
/ This is \
 \ ducksay! /

\ \
 >(')
)/
 /(
 / '----/
 \ ~== /
 ~~~~

-----  
( But which Version? )  
-----

\ \  
 >()\_  
 (\_\_)\_\_

-----  
( v2.6 )  
-----

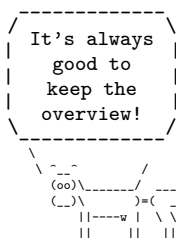
^\_\_^  
 /oo) \  
 /(\_\_) \  
 / \ \  
 | w----|  
 || ||

-----  
( by Jonathan P. Spratte )  
-----

/  
 .-----, /  
 .'\_/\_|\\_\'  
 /<::[0]8::>>\  
 |-----|  
 | | -==--== | |  
 | | ==--== | |  
 \ ||( )|::: | /  
 | ||( )|... | |  
 | |-----| |  
 | | \\_\_\_\_\_/ | |  
 / \ / \ / \ \  
 (\_\_\_\_) (\_\_\_\_) (\_\_\_\_)

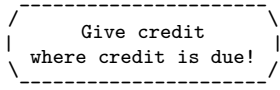
-----  
( Today is 2022-09-29 )  
-----

\ .\|//|\\|  
 \ |/\|/|/|/|/|  
 /. '|\|/|/|/|  
 o\_\_ \_|/|/|/|/|'

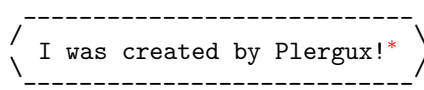


# Contents

|          |                                          |           |
|----------|------------------------------------------|-----------|
| <b>1</b> | <b>Acknowledgements</b>                  | <b>1</b>  |
| <b>2</b> | <b>Documentation</b>                     | <b>2</b>  |
| 2.1      | Downward Compatibility Issues            | 2         |
| 2.2      | Shared between versions                  | 2         |
| 2.2.1    | Macros                                   | 2         |
| 2.2.2    | Options                                  | 3         |
| 2.3      | Version 1                                | 6         |
| 2.3.1    | Introductkion                            | 6         |
| 2.3.2    | Macros                                   | 6         |
| 2.3.3    | Options                                  | 6         |
| 2.3.4    | Defects                                  | 7         |
| 2.4      | Version 2                                | 8         |
| 2.4.1    | Introductkion                            | 8         |
| 2.4.2    | Macros                                   | 8         |
| 2.4.3    | Options                                  | 8         |
| 2.5      | Dependencies                             | 14        |
| 2.6      | Available Animals                        | 14        |
| 2.7      | License and Bug Reports                  | 16        |
| <b>3</b> | <b>Implementation</b>                    | <b>17</b> |
| 3.1      | Shared between versions                  | 17        |
| 3.1.1    | Variables                                | 17        |
| 3.1.2    | Regular Expressions                      | 17        |
| 3.1.3    | Messages                                 | 17        |
| 3.1.4    | Key-value setup                          | 17        |
| 3.1.5    | Functions                                | 19        |
| 3.1.6    | Load the Correct Version and the Animals | 22        |
| 3.2      | Version 1                                | 23        |
| 3.2.1    | Functions                                | 23        |
| 3.3      | Version 2                                | 26        |
| 3.3.1    | Messages                                 | 26        |
| 3.3.2    | Variables                                | 26        |
| 3.3.3    | Options                                  | 26        |
| 3.3.4    | Functions                                | 29        |
| 3.4      | Definition of the Animals                | 38        |



# 1 Acknowledgements



\*<https://chat.stackexchange.com/transcript/message/55986902#55986902>

Just beest mod'rn,  
thee peasant!

## 2 Documentation



This is ducksay! A cowsay for L<sup>A</sup>T<sub>E</sub>X. ducksay is part of T<sub>E</sub>XLive and MiK<sub>T</sub>E<sub>X</sub> since September 2017. If it is not part of your installation it means that your L<sup>A</sup>T<sub>E</sub>X installation is *really* out of date, you have two options: Update your installation or try to install ducksay yourself. Chances are that if you opt for the latter, the version of expl3 in your L<sup>A</sup>T<sub>E</sub>X installation is too old, too, and the l3regex module is not yet part of expl3. In that case you'll get a few undefined control sequence errors. `\usepackage{l3regex}` prior to loading ducksay might fix these issues. Additionally you'll need **grabbox** for version 2 of ducksay that won't be part of your L<sup>A</sup>T<sub>E</sub>X installation, too. Please note that I don't actively support out of date L<sup>A</sup>T<sub>E</sub>X installations, so if loading l3regex doesn't fix the issues and you're on an old installation, I won't provide further support.

### 2.1 Downward Compatibility Issues

In the following list I use the term “version” to refer to package versions, the same is true if I use an abbreviation like “v2.0” (or anything that matches the regular expression `v\d+(\.d+)?`). For the code variant which can be set using the `version` option I'll use the term “variant” or specify directly that I'm referring to that option (the used font may be a hint, too).

- v2.0
  - Versions prior to v2.0 did use a regular expression for the option `ligatures`, see [subsection 2.2.2](#) for more on this issue.
  - In a document created with package versions prior to v2.0 you'll have to specify the option `version=1` with newer package versions to make those old documents behave like they used to.
- v2.3
  - Since v2.3 `\AddAnimal` and `\AddColoredAnimal` behave differently. You no longer have to make sure that in the first three lines every backslash which is only preceded by spaces is the bubble's tail. Instead you can specify which symbol should be the tail and how many of such symbols there are. See [subsection 2.2.1](#) for more about the current behaviour.
- v2.4
  - The `add-think` key was deprecated in v2.3 and was removed in v2.4 since the output symbols of the bubble tail are handled differently and more efficient now.

Macros for everyone!

### 2.2 Shared between versions

#### 2.2.1 Macros

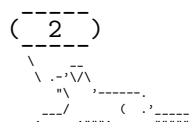
A careful reader might notice that in the below list of macros there is no `\ducksay` and no `\duckthink` contained. This is due to differences between the two usable code variants (see the `version` key in [subsection 2.2.2](#) for the code variants, [subsection 2.3.2](#) and [subsection 2.4.2](#) for descriptions of the two macros).

---

`\DefaultAnimal` `\DefaultAnimal{<animal>}`

---

use the `<animal>` if none is given in the optional argument to `\ducksay` or `\duckthink`. Package default is `duck`.



$$\backslash\text{DucksayOptions} \quad \backslash\text{DucksayOptions}\{\langle options \rangle\}$$

set the defaults to the keys described in [subsubsection 2.2.2](#), [subsubsection 2.3.3](#) and [subsubsection 2.4.3](#). Don't use an `<animal>` here, it has no effect.

`\AddAnimal \AddAnimal[*][options]{animal}{ascii-art}`

adds `<animal>` to the known animals. `<ascii-art>` is multi-line verbatim and therefore should be delimited either by matching braces or by anything that works for `\verb`. If the star is given `<animal>` is the new default. One space is added to the begin of `<animal>` (compensating the opening symbol). The symbols signaling the speech bubble's tail (in the `hedgehog` example below the two `s`) can be set using the `tail-symbol` option and only the first `tail-count` occurrences will be substituted (see [paragraph 2.2.2.1](#) for more about these options). For example, `hedgehog` is added with:

```
\AddAnimal[tail-symbol=s]{hedgehog}
{ s .\|//|\\|.
  s |/\|//|//|/|
    /. ‘|/\|//|//|
      o . |//|/|\\|’}
```

It is not checked whether the animal already exists, you could therefore redefine existing animals with this macro.

`\AddColoredAnimal \AddColoredAnimal[*][options]{animal}{ascii-art}`

It does the same as `\AddAnimal` but allows three different colouring syntaxes. You can use `\textcolor` in the `<ascii-art>` with the syntax `\textcolor{<color>}{<text>}`. Note that you can't use braces in the arguments of `\textcolor`.

You can also use a delimited `\color` of the form `\bgroup\color{color}\text\egroup`, a space after that `\egroup` will be considered a space in the output, so you don't have to care for correct termination of the `\egroup` (so `\bgroup\color{red}RedText\egroupOtherText` is valid syntax). You can't nest delimited `\colors`.

Also you can use an undelimited `\color`. It affects anything until the end of the current line (or, if used inside of the `\text` of a delimited `\color`, anything until the end of that delimited `\color`'s `\text`). The syntax would be `\color{color}`.

The package doesn't load anything providing those colouring commands for you and it doesn't provide any coloured animals. The parsing is done using regular expressions provided by L<sup>A</sup>T<sub>E</sub>X3. It is therefore slower than the normal `\AddAnimal`.

$$\backslash \text{AnimalOptions} \quad \backslash \text{AnimalOptions} \langle * \rangle \{ \langle \text{animal} \rangle \} \{ \langle \text{options} \rangle \}$$

With this macro you can set `<animal>` specific `<options>`. If the star is given any currently set options for this `<animal>` are dropped and only the ones specified in `<options>` will be applied, else `<options>` will be added to the set options for this `<animal>`. The set `<options>` can set the `tail-1` and `tail-2` options and therefore overwrite the effects of `\duckthink`, as `\duckthink` really is just `\ducksay` with the `think` option.

### 2.2.2 Options

The following options are available independent on the used code variant (the value of the `version` key). They might be used as package options – unless otherwise specified – or used in the macros `\DucksayOptions`, `\ducksay` and `\duckthink` – again unless otherwise specified. Some options might be accessible in both code variants but do

( 3 )

slightly different things. If that's the case they will be explained in [subsubsection 2.3.3](#) and [subsubsection 2.4.3](#) for **version 1** and **2**, respectively.

`version=<number>`

With this you can choose the code variant to be used. Currently **1** and **2** are available. This can be set only during package load time. For a dedicated description of each version look into [subsection 2.3](#) and [subsection 2.4](#). The package author would choose `version=2`, the other version is mostly for legacy reasons. The default is **2**.

`<animal>`

One of the animals listed in [subsection 2.6](#) or any of the ones added with `\AddAnimal`. Not useable as package option. Also don't use it in `\DucksayOptions`, it'll break the default animal selection.

`animal=<animal>`

Locally sets the default animal. Note that `\ducksay` and `\duckthink` do digest their options inside of a group, so it just results in a longer alternative to the use of `<animal>` if used in their options.

`ligatures=<token list>`

each token you don't want to form ligatures during `\AddAnimal` should be contained in this list. All of them get enclosed by grouping `{` and `}` so that they can't form ligatures. Giving no argument (or an empty one) might enhance compilation speed by disabling this replacement. The formation of ligatures was only observed in combination with `\usepackage[T1]{fontenc}` by the author of this package. Therefore giving the option `ligatures` without an argument might enhance the compilation speed for you without any drawbacks. Initially this is set to `'<>,-'`.

**Note:** In earlier releases this option's expected argument was a regular expression. This means that this option is not fully downward compatible with older versions. The speed gain however seems worth it (and I hope the affected documents are few).

`no-tail`

Sets `tail-1` and `tail-2` to be a space.

`random=<bool>`

If `true` a random animal will be used instead of the default one on each usage of `\ducksay` or `\duckthink`. The initial value is false and it defaults to true.

`say`

Sets `tail-1` and `tail-2` as backslashes.

`schroedinger`

Sets randomly either `animal=schroedinger-alive` or `animal=schroedinger-dead` at the time of use. Both have the same size, so this doesn't affect the needed space.

`tail-1=<token list>`

Sets the first tail symbol in the output to be `<token list>`. If set outside of `\ducksay` and `\duckthink` it will be overwritten inside of `\duckthink` to be `0`.

`tail-2=<token list>`

Sets every other tail symbol except the first one in the output to be `<token list>`. If set outside of `\ducksay` and `\duckthink` it will be overwritten inside of `\duckthink` to be `o`.

`think`

Sets `tail-1=0` and `tail-2=o`.

```
( 4 )
\ .-'\ \
  "\   '-----
   /      ( .)-----
  /-----) H H H )-----) H H H H H )
```

2.2.2.1 Options for \AddAnimal

The options described here are only available in \AddAnimal and \AddColoredAnimal.

- tail-count=<int>  
sets the number of tail symbols to be replaced in \AddAnimal and \AddColoredAnimal.  
Initial value is 2. If the value is negative every occurrence of tail-symbol will be replaced.
- tail-symbol=<str>  
the symbol used in \AddAnimal and \AddColoredAnimal to mark the bubble's tail. The  
argument gets \detokenized. Initially a single backslash.



## 2.3 Version 1

### 2.3.1 Introducktion

This version is included for legacy support (old documents should behave the same without any change to them – except the usage of `version=1` as an option, for a more or less complete list of downward compatibility related problems see [subsection 2.1](#)). For the bleeding edge version of `ducksay` skip this subsection and read [subsection 2.4](#).

### 2.3.2 Macros

The following is the description of macros which differ in behaviour from those of version 2.

`\ducksay` `\ducksay[⟨options⟩]{⟨message⟩}`

options might include any of the options described in [subsection 2.2.2](#) and [subsection 2.3.3](#) if not otherwise specified. Prints an `⟨animal⟩` saying `⟨message⟩`. `⟨message⟩` is not read in verbatim. Multi-line `⟨message⟩`s are possible using `\\`. `\\` should not be contained in a macro definition but at toplevel. Else use the option `ht`.

`\duckthink` `\duckthink[⟨options⟩]{⟨message⟩}`

options might include any of the options described in [subsection 2.2.2](#) and [subsection 2.3.3](#) if not otherwise specified. Prints an `⟨animal⟩` thinking `⟨message⟩`. `⟨message⟩` is not read in verbatim. Multi-line `⟨message⟩`s are possible using `\\`. `\\` should not be contained in a macro definition but at toplevel. Else use the option `ht`.

### 2.3.3 Options

The following options are available to `\ducksay`, `\duckthink`, and `\DucksayOptions` and if not otherwise specified also as package options:

`bubble=⟨code⟩`

use `⟨code⟩` in a group right before the bubble (for font switches). Might be used as a package option but not all control sequences work out of the box there.

`body=⟨code⟩` use `⟨code⟩` in a group right before the body (meaning the `⟨animal⟩`). Might be used as a package option but not all control sequences work out of the box there. E.g. to right-align the `⟨animal⟩` to the bubble, use `body=\hfill`.

`align=⟨valign⟩`

use `⟨valign⟩` as the vertical alignment specifier given to the `tabular` which is around the contents of `\ducksay` and `\duckthink`.

`msg-align=⟨halign⟩`

use `⟨halign⟩` for alignment of the rows of multi-line `⟨message⟩`s. It should match a `tabular` column specifier. Default is 1. It only affects the contents of the speech bubble not the bubble.

`rel-align=⟨column⟩`

use `⟨column⟩` for alignment of the bubble and the body. It should match a `tabular` column specifier. Default is 1.

( 6 )  
 \ .-'\ \  
 " \  
 \_ \_ \_ \_ \_ ( . ) \_ \_ \_ \_ \_  
 \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_ \_

**wd**= $\langle count \rangle$  in order to detect the width the  $\langle message \rangle$  is expanded. This might not work out for some commands (e.g. `\url` from `hyperref`). If you specify the width using **wd** the  $\langle message \rangle$  is not expanded and therefore the command *might* work out.  $\langle count \rangle$  should be the character count.

**ht**= $\langle count \rangle$  you might explicitly set the height (the row count) of the  $\langle message \rangle$ . This only has an effect if you also specify **wd**.

### 2.3.4 Defects

```

{
  Ohh, no!
}

```

- no automatic line wrapping
- message width detection based on token count with `\edef` expansion, might fail badly

```

( 7 )

```



Here's all the good stuff!

## 2.4 Version 2

### 2.4.1 Introductkion

Version 2 is the current version of `ducksay`. It features automatic line wrapping (if you specify a fixed width) and in general more options (with some nasty argument parsing).

If you're already used to version 1 you should note one important thing: You should only specify the `version` and the `ligatures` during package load time as arguments to `\usepackage`. The other keys might not work or do unintended things and only don't throw errors or warnings because of the legacy support of version 1. After the package is loaded, keys only used for version 1 will throw an error.

### 2.4.2 Macros

The following is the description of macros which differ in behaviour from those of version 1.

Look at those, kids!

---

`\ducksay` `\ducksay[⟨options⟩]{⟨message⟩}`

options might include any of the options described in [subsection 2.2.2](#) and [subsection 2.4.3](#) if not otherwise specified. Prints an `⟨animal⟩` saying `⟨message⟩`.

The `⟨message⟩` can be read in in four different ways. For an explanation of the `⟨message⟩` reading see the description of the `arg` key in [subsection 2.4.3](#).

The height and width of the message is determined by measuring its dimensions and the bubble will be set accordingly. The box surrounding the message will be placed both horizontally and vertically centred inside of the bubble. The output utilizes L<sup>A</sup>T<sub>E</sub>X3's coffin mechanism described in [interface3.pdf](#) and the documentation of `xcoffins`.

---

`\duckthink` `\duckthink[⟨options⟩]{⟨message⟩}`

The only difference to `\ducksay` is that in `\duckthink` the `⟨animal⟩`s think the `⟨message⟩` and don't say it.

Fast, use options!

### 2.4.3 Options

In version 2 the following options are available. Keep in mind that you shouldn't use them during package load time but in the arguments of `\ducksay`, `\duckthink` or `\DucksayOptions`.

`arg=⟨choice⟩`

specifies how the `⟨message⟩` argument of `\ducksay` and `\duckthink` should be read in. Available options are `box`, `tab` and `tab*`:

**box** the argument is read in either as a `\hbox` or a `\vbox` (the latter if a fixed width is specified with either `wd` or `wd*`). Note that in this mode any arguments relying on category code changes like e.g. `\verb` will work (provided that you don't use `\ducksay` or `\duckthink` inside of an argument of another macro of course).

**tab** the argument is read in as the contents of a `tabular`. Note that in this mode any arguments relying on category code changes like e.g. `\verb` will *not* work. This mode comes closest to the behaviour of version 1 of `ducksay`.

8

**tab\***

the argument is read in as the contents of a **tabular**. However it is read in verbatim and uses `\scantokens` to rescan the argument. Note that in this mode any arguments relying on category code changes like e.g. `\verb` will work. You can't use `\ducksay` or `\duckthink` as an argument to another macro in this mode however.

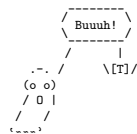
**b** shortcut for `out-v=b`.

`body=<font>` add `<font>` to the font definitions in use to typeset the `<animal>`'s body.

`body*=<font>`  
clear any definitions previously made (including the package default) and set the font definitions in use to typeset the `<animal>`'s body to `<font>`. The package default is `\verbatim@font`. In addition `\frenchspacing` will always be used prior to the defined `<font>`.

`body-align=<choice>`  
sets the relative alignment of the `<animal>` to the `<message>`. Possible choices are `l`, `c` and `r`. For `l` the `<animal>` is flushed to the left of the `<message>`, for `c` it is centred and for `r` it is flushed right. More fine grained control over the alignment can be obtained with the keys `msg-to-body`, `body-to-msg`, `body-x` and `body-y`. Package default is `l`.

`body-bigger=<count>`  
vertically enlarge the body by `<count>` empty lines added to the bottom. This way top-aligning two different body types is easier (by actually bottom aligning the two):



```
\ducksay[ghost,body-x=-7mm,b,body-mirrored]{Buuuh!}
\ducksay[crusader,body-bigger=4,b,out-h=r,no-bubble]{}
```

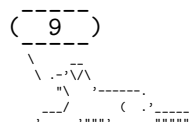
`body-mirrored=<bool>`  
if set true the `<animal>` will be mirrored along its vertical centre axis. Package default is `false`. If you set it `true` you'll most likely need to manually adjust the alignment of the body with one or more of the keys `body-align`, `body-to-msg`, `msg-to-body`, `body-x` and `body-y`.

`body-to-msg=<pole>`  
defines the horizontal coffin `<pole>` to be used for the placement of the `<animal>` beneath the `<message>`. See [interface3.pdf](#) and the documentation of `xcoffins` for information about coffin poles.

`body-x=<dimen>`  
defines a horizontal offset of `<dimen>` length of the `<animal>` from its placement beneath the `<message>`.

`body-y=<dimen>`  
defines a vertical offset of `<dimen>` length of the `<animal>` from its placement beneath the `<message>`.

`bubble=<font>`  
add `<font>` to the font definitions in use to typeset the bubble. This does not affect the `<message>` only the bubble put around it.



`bubble*=<font>`  
clear any definitions previously made (including the package default) and set the font definitions in use to typeset the bubble to *<font>*. This does not affect the *<message>* only the bubble put around it. The package default is `\verbatim@font`.

`bubble-bot-kern=<dimen>`  
specifies a vertical offset of the placement of the lower border of the bubble from the bottom of the left and right borders.

`bubble-delim-left-1=<token list>`  
the left delimiter used if only one line of delimiters is needed. Package default is `(`.

`bubble-delim-left-2=<token list>`  
the upper most left delimiter used if more than one line of delimiters is needed. Package default is `/`.

`bubble-delim-left-3=<token list>`  
the left delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.

`bubble-delim-left-4=<token list>`  
the lower most left delimiter used if more than one line of delimiters is needed. Package default is `\`.

`bubble-delim-right-1=<token list>`  
the right delimiter used if only one line of delimiters is needed. Package default is `)`.

`bubble-delim-right-2=<token list>`  
the upper most right delimiter used if more than one line of delimiters is needed. Package default is `\`.

`bubble-delim-right-3=<token list>`  
the right delimiters used to fill the gap if more than two lines of delimiters are needed. Package default is `|`.

`bubble-delim-right-4=<token list>`  
the lower most right delimiter used if more than one line of delimiters is needed. Package default is `/`.

`bubble-delim-top=<token list>`  
the delimiter used to create the top and bottom border of the bubble. The package default is `{-}` (the braces are important to suppress ligatures here).

`bubble-side-kern=<dimen>`  
specifies the kerning used to move the sideways delimiters added to fill the gap for more than two lines of bubble height. (the left one is moved to the left, the right one to the right)

`bubble-top-kern=<dimen>`  
specifies a vertical offset of the placement of the upper border of the bubble from the top of the left and right borders.

`c`  
shortcut for `out-v=vc`.

( 11 )

- `msg-align-r=<token list>`  
 set the `<token list>` which is responsible to typeset the message flushed right if the option `msg-align=r` is used. It is used independent of the `arg` key. For `arg=tab` and `arg=tab*` it is only used if a fixed width is specified and the macro `\arraybackslash` provided by `array` is used afterwards. The package default is `\raggedleft`. It might be useful if you want to use `ragged2e`'s `\RaggedLeft` for example.
- `msg-to-body=<pole>`  
 defines the horizontal coffin `<pole>` to be used as the reference point for the placement of the `<animal>` beneath the `<message>`. See [interface3.pdf](#) and the documentation of [xcoffins](#) for information about coffin poles.
- `no-bubble=<bool>`  
 If `true` the `<message>` will not be surrounded by a bubble. Package default is of course `false`.
- `none=<bool>` One could say this is a special animal. If `true` no animal body will be used (resulting in just the speech bubble). Package default is of course `false`.
- `out-h=<pole>`  
 defines the horizontal coffin `<pole>` to be used as the anchor point for the print out of the complete result of `\ducksay` and `\duckthink`. See [interface3.pdf](#) and the documentation of [xcoffins](#) for information about coffin poles.
- `out-v=<pole>`  
 defines the vertical coffin `<pole>` to be used as the anchor point for the print out of the complete result of `\ducksay` and `\duckthink`. See [interface3.pdf](#) and the documentation of [xcoffins](#) for information about coffin poles.
- `out-x=<dimen>`  
 specifies an additional horizontal offset of the print out of the complete result of `\ducksay` and `\duckthink`.
- `out-y=<dimen>`  
 specifies an additional vertical offset of the print out of the complete result of `\ducksay` and `\duckthink`.
- `strip-spaces=<bool>`  
 if set `true` leading and trailing spaces are stripped from the `<message>` if `arg=box` is used. Initially this is set to `false`.
- `t`  
 shortcut for `out-v=t`.
- `vpad=<count>`  
 add `<count>` to the lines used for the bubble, resulting in `<count>` more lines than necessary to enclose the `<message>` inside of the bubble.
- `wd=<count>` specifies the width of the `<message>` to be fixed to `<count>` times the width of an upper case M in the `<message>`'s font declaration. A value smaller than 0 is considered deactivated, else the width is considered as fixed. For a fixed width the argument of `\ducksay` and `\duckthink` is read in as a `\vbox` for `arg=box` and the column definition uses a p-type column for `arg=tab` and `arg=tab*`. If both `wd` is not smaller than 0 and `wd*` is not smaller than 0pt, `wd*` will take precedence.

```

( 12 )
-----
 \ .-'\ \
  " \   \
----- ( .) -----
) -----) -----) -----) -----)

```

`wd*=<dimen>` specifies the width of the *<message>* to be fixed to *<dimen>*. A value smaller than 0pt is considered deactivated, else the width is considered as fixed. For a fixed width the argument of `\ducksay` and `\duckthink` is read in as a `\vbox` for `arg=box` and the column definition uses a p-type column for `arg=tab` and `arg=tab*`. If both `wd` is not smaller than 0 and `wd*` is not smaller than 0pt, `wd*` will take precedence.

`wd-eq-body=<bool>`  
 if this is `true`, `wd` is smaller than 0, and `wd*` is smaller than 0pt the *<message>* will be as wide as the *<animal>*'s body. Note that because the *<animal>* bodies contain white space on their left end and due to the additional horizontal bubble delimiters the bubble will be wider than the *<animal>*'s body. If the `none` option was also used this option has no effect.

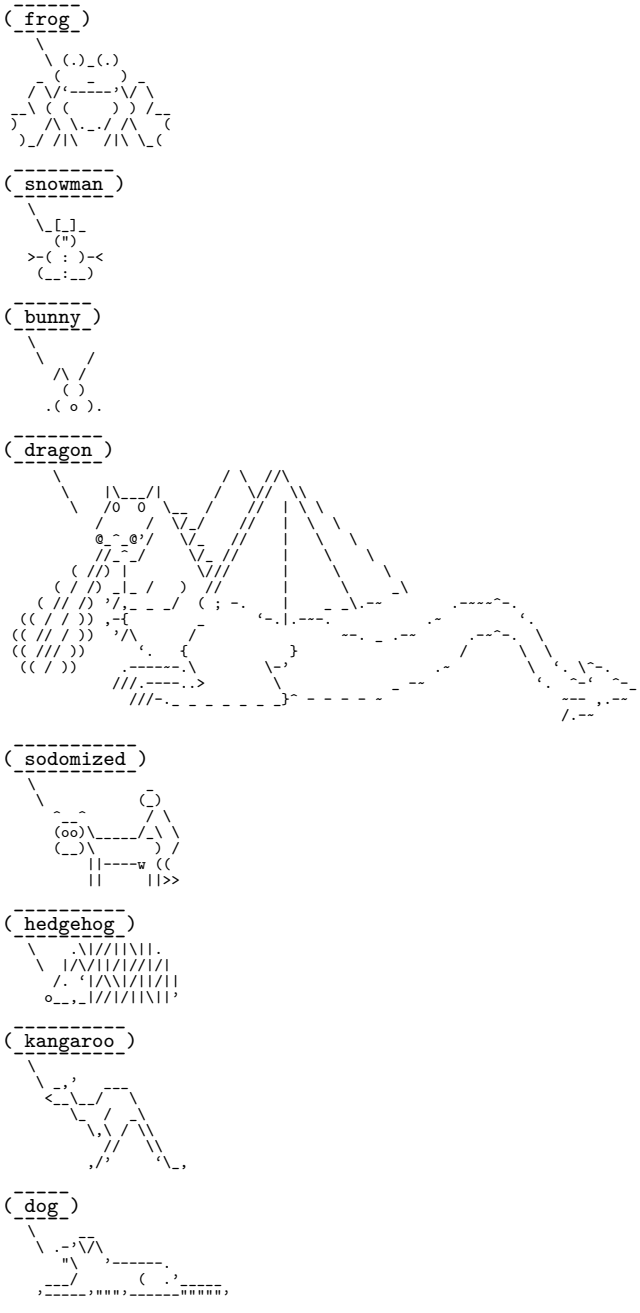
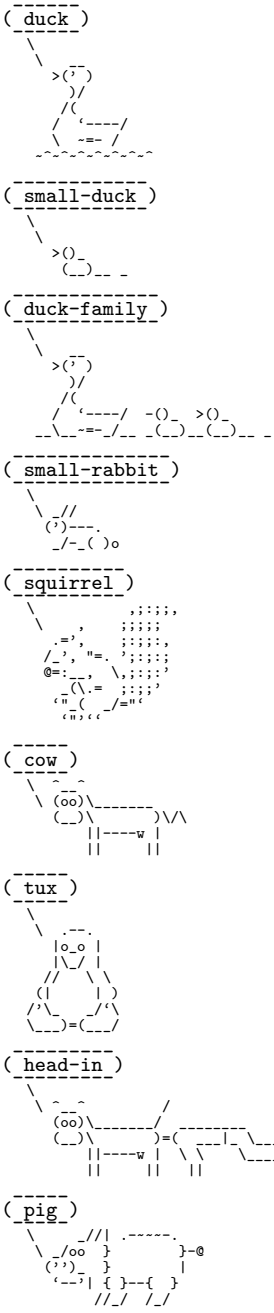


2.5 Dependencies

The package depends on the two packages `xparse` and `l3keys2e` and all of their dependencies. Version 2 additionally depends on `array` and `grabbbox`.

2.6 Available Animals

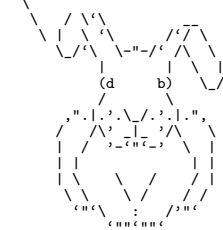
The following animals are provided by this package. I did not create them (but altered some), they belong to their original creators.



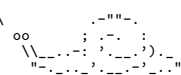
( sheep )



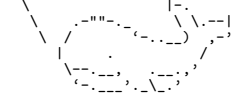
( rabbit )



( snail )



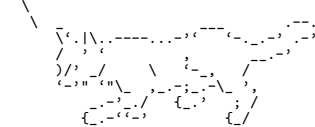
( whale )



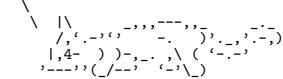
( snake )



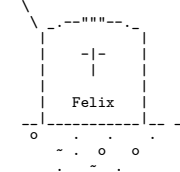
( cat )



( sleepy-cat )



( schroedinger-dead )



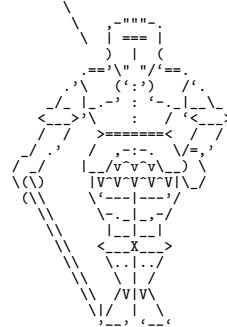
( schroedinger-alive )



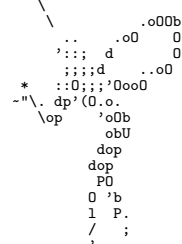
( crusader )



( knight )



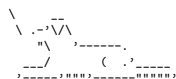
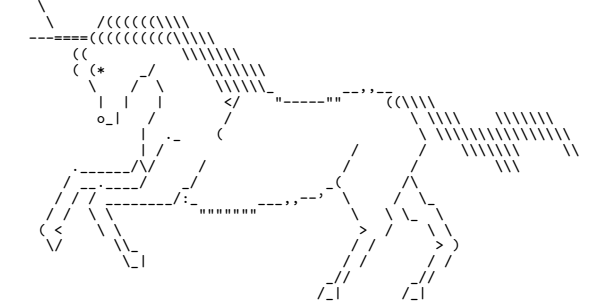
( fairy )



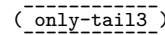
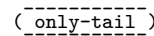
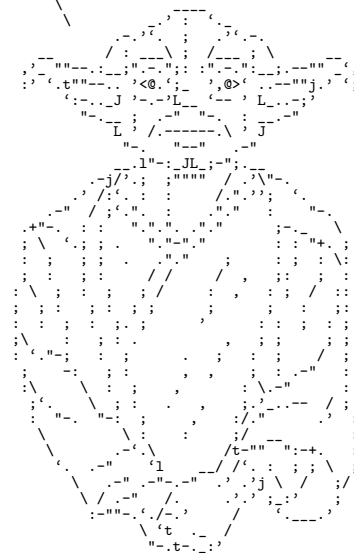
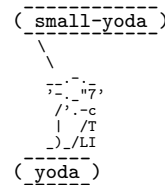
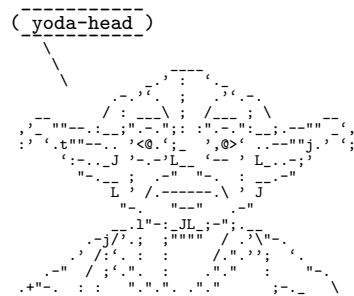
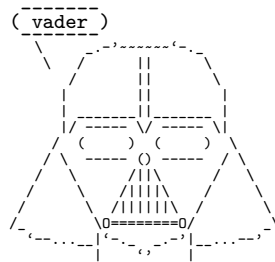
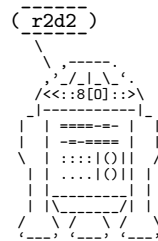
( ghost )



( unicorn )



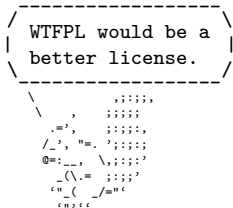


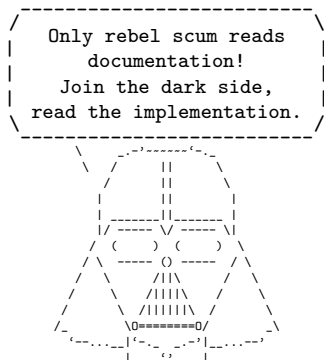


## 2.7 License and Bug Reports

This work may be distributed and/or modified under the conditions of the L<sup>A</sup>T<sub>E</sub>X Project Public License (LPPL), either version 1.3c of this license or (at your option) any later version. The latest version of this license is in the file: <http://www.latex-project.org/lppl.txt>

The package is hosted on [https://github.com/Skillmon/ltx\\_ducksay](https://github.com/Skillmon/ltx_ducksay), you might report bugs there.





## 3 Implementation

1 `<*pkg>`

### 3.1 Shared between versions

#### 3.1.1 Variables

##### 3.1.1.1 Integers

2 `\int_new:N \l_ducksay_msg_width_int`  
 3 `\int_new:N \l_ducksay_msg_height_int`  
 4 `\int_new:N \l_ducksay_tail_symbol_count_int`

##### 3.1.1.2 Sequences

5 `\seq_new:N \l_ducksay_msg_lines_seq`  
 6 `\seq_new:N \l_ducksay_defined_animals_seq`

##### 3.1.1.3 Token lists

7 `\tl_new:N \l_ducksay_align_tl`  
 8 `\tl_new:N \l_ducksay_msg_align_tl`  
 9 `\tl_new:N \l_ducksay_animal_tl`  
 10 `\tl_new:N \l_ducksay_body_tl`  
 11 `\tl_new:N \l_ducksay_bubble_tl`  
 12 `\tl_new:N \l_ducksay_tmpa_tl`  
 13 `\tl_new:N \l_ducksay_tail_symbol_out_one_tl`  
 14 `\tl_new:N \l_ducksay_tail_symbol_out_two_tl`  
 15 `\tl_new:N \l_ducksay_tail_symbol_in_tl`

##### 3.1.1.4 Boolean

16 `\bool_new:N \l_ducksay_version_one_bool`  
 17 `\bool_new:N \l_ducksay_version_two_bool`  
 18 `\bool_new:N \l_ducksay_random_animal_bool`

##### 3.1.1.5 Boxes

19 `\box_new:N \l_ducksay_tmpa_box`

### 3.1.2 Regular Expressions

Regular expressions for `\AddColoredAnimal`

20 `\regex_const:Nn \c_ducksay_textcolor_regex`  
 21 `{ \c0(?:\\textcolor\{(.*)\}\{(.*)\}) }`  
 22 `\regex_const:Nn \c_ducksay_color_delim_regex`  
 23 `{ \c0(?:\\bgroup\\color\{(.*)\}(.*))\\egroup }`  
 24 `\regex_const:Nn \c_ducksay_color_regex`  
 25 `{ \c0(?:\\color\{(.*)\}) }`

### 3.1.3 Messages

26 `\msg_new:nnn { ducksay } { load-time-only }`  
 27 `{ The~'#1'~key-is-to-be-used-only~during-package-load-time. }`

### 3.1.4 Key-value setup

28 `\keys_define:nn { ducksay }`  
 29 `{`  
 30  `,bubble .tl_set:N = \l_ducksay_bubble_tl`  
 31  `,body .tl_set:N = \l_ducksay_body_tl`

( 17 )  
 \ .-'\V\-----  
 " \ '-----  
 \_--/ ( '-----  
 }-----}-----}



```

84 \keys_define:nn { ducksay }
85 {
86   version .code:n = \msg_error:nnn { ducksay } { load-time-only } { version }
87 }

```

#### 3.1.4.1 Keys for \AddAnimal

Define keys meant for \AddAnimal and \AddColoredAnimal only in their own regime:

```

88 \keys_define:nn { ducksay / add-animal }
89 {
90   ,tail-symbol .code:n =
91     \tl_set:Nx \l_ducksay_tail_symbol_in_tl { \tl_to_str:n { #1 } }
92   ,tail-symbol .initial:o = \c_backslash_str
93   ,tail-count .int_set:N = \l_ducksay_tail_symbol_count_int
94   ,tail-count .initial:n = 2
95 }

```

### 3.1.5 Functions

#### 3.1.5.1 Generating Variants of External Functions

```

96 \cs_generate_variant:Nn \tl_replace_once:Nnn { NVn }
97 \cs_generate_variant:Nn \tl_replace_all:Nnn { NVn }
98 \cs_generate_variant:Nn \keys_set:nn { nx }

```

#### 3.1.5.2 Internal

\\_ducksay\_everyeof:w

```

99 \cs_set_eq:NN \_ducksay_everyeof:w \tex_everyeof:D

```

(End definition for \\_ducksay\_everyeof:w.)

\\_ducksay\_scantokens:w

```

100 \cs_set_eq:NN \_ducksay_scantokens:w \tex_scantokens:D

```

(End definition for \\_ducksay\_scantokens:w.)

\ducksay\_replace\_verb\_newline:Nn

```

101 \cs_new_protected:Npx \ducksay_replace_verb_newline:Nn #1 #2
102 {
103   \tl_replace_all:Nnn #1 { \char_generate:nn { 13 } { 12 } } { #2 }
104 }

```

(End definition for \ducksay\_replace\_verb\_newline:Nn.)

\ducksay\_replace\_verb\_newline\_newline:Nn

```

105 \cs_new_protected:Npx \ducksay_replace_verb_newline_newline:Nn #1 #2
106 {
107   \tl_replace_all:Nnn #1
108     { \char_generate:nn { 13 } { 12 } \char_generate:nn { 13 } { 12 } } { #2 }
109 }

```

(End definition for \ducksay\_replace\_verb\_newline\_newline:Nn.)

\ducksay\_process\_verb\_newline:nnn

```
110 \cs_new_protected:Npn \ducksay_process_verb_newline:nnn #1 #2 #3
111 {
112   \tl_set:Nn \ProcessedArgument { #3 }
113   \ducksay_replace_verb_newline_newline:Nn \ProcessedArgument { #2 }
114   \ducksay_replace_verb_newline:Nn \ProcessedArgument { #1 }
115 }
```

(End definition for \ducksay\_process\_verb\_newline:nnn.)

\ducksay\_add\_animal\_inner:nnnn

```
116 \cs_new_protected:Npn \ducksay_add_animal_inner:nnnn #1 #2 #3 #4
117 {
118   \group_begin:
119   \keys_set:nn { ducksay / add-animal } { #1 }
120   \tl_set:Nn \l_ducksay_tmpa_tl { \ #3 }
121   \int_compare:nNnTF { \l_ducksay_tail_symbol_count_int } < { \c_zero_int }
122   {
123     \tl_replace_once:NVn
124     \l_ducksay_tmpa_tl
125     \l_ducksay_tail_symbol_in_tl
126     \l_ducksay_tail_symbol_out_one_tl
127     \tl_replace_all:NVn
128     \l_ducksay_tmpa_tl
129     \l_ducksay_tail_symbol_in_tl
130     \l_ducksay_tail_symbol_out_two_tl
131   }
132   {
133     \int_compare:nNnT { \l_ducksay_tail_symbol_count_int } >
134     { \c_zero_int }
135     {
136       \tl_replace_once:NVn
137       \l_ducksay_tmpa_tl
138       \l_ducksay_tail_symbol_in_tl
139       \l_ducksay_tail_symbol_out_one_tl
140       \int_step_inline:nnn { 2 } { \l_ducksay_tail_symbol_count_int }
141       {
142         \tl_replace_once:NVn
143         \l_ducksay_tmpa_tl
144         \l_ducksay_tail_symbol_in_tl
145         \l_ducksay_tail_symbol_out_two_tl
146       }
147     }
148   }
149   \tl_map_inline:Nn \l_ducksay_ligatures_tl
150   { \tl_replace_all:Nnn \l_ducksay_tmpa_tl { ##1 } { { ##1 } } }
151   \ducksay_replace_verb_newline:Nn \l_ducksay_tmpa_tl
152   { \tabularnewline\null }
153   \exp_args:NNnV
154   \group_end:
155   \tl_set:cn { l_ducksay_animal_#2_tl } \l_ducksay_tmpa_tl
156   \exp_args:Nnx \keys_define:nn { ducksay }
157   {
158     #2 .code:n =
```

( 20 )  
-----  
 \ .-' \ /  
 " \ '-----  
 / ( '-----  
 )-----)-----)-----)

```

159         {
160             \exp_not:n { \tl_set_eq:NN \l_ducksay_animal_tl }
161             \exp_not:c { l_ducksay_animal_#2_tl }
162             \exp_not:n { \exp_args:NV \DucksayOptions }
163             \exp_not:c { l_ducksay_animal_#2_options_tl }
164         }
165     }
166     \tl_if_exist:cF { l_ducksay_animal_#2_options_tl }
167     { \tl_new:c { l_ducksay_animal_#2_options_tl } }
168     \IfBooleanT { #4 }
169     { \keys_define:nn { ducksay } { default_animal .meta:n = { #2 } } }
170     \seq_if_in:NnF \l_ducksay_defined_animals_seq { #2 }
171     { \seq_push:Nn \l_ducksay_defined_animals_seq { #2 } }
172 }
173 \cs_generate_variant:Nn \ducksay_add_animal_inner:nmmm { nnVn }

```

(End definition for \ducksay\_add\_animal\_inner:nmmm.)

\ducksay\_default\_or\_random\_animal:

```

174 \cs_new_protected:Npn \ducksay_default_or_random_animal:
175 {
176     \tl_if_empty:NT \l_ducksay_animal_tl
177     {
178         \bool_if:NTF \l_ducksay_random_animal_bool
179         {
180             \keys_set:nx { ducksay }
181             { \seq_rand_item:N \l_ducksay_defined_animals_seq }
182         }
183         { \keys_set:nn { ducksay } { default_animal } }
184     }
185 }

```

(End definition for \ducksay\_default\_or\_random\_animal:.)

### 3.1.5.3 Document level

**\DefaultAnimal**

```

186 \NewDocumentCommand \DefaultAnimal { m }
187 {
188     \keys_define:nn { ducksay } { default_animal .meta:n = { #1 } }
189 }

```

(End definition for \DefaultAnimal. This function is documented on page 2.)

**\DucksayOptions**

```

190 \NewDocumentCommand \DucksayOptions { m }
191 {
192     \keys_set:nn { ducksay } { #1 }
193 }

```

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

### **\AddAnimal**

```
194 \NewDocumentCommand \AddAnimal { s O{} m +v }
195 {
196   \ducksay_add_animal_inner:nnnn { #2 } { #3 } { #4 } { #1 }
197 }
```

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

### **\AddColoredAnimal**

```
198 \NewDocumentCommand \AddColoredAnimal { s O{} m +v }
199 {
200   \tl_set:Nn \l_ducksay_tmpa_tl { #4 }
201   \regex_replace_all:NnN \c_ducksay_color_delim_regex
202     { \c{bgroup}\c{color}\cB{\1\cE}\2\c{egroup} }
203     \l_ducksay_tmpa_tl
204   \regex_replace_all:NnN \c_ducksay_color_regex
205     { \c{color}\cB{\1\cE}\2 }
206     \l_ducksay_tmpa_tl
207   \regex_replace_all:NnN \c_ducksay_textcolor_regex
208     { \c{textcolor}\cB{\1\cE}\2\cE }
209     \l_ducksay_tmpa_tl
210   \ducksay_add_animal_inner:nnVn { #2 } { #3 } \l_ducksay_tmpa_tl { #1 }
211 }
```

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

### **\AnimalOptions**

```
212 \NewDocumentCommand \AnimalOptions { s m m }
213 {
214   \tl_if_exist:cTF { l_ducksay_animal_#2_options_tl }
215     {
216       \IfBooleanTF { #1 }
217         { \tl_set:cn }
218         { \tl_put_right:cn }
219     }
220     { \tl_set:cn }
221     { l_ducksay_animal_#2_options_tl } { #3, }
222 }
```

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

## **3.1.6 Load the Correct Version and the Animals**

```
223 \bool_if:NT \l_ducksay_version_one_bool
224   { \file_input:n { ducksay.code.v1.tex } }
225 \bool_if:NT \l_ducksay_version_two_bool
226   { \file_input:n { ducksay.code.v2.tex } }
227 \ExplSyntaxOff
228 \input{ducksay.animals.tex}
229 </pkg>
```

## 3.2 Version 1

```

230 <*code.v1>
231 \ProvidesFile{ducksay.code.v1.tex}
232 [\ducksay@date\space v\ducksay@version\space ducksay code version 1]

```

### 3.2.1 Functions

#### 3.2.1.1 Internal

\ducksay\_longest\_line:n Calculate the length of the longest line

```

233 \cs_new:Npn \ducksay_longest_line:n #1
234 {
235   \int_incr:N \l_ducksay_msg_height_int
236   \exp_args:NNx \tl_set:Nn \l_ducksay_tmpa_tl { #1 }
237   \regex_replace_all:nnN { \s } { \c { space } } \l_ducksay_tmpa_tl
238   \int_set:Nn \l_ducksay_msg_width_int
239   {
240     \int_max:nn
241     { \l_ducksay_msg_width_int } { \tl_count:N \l_ducksay_tmpa_tl }
242   }
243 }

```

(End definition for \ducksay\_longest\_line:n.)

\ducksay\_open\_bubble: Draw the opening bracket of the bubble

```

244 \cs_new:Npn \ducksay_open_bubble:
245 {
246   \begin{tabular}{@{}l@{}}
247     \null\
248     \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 } { ( }
249     {
250       /
251       \int_step_inline:nnn
252       { 3 } { \l_ducksay_msg_height_int } { \\ \kern-0.2em| }
253       \\ \detokenize{ \ }
254     }
255     \\ [-1ex] \null
256   \end{tabular}
257   \begin{tabular}{@{}l@{}}
258     _\\
259     \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\ [-1ex]
260     \mbox { - }
261   \end{tabular}
262 }

```

(End definition for \ducksay\_open\_bubble:.)

\ducksay\_close\_bubble: Draw the closing bracket of the bubble

```

263 \cs_new:Npn \ducksay_close_bubble:
264 {
265   \begin{tabular}{@{}l@{}}
266     _\\
267     \int_step_inline:nnn { 2 } { \l_ducksay_msg_height_int } { \\ } \\ [-1ex]
268     { - }
269   \end{tabular}

```

```

( 23 )
\ .-'\
" \
----- ( .) -----
) -----) -----) -----)

```



```

270 \begin{tabular}{@{}r@{}}
271 \null\
272 \int_compare:nNnTF { \l_ducksay_msg_height_int } = { 1 }
273 { ) }
274 {
275 \detokenize {\ }
276 \int_step_inline:nnn
277 { 3 } { \l_ducksay_msg_height_int } { \|\kern-0.2em }
278 \\\
279 }
280 \\\[-1ex]\null
281 \end{tabular}
282 }

```

(End definition for \ducksay\_close\_bubble:.)

\ducksay\_print\_msg:nn Print out the message

```

283 \cs_new:Npn \ducksay_print_msg:nn #1 #2
284 {
285 \begin{tabular}{@{} #2 @{}}
286 \int_step_inline:nn { \l_ducksay_msg_width_int } { _ } \\\
287 #1\|[-1ex]
288 \int_step_inline:nn { \l_ducksay_msg_width_int } { { - } }
289 \end{tabular}
290 }
291 \cs_generate_variant:Nn \ducksay_print_msg:nn { nV }

```

(End definition for \ducksay\_print\_msg:nn.)

\ducksay\_print:nn Print out the whole thing

```

292 \cs_new:Npn \ducksay_print:nn #1 #2
293 {
294 \int_compare:nNnTF { \l_ducksay_msg_width_int } < { 0 }
295 {
296 \int_zero:N \l_ducksay_msg_height_int
297 \seq_set_split:Nnn \l_ducksay_msg_lines_seq { \\\ } { #1 }
298 \seq_map_function:NN \l_ducksay_msg_lines_seq \ducksay_longest_line:n
299 }
300 {
301 \int_compare:nNnT { \l_ducksay_msg_height_int } < { 0 }
302 {
303 \regex_count:nnN { \c { \\\ } } { #1 } \l_ducksay_msg_height_int
304 \int_incr:N \l_ducksay_msg_height_int
305 }
306 }
307 \group_begin:
308 \frenchspacing
309 \verbatim@font
310 \@noligs
311 \begin{tabular}[\l_ducksay_align_tl]{@{}#2@{}}
312 \l_ducksay_bubble_tl
313 \begin{tabular}{@{}l@{}}
314 \ducksay_open_bubble:
315 \ducksay_print_msg:nV { #1 } \l_ducksay_msg_align_tl
316 \ducksay_close_bubble:

```

```

( 24 )
\ .-'\
" \
----- ( .)-----
)-----)

```

```

317         \end{tabular}\\
318         \l_ducksay_body_tl
319         \begin{tabular}{@{}l@{}}
320             \l_ducksay_animal_tl
321         \end{tabular}
322     \end{tabular}
323 \group_end:
324 }
325 \cs_generate_variant:Nn \ducksay_print:nn { nV }

```

(End definition for \ducksay\_print:nn.)

\ducksay\_say\_and\_think:nn Reset some variables

```

326 \cs_new:Npn \ducksay_say_and_think:nn #1 #2
327 {
328     \group_begin:
329     \int_set:Nn \l_ducksay_msg_width_int { -\c_max_int }
330     \int_set:Nn \l_ducksay_msg_height_int { -\c_max_int }
331     \keys_set:nn { ducksay } { #1 }
332     \ducksay_default_or_random_animal:
333     \ducksay_print:nV { #2 } \l_ducksay_rel_align_tl
334 \group_end:
335 }

```

(End definition for \ducksay\_say\_and\_think:nn.)

### 3.2.1.2 Document level

**\ducksay**

```

336 \NewDocumentCommand \ducksay { 0{} m }
337 {
338     \ducksay_say_and_think:nn { #1 } { #2 }
339 }

```

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

**\duckthink**

```

340 \NewDocumentCommand \duckthink { 0{} m }
341 {
342     \ducksay_say_and_think:nn { think, #1 } { #2 }
343 }

```

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

344 </code.v1>

### 3.3 Version 2

```

345 <*code.v2>
346 \ProvidesFile{ducksay.code.v2.tex}
347 [\ducksay@date\space v\ducksay@version\space ducksay code version 2]

Load the additional dependencies of version 2.
348 \RequirePackage{array,grabbox}

```

#### 3.3.1 Messages

```

349 \msg_new:nnn { ducksay } { justify~unavailable }
350 {
351   Justified~content~is~not~available~for~tabular~argument~mode~without~fixed~
352   width.~'l'~column~is~used~instead.
353 }
354 \msg_new:nnn { ducksay } { unknown~message~alignment }
355 {
356   The~specified~message~alignment~'\exp_not:n { #1 }'~is~unknown.~
357   'l'~is~used~as~fallback.
358 }
359 \msg_new:nnn { ducksay } { v1~key~only }
360 { The~'\l_keys_key_tl'~key~is~only~available~for~'version=1'. }
361 \msg_new:nnn { ducksay } { zero~baselineskip }
362 { Current~ baselineskip~ is~ 0pt~ (fallback~ #1~ used). }

```

#### 3.3.2 Variables

##### 3.3.2.1 Token Lists

```

363 \tl_new:N \l_ducksay_msg_align_vbox_tl

```

##### 3.3.2.2 Boxes

```

364 \box_new:N \l_ducksay_msg_box

```

##### 3.3.2.3 Bools

```

365 \bool_new:N \l_ducksay_eat_arg_box_bool
366 \bool_new:N \l_ducksay_eat_arg_tab_verb_bool
367 \bool_new:N \l_ducksay_mirrored_body_bool
368 \bool_new:N \l_ducksay_msg_eq_body_width_bool

```

##### 3.3.2.4 Coffins

```

369 \coffin_new:N \l_ducksay_body_coffin
370 \coffin_new:N \l_ducksay_bubble_close_coffin
371 \coffin_new:N \l_ducksay_bubble_open_coffin
372 \coffin_new:N \l_ducksay_bubble_top_coffin
373 \coffin_new:N \l_ducksay_msg_coffin

```

##### 3.3.2.5 Dimensions

```

374 \dim_new:N \l_ducksay_hpad_dim
375 \dim_new:N \l_ducksay_bubble_bottom_kern_dim
376 \dim_new:N \l_ducksay_bubble_top_kern_dim
377 \dim_new:N \l_ducksay_msg_width_dim

```

#### 3.3.3 Options

```

378 \keys_define:nn { ducksay }
379 {
380   ,arg .choice:

```

```

381 ,arg / box .code:n = \bool_set_true:N \l_ducksay_eat_arg_box_bool
382 ,arg / tab .code:n =
383 {
384     \bool_set_false:N \l_ducksay_eat_arg_box_bool
385     \bool_set_false:N \l_ducksay_eat_arg_tab_verb_bool
386 }
387 ,arg / tab* .code:n =
388 {
389     \bool_set_false:N \l_ducksay_eat_arg_box_bool
390     \bool_set_true:N \l_ducksay_eat_arg_tab_verb_bool
391 }
392 ,arg .initial:n = tab
393 ,wd* .dim_set:N = \l_ducksay_msg_width_dim
394 ,wd* .initial:n = -\c_max_dim
395 ,wd* .value_required:n = true
396 ,wd-eq-body .bool_set:N = \l_ducksay_msg_eq_body_width_bool
397 ,none .bool_set:N = \l_ducksay_no_body_bool
398 ,no-bubble .bool_set:N = \l_ducksay_no_bubble_bool
399 ,body-mirrored .bool_set:N = \l_ducksay_mirrored_body_bool
400 ,ignore-body .bool_set:N = \l_ducksay_ignored_body_bool
401 ,body-x .dim_set:N = \l_ducksay_body_x_offset_dim
402 ,body-x .value_required:n = true
403 ,body-y .dim_set:N = \l_ducksay_body_y_offset_dim
404 ,body-y .value_required:n = true
405 ,body-to-msg .tl_set:N = \l_ducksay_body_to_msg_align_body_tl
406 ,msg-to-body .tl_set:N = \l_ducksay_body_to_msg_align_msg_tl
407 ,body-align .choice:
408 ,body-align / l .meta:n = { body-to-msg = l , msg-to-body = l }
409 ,body-align / c .meta:n = { body-to-msg = hc , msg-to-body = hc }
410 ,body-align / r .meta:n = { body-to-msg = r , msg-to-body = r }
411 ,body-align .initial:n = l
412 ,body-bigger .int_set:N = \l_ducksay_body_bigger_int
413 ,body-bigger .initial:n = \c_zero_int
414 ,msg-align .choice:
415 ,msg-align / l .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { l } }
416 ,msg-align / c .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { c } }
417 ,msg-align / r .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { r } }
418 ,msg-align / j .code:n = { \tl_set:Nn \l_ducksay_msg_align_tl { j } }
419 ,msg-align-l .tl_set:N = \l_ducksay_msg_align_l_tl
420 ,msg-align-l .initial:n = \raggedright
421 ,msg-align-c .tl_set:N = \l_ducksay_msg_align_c_tl
422 ,msg-align-c .initial:n = \centering
423 ,msg-align-r .tl_set:N = \l_ducksay_msg_align_r_tl
424 ,msg-align-r .initial:n = \raggedleft
425 ,msg-align-j .tl_set:N = \l_ducksay_msg_align_j_tl
426 ,msg-align-j .initial:n = {}
427 ,out-h .tl_set:N = \l_ducksay_output_h_pole_tl
428 ,out-h .initial:n = l
429 ,out-v .tl_set:N = \l_ducksay_output_v_pole_tl
430 ,out-v .initial:n = vc
431 ,out-x .dim_set:N = \l_ducksay_output_x_offset_dim
432 ,out-x .value_required:n = true
433 ,out-y .dim_set:N = \l_ducksay_output_y_offset_dim
434 ,out-y .value_required:n = true

```

```

435 ,t      .meta:n      = { out-v = t }
436 ,c      .meta:n      = { out-v = vc }
437 ,b      .meta:n      = { out-v = b }
438 ,body*   .tl_set:N    = \l_ducksay_body_fount_tl
439 ,msg*    .tl_set:N    = \l_ducksay_msg_fount_tl
440 ,bubble* .tl_set:N    = \l_ducksay_bubble_fount_tl
441 ,body*   .initial:n   = \verbatim@font
442 ,msg*    .initial:n   = \verbatim@font
443 ,bubble* .initial:n   = \verbatim@font
444 ,body    .code:n      = \tl_put_right:Nn \l_ducksay_body_fount_tl { #1 }
445 ,msg     .code:n      = \tl_put_right:Nn \l_ducksay_msg_fount_tl { #1 }
446 ,bubble .code:n      = \tl_put_right:Nn \l_ducksay_bubble_fount_tl { #1 }
447 ,MSG     .meta:n      = { msg = #1 , bubble = #1 }
448 ,MSG*    .meta:n      = { msg* = #1 , bubble* = #1 }
449 ,hpad    .int_set:N   = \l_ducksay_hpad_int
450 ,hpad    .initial:n   = 2
451 ,hpad    .value_required:n = true
452 ,vpad    .int_set:N   = \l_ducksay_vpad_int
453 ,vpad    .value_required:n = true
454 ,col     .tl_set:N    = \l_ducksay_msg_tabular_column_tl
455 ,bubble-top-kern .tl_set:N = \l_ducksay_bubble_top_kern_tl
456 ,bubble-top-kern .initial:n = { -.5ex }
457 ,bubble-top-kern .value_required:n = true
458 ,bubble-bot-kern .tl_set:N = \l_ducksay_bubble_bottom_kern_tl
459 ,bubble-bot-kern .initial:n = { .2ex }
460 ,bubble-bot-kern .value_required:n = true
461 ,bubble-side-kern .tl_set:N = \l_ducksay_bubble_side_kern_tl
462 ,bubble-side-kern .initial:n = { .2em }
463 ,bubble-side-kern .value_required:n = true
464 ,bubble-delim-top .tl_set:N = \l_ducksay_bubble_delim_top_tl
465 ,bubble-delim-left-1 .tl_set:N = \l_ducksay_bubble_delim_left_a_tl
466 ,bubble-delim-left-2 .tl_set:N = \l_ducksay_bubble_delim_left_b_tl
467 ,bubble-delim-left-3 .tl_set:N = \l_ducksay_bubble_delim_left_c_tl
468 ,bubble-delim-left-4 .tl_set:N = \l_ducksay_bubble_delim_left_d_tl
469 ,bubble-delim-right-1 .tl_set:N = \l_ducksay_bubble_delim_right_a_tl
470 ,bubble-delim-right-2 .tl_set:N = \l_ducksay_bubble_delim_right_b_tl
471 ,bubble-delim-right-3 .tl_set:N = \l_ducksay_bubble_delim_right_c_tl
472 ,bubble-delim-right-4 .tl_set:N = \l_ducksay_bubble_delim_right_d_tl
473 ,bubble-delim-top .initial:n = { { - } }
474 ,bubble-delim-left-1 .initial:n = (
475 ,bubble-delim-left-2 .initial:n = /
476 ,bubble-delim-left-3 .initial:n = |
477 ,bubble-delim-left-4 .initial:n = \c_backslash_str
478 ,bubble-delim-right-1 .initial:n = )
479 ,bubble-delim-right-2 .initial:n = \c_backslash_str
480 ,bubble-delim-right-3 .initial:n = |
481 ,bubble-delim-right-4 .initial:n = /
482 ,strip-spaces .bool_set:N = \l_ducksay_msg_strip_spaces_bool
483 }

```

Redefine keys only intended for version 1 to throw an error:

```

484 \clist_map_inline:nn
485 { align, rel-align }
486 {
487   \keys_define:nn { ducksay }

```

```
488     { #1 .code:n = \msg_error:nn { ducksay } { v1-key-only } }
489 }
```

### 3.3.4 Functions

#### 3.3.4.1 Internal

```
luate_message_alignment_fixed_width_common:
```

```

490 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_common:
491 {
492   \str_case:Vn \l_ducksay_msg_align_tl
493   {
494     { l } { \exp_not:N \l_ducksay_msg_align_l_tl }
495     { c } { \exp_not:N \l_ducksay_msg_align_c_tl }
496     { r } { \exp_not:N \l_ducksay_msg_align_r_tl }
497     { j } { \exp_not:N \l_ducksay_msg_align_j_tl }
498   }
499 }

```

(End definition for \ducksay\_evaluate\_message\_alignment\_fixed\_width\_common:.)

```
uate_message_alignment_fixed_width_tabular:
```

```

500 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_tabular:
501 {
502   \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
503   {
504     \tl_set:Nx \l_ducksay_msg_tabular_column_tl
505     {
506       >
507       {
508         \ducksay_evaluate_message_alignment_fixed_width_common:
509         \exp_not:N \arraybackslash
510       }
511       p { \exp_not:N \l_ducksay_msg_width_dim }
512     }
513   }
514 }

```

(End definition for \ducksay\_evaluate\_message\_alignment\_fixed\_width\_tabular:.)

```
evaluate_message_alignment_fixed_width_vbox:
```

```

515 \cs_new:Npn \ducksay_evaluate_message_alignment_fixed_width_vbox:
516 {
517   \tl_set:Nx \l_ducksay_msg_align_vbox_tl
518     { \ducksay_evaluate_message_alignment_fixed_width_common: }
519 }

```

(End definition for \duksay\_evaluate\_message\_alignment\_fixed\_width\_vbox:.)

```
\ducksay calculate msg width from int:
```

```

520 \cs_new:Npn \ducksay_calculate_msg_width_from_int:
521 {
522   \hbox_set:Nn \l_ducksay_tmpa_box { { \l_ducksay_msg_fount_tl M } }
523   \dim_set:Nn \l_ducksay_msg_width_dim
524     { \l_ducksay_msg_width_int \box_wd:N \l_ducksay_tmpa_box }
525 }

```

(End definition for \ducksay\_calculate\_msg\_width\_from\_int:.)

\ducksay\_msg\_tabular\_begin:

```

526 \cs_new:Npn \ducksay_msg_tabular_begin:
527 {
528   \ducksay_msg_tabular_begin_inner:V \l_ducksay_msg_tabular_column_tl
529 }
530 \cs_new:Npn \ducksay_msg_tabular_begin_inner:n #1
531 {
532   \begin { tabular } { @{} #1 @{} }
533 }
534 \cs_generate_variant:Nn \ducksay_msg_tabular_begin_inner:n { V }

```

(End definition for \ducksay\_msg\_tabular\_begin:.)

\ducksay\_msg\_tabular\_end:

```

535 \cs_new:Npn \ducksay_msg_tabular_end:
536 {
537   \end { tabular }
538 }

```

(End definition for \ducksay\_msg\_tabular\_end:.)

\ducksay\_width\_case\_none\_int\_dim:nnn

```

539 \cs_new:Npn \ducksay_width_case_none_int_dim:nnn #1 #2 #3
540 {
541   \dim_compare:nNnTF { \l_ducksay_msg_width_dim } < { \c_zero_dim }
542   {
543     \int_compare:nNnTF { \l_ducksay_msg_width_int } < { \c_zero_int }
544     { #1 }
545     { #2 }
546   }
547   { #3 }
548 }

```

(End definition for \ducksay\_width\_case\_none\_int\_dim:nnn.)

\ducksay\_digest\_options:n

```

549 \cs_new:Npn \ducksay_digest_options:n #1
550 {
551   \group_begin:
552   \keys_set:nn { ducksay } { #1 }
553   \ducksay_default_or_random_animal:
554   \bool_if:NF \l_ducksay_no_body_bool
555   {
556     \hcoffin_set:Nn \l_ducksay_body_coffin
557     {
558       \frenchspacing
559       \l_ducksay_body_fount_tl
560       \begin{tabular} { @{} l @{} }
561         \l_ducksay_animal_tl
562         \ducksay_make_body_bigger:
563         \relax
564       \end{tabular}
565     }

```

```

( 30 )
\ .-' \
" \
----- ( '-----
)-----)

```

```

566 \bool_if:NT \l_ducksay_msg_eq_body_width_bool
567 {
568   \bool_lazy_and:nnT
569   { \int_compare_p:nNn \l_ducksay_msg_width_int < \c_zero_int }
570   { \dim_compare_p:nNn \l_ducksay_msg_width_dim < \c_zero_dim }
571   {
572     \dim_set:Nn \l_ducksay_msg_width_dim
573     { \coffin_wd:N \l_ducksay_body_coffin }
574   }
575 }
576 }
577 \bool_if:NTF \l_ducksay_eat_arg_box_bool
578 {
579   \ducksay_width_case_none_int_dim:nnn
580   { \ducksay_eat_argument_hbox:w }
581   {
582     \ducksay_calculate_msg_width_from_int:
583     \ducksay_eat_argument_vbox:w
584   }
585   { \ducksay_eat_argument_vbox:w }
586 }
587 {
588   \ducksay_width_case_none_int_dim:nnn
589   {
590     \tl_if_empty:NT \l_ducksay_msg_tabular_column_tl
591     {
592       \str_case:Vn \l_ducksay_msg_align_tl
593       {
594         { l } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l } }
595         { c } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { c } }
596         { r } { \tl_set:Nn \l_ducksay_msg_tabular_column_tl { r } }
597         { j }
598         {
599           \msg_error:nn { ducksay } { justify~unavailable }
600           \tl_set:Nn \l_ducksay_msg_tabular_column_tl { l }
601         }
602       }
603     }
604   }
605   {
606     \ducksay_calculate_msg_width_from_int:
607     \ducksay_evaluate_message_alignment_fixed_width_tabular:
608   }
609   { \ducksay_evaluate_message_alignment_fixed_width_tabular: }
610   \ducksay_eat_argument_tabular:w
611 }
612 }

```

(End definition for \ducksay\_digest\_options:n.)

\ducksay\_set\_bubble\_top\_kern:

```

613 \cs_new:Npn \ducksay_set_bubble_top_kern:
614 {
615   \group_begin:

```

```

( 31 )
\ .-'\
" \
----- ( .) -----
)-----)-----)-----)

```



```

616 \l_ducksay_bubble_fount_tl
617 \exp_args:NNNx
618 \group_end:
619 \dim_set:Nn \l_ducksay_bubble_top_kern_dim
620 { \dim_eval:n { \l_ducksay_bubble_top_kern_tl } }
621 }

```

(End definition for \ducksay\_set\_bubble\_top\_kern:.)

\ducksay\_set\_bubble\_bottom\_kern:

```

622 \cs_new:Npn \ducksay_set_bubble_bottom_kern:
623 {
624 \group_begin:
625 \l_ducksay_bubble_fount_tl
626 \exp_args:NNNx
627 \group_end:
628 \dim_set:Nn \l_ducksay_bubble_bottom_kern_dim
629 { \dim_eval:n { \l_ducksay_bubble_bottom_kern_tl } }
630 }

```

(End definition for \ducksay\_set\_bubble\_bottom\_kern:.)

\ducksay\_make\_body\_bigger:

```

631 \cs_new:Npn \ducksay_make_body_bigger:
632 { \prg_replicate:nn \l_ducksay_body_bigger_int \l }

```

(End definition for \ducksay\_make\_body\_bigger:.)

\ducksay\_baselineskip: This is an overly cautious way to get the current baselineskip. Inside of `tabular` the baselineskip is 0pt, so we fall back to `\normalbaselineskip`, or issue an error and fall back to some arbitrary value not producing an error if that one is also 0pt.

```

633 \cs_new_protected_nopar:Npn \ducksay_baselineskip:
634 {
635 \the\dimexpr
636 \ifdim \baselineskip = \c_zero_dim
637 \ifdim \normalbaselineskip = \c_zero_dim
638 \msg_expandable_error:nn { ducksay } { zero-baselineskip } { 12pt }
639 12pt
640 \else
641 \normalbaselineskip
642 \fi
643 \else
644 \baselineskip
645 \fi
646 \relax
647 }

```

(End definition for \ducksay\_baselineskip:.)

\ducksay\_measure\_msg:

```

648 \cs_new_protected_nopar:Npn \ducksay_measure_msg:
649 {
650 \hbox_set:Nn \l_ducksay_tmpa_box
651 { \l_ducksay_bubble_fount_tl \l_ducksay_bubble_delim_top_tl }
652 \int_set:Nn \l_ducksay_msg_width_int

```

```

( 32 )
\ .-'\
" \
----- ( .) -----
) -----) -----) -----) -----)

```

```

653 {
654   \fp_eval:n
655   {
656     ceil
657     ( \box_wd:N \l_ducksay_msg_box / \box_wd:N \l_ducksay_tmpa_box )
658   }
659 }
660 \group_begin:
661 \l_ducksay_bubble_fount_tl
662 \exp_args:NNNx
663 \group_end:
664 \int_set:Nn \l_ducksay_msg_height_int
665 {
666   \int_max:nn
667   {
668     \fp_eval:n
669     {
670       ceil
671       (
672         (
673           \box_ht:N \l_ducksay_msg_box
674           + \box_dp:N \l_ducksay_msg_box
675         )
676         / ( \arraystretch * \ducksay_baselineskip: )
677       )
678     }
679     + \l_ducksay_vpad_int
680   }
681   { \l_ducksay_msg_height_int }
682 }
683 }

```

(End definition for \ducksay\_measure\_msg:.)

\ducksay\_set\_bubble\_coffins:

```

684 \cs_new_protected_nopar:Npn \ducksay_set_bubble_coffins:
685 {
686   \hcoffin_set:Nn \l_ducksay_bubble_open_coffin
687   {
688     \l_ducksay_bubble_fount_tl
689     \begin{tabular}{@{}l@{}}
690       \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
691       {
692         \l_ducksay_bubble_delim_left_a_tl
693       }
694       {
695         \l_ducksay_bubble_delim_left_b_tl\\
696         \int_step_inline:nnn
697         { 3 } { \l_ducksay_msg_height_int }
698         {
699           \kern-\l_ducksay_bubble_side_kern_tl
700           \l_ducksay_bubble_delim_left_c_tl
701           \\
702         }

```

```

703         \l_ducksay_bubble_delim_left_d_tl
704     }
705     \end{tabular}
706 }
707 \hcoffin_set:Nn \l_ducksay_bubble_close_coffin
708 {
709     \l_ducksay_bubble_fount_tl
710     \begin{tabular}{@{}r@{}}
711         \int_compare:nNnTF { \l_ducksay_msg_height_int } = { \c_one_int }
712         {
713             \l_ducksay_bubble_delim_right_a_tl
714         }
715         {
716             \l_ducksay_bubble_delim_right_b_tl \\\
717             \int_step_inline:nnn
718                 { 3 } { \l_ducksay_msg_height_int }
719                 {
720                     \l_ducksay_bubble_delim_right_c_tl
721                     \kern-\l_ducksay_bubble_side_kern_tl
722                     \\
723                 }
724             \l_ducksay_bubble_delim_right_d_tl
725         }
726     \end{tabular}
727 }
728 \hcoffin_set:Nn \l_ducksay_bubble_top_coffin
729 {
730     \l_ducksay_bubble_fount_tl
731     \int_step_inline:nn
732         { \l_ducksay_msg_width_int + \l_ducksay_hpad_int }
733         { \l_ducksay_bubble_delim_top_tl }
734 }
735 }

```

(End definition for \ducksay\_set\_bubble\_coffins:.)

\ducksay\_join\_bubble\_to\_msg\_coffin:

```

736 \cs_new_protected_nopar:Npn \ducksay_join_bubble_to_msg_coffin:
737 {
738     \dim_set:Nn \l_ducksay_hpad_dim
739     {
740         (
741             \coffin_wd:N \l_ducksay_bubble_top_coffin
742             - \coffin_wd:N \l_ducksay_msg_coffin
743         ) / 2
744     }
745     \coffin_join:NnnNnnnn
746         \l_ducksay_msg_coffin          { l } { vc }
747         \l_ducksay_bubble_open_coffin { r } { vc }
748         { - \l_ducksay_hpad_dim } { \c_zero_dim }
749     \coffin_join:NnnNnnnn
750         \l_ducksay_msg_coffin          { r } { vc }
751         \l_ducksay_bubble_close_coffin { l } { vc }
752         { \l_ducksay_hpad_dim } { \c_zero_dim }

```

```

( 34 )
\ .-' \
" \
----- ( . