l__lipsum_b_int
217       \tl_set:Nx \l__lipsum_output_tl
218         {
219           \lipsum_get_sentences:nnV { \l__lipsum_a_int } { \l__lipsum_b_int }
220           \l__lipsum_output_tl
221         }
222     }
223     \str_set_eq:NN \l__lipsum_sep_set_str \l__lipsum_tmpa_str
224     \__lipsum_do:N \l__lipsum_output_tl
225   }
226 \cs_new_eq:NN \__lipsum_do:N ?
```

(End definition for `\__lipsum_do:nnnn` and `\__lipsum_do:N`.)

`\__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 re-defines `\__lipsum_restore_hyphens:` to reset the hyphenation patterns.

`\__lipsum_restore_hyphens:`

```
227 \cs_new_protected:Npn \__lipsum_set_hyphens:
```

```

228 {
229   \bool_if:NTF \l__lipsum_autolang_bool
230   { \use:n } { \use_none:n }
231   {
232     \cs_if_exist:NTF \hyphenrules
233     {
234       \cs_if_exist:cTF { ver@polyglossia.sty }
235       { \__lipsum_set_hyphens_polyglossia: }
236       { \__lipsum_set_hyphens_babel: }
237     }
238     { \__lipsum_set_hyphens_raw: }
239   }
240 }
241 \cs_new_protected:Npn \__lipsum_restore_hyphens:
242 { \prg_do_nothing: }

```

*(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:`.

```

243 \cs_new_protected:Npn \__lipsum_set_hyphens_babel:
244 {
245   \cs_if_exist:cTF { l@ \g__lipsum_language_tl }
246   {
247     \exp_args:NV \hyphenrules \g__lipsum_language_tl
248     \cs_set_protected:Npx \__lipsum_restore_hyphens:
249     { \exp_not:N \hyphenrules { \language } }
250   }
251   { \__lipsum_lang_not_available: }
252 }

```

*(End definition for \\_\_lipsum\_set\_hyphens\_babel:.)*

`\__lipsum_set_hyphens_polyglossia:` `polyglossia` less friendly. We also check if the language is loaded (looking at `\<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<sup>5</sup> when used with `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.

```

253 \cs_new_protected:Npn \__lipsum_set_hyphens_polyglossia:
254 {
255   \cs_if_exist:cTF { \g__lipsum_language_tl @loaded }
256   {
257     \exp_args:NnV \begin{hyphenrules} \g__lipsum_language_tl
258     \cs_set_protected:Npn \__lipsum_restore_hyphens:
259     { \end{hyphenrules} }

```

<sup>5</sup><https://github.com/PhelypeOleinik/lipsum/issues/1>

```

260     }
261     { \_lipsum_set_hyphens_raw: }
262 }

```

(End definition for \\_lipsum\_set\_hyphens\_polyglossia:.)

**\\_lipsum\_set\_hyphens\_raw:** 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.

```

263 \cs_new_protected:Npn \_lipsum_set_hyphens_raw:
264 {
265   \cs_if_exist:cTF { l@ \g_lipsum_language_tl }
266   {
267     \use:x
268     {
269       \language \use:c { l@ \g_lipsum_language_tl }
270       \cs_set_protected:Npn \_lipsum_restore_hyphens:
271       { \language \int_eval:n { \language } \scan_stop: }
272     }
273   }
274   { \_lipsum_lang_not_available: }
275 }

```

(End definition for \\_lipsum\_set\_hyphens\_raw:.)

**\\_lipsum\_lang\_not\_available:** If the requested language is for some reason unavailable, warn the user, then fall back to the current language.

```

276 \cs_new_protected:Npn \_lipsum_lang_not_available:
277 {
278   \msg_warning:nnx { lipsum } { missing-language }
279   { \g_lipsum_language_tl }
280   \tl_gset_eq:NN \g_lipsum_language_tl \languagename
281 }

```

(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.

```

282 \cs_new_protected:Npn \NewLipsumPar #1
283 {
284   \int_gincr:N \g_lipsum_par_int
285   \tl_gclear_new:c { g_lipsum_par_ \int_use:N \g_lipsum_par_int _tl }
286   \tl_gset:cn { g_lipsum_par_ \int_use:N \g_lipsum_par_int _tl } {#1}
287 }

```

(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.

```

288 \NewDocumentCommand \SetLipsumText { m }
289 {
290   \str_if_eq:VnF \g_lipsum_text_str {#1}
291   {

```



```

292     \tl_gset:Nn \g__lipsum_language_tl { english }
293     \int_gzero:N \g__lipsum_par_int
294     \file_input:n { #1.ltd }
295     \str_gset:Nn \g__lipsum_text_str {#1}
296   }
297 }

```

(End definition for `\SetLipsumText`. This function is documented on page 4.)

**`\SetLipsumLanguage`** This macro sets the language for hyphenation patterns of the dummy text. When a new lipsum text is read, this is reset.

```

298 \NewDocumentCommand \SetLipsumLanguage { m }
299 { \tl_gset:Nn \g__lipsum_language_tl {#1} }

```

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

## 5.5 Package options and defaults

`\LipsumRestoreParList` These are some auxiliaries for the package options and for setting up the default behaviour.

```

\LipsumRestoreSentenceList
\LipsumRestoreAll
299 \cs_new_protected:Npn \__lipsum_delim_restore:nnn #1 #2 #3
300 {
301   \keys_set:nn { lipsum }
302   {
303     #1-before = , #1-begin = , #1-end = , #1-after = ,
304     #1-before* = , #1-begin* = , #1-end* = , #1-after* = ,
305     #1-sep = {#2}, #1-sep* = {#3}
306   }
307 }
308 }
309 \cs_new_protected:Nn \__lipsum_restore_sentence_list:
310 { \__lipsum_delim_restore:nnn { sentence } { ~ } { ~ } }
311 \cs_new_eq:NN \__lipsum_restore_par_list: ?
312 \cs_new_protected:Npn \LipsumRestoreParList
313 {
314   \__lipsum_deprecated:n { LipsumRestoreParList }
315   \__lipsum_restore_par_list:
316 }
317 \cs_new_protected:Npn \LipsumRestoreSentenceList
318 {
319   \__lipsum_deprecated:n { LipsumRestoreSentenceList }
320   \__lipsum_restore_sentence_list:
321 }
322 \cs_new_protected:Npn \LipsumRestoreAll
323 {
324   \__lipsum_deprecated:n { LipsumRestoreAll }
325   \__lipsum_restore_par_list: \__lipsum_restore_sentence_list:
326 }

```

(End definition for `\LipsumRestoreParList` and others.)

**`\setlipsum`** Here are the options available at load-time and to `\setlipsum`.

```

327 \NewDocumentCommand \setlipsum { +m }
328 { \keys_set:nn { lipsum } {#1} }
329 \keys_define:nn { lipsum }
330 {

```

`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.

```

331     nopar .choice: ,
332     nopar / true .code:n =
333     {
334         \cs_gset_protected:Npn \__lipsum_restore_par_list:
335         { \__lipsum_delim_restore:nnn { par } { ~ } { \par } }
336     } ,
337     nopar / false .code:n =
338     {
339         \cs_gset_protected:Nn \__lipsum_restore_par_list:
340         { \__lipsum_delim_restore:nnn { par } { \par } { ~ } }
341     } ,
342     nopar .initial:n = false ,
343     nopar .default:n = true ,

```

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

```

344     auto-lang .bool_set:N = \l__lipsum_autolang_bool ,
345     auto-lang .initial:n = true ,
346     auto-lang .default:n = true ,

```

`text` just does `\SetLipsumText`. The initial value is not set here because this chunk of code is executed in `expl3` syntax, then `thetextloadswithoutspaces`, so `\setlipsum{text=lipsum}` is used later.

```

347     text .code:n = \SetLipsumText{#1} ,
348     text .value_required:n = true ,

```

`language` sets the language to be used when typesetting.

```

349     language .tl_gset:N = \g__lipsum_language_tl ,
350     language .value_required:n = true ,

```

`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 :)

```

351     default-range .code:n = \SetLipsumDefault{#1} ,
352     default-range .initial:n = 1-7 ,
353     default-range .default:n = 1-7 ,
354 }

```

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 `\l__lipsum_<thing>_<place>_#no]star_tl`.

```

355 \cs_set_protected:Npn \__lipsum_tmp:w #1 #2 #3
356 {
357     \keys_define:nn { lipsum }
358     {
359         #1-before #2 .tl_set:c = l__lipsum_#1_start           _#3star_tl ,
360         #1-begin  #2 .tl_set:c = l__lipsum_#1_itemstart      _#3star_tl ,
361         #1-sep    #2 .tl_set:c = l__lipsum_#1_itemseparator  _#3star_tl ,
362         #1-end    #2 .tl_set:c = l__lipsum_#1_itemend        _#3star_tl ,
363         #1-after  #2 .tl_set:c = l__lipsum_#1_end            _#3star_tl ,

```

```

364     }
365   }
366   \__lipsum_tmp:w { par } { } { no } \__lipsum_tmp:w { sentence } { } { no }
367   \__lipsum_tmp:w { par } * { } \__lipsum_tmp:w { sentence } * { }

```

(End definition for `\setlipsum`. This function is documented on page 4.)

Now turn `\ExplSyntaxOff` for a while, and load the default *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`).

```

368 \ExplSyntaxOff
369 \setlipsum{text=lipsum}
370 \ProcessKeysOptions{lipsum}
371 \ExplSyntaxOn
372 \__lipsum_restore_par_list:
373 \__lipsum_restore_sentence_list:

```

## 5.6 Messages

Now define the messages used throughout the package.

```

374 \msg_new:nnn { lipsum } { invalid-range }
375   { Invalid~number~or~range~'#1'. }
376 \msg_new:nnn { lipsum } { cmd-deprecated }
377   {
378     Command~'\iow_char:N\#1'~deprecated. \
379     See~the~lipsum~documentation~for~help.
380   }
381 \msg_new:nnn { lipsum } { missing-language }
382   {
383     Unknown~language~'#1'.~Hyphenation~patterns~for~
384     '\languagename'~will~be~used~instead.
385     \sys_if_engine_luatex:T
386     {
387       \ \
388       \cs_if_exist:cTF { ver@polyglossia.sty }
389       {
390         With~polyglossia,~you~have~to~explicitly~load~languages~
391         with~\iow_char:N\setotherlanguage{#1}~or~similar.
392       }
393       {
394         With~LuaTeX,~lipsum~requires~babel~to~get~proper~
395         hyphenation~(you~can~use~
396         \iow_char:N\usepackage[base]{babel}).
397       }
398     }
399   }
400 </package>

```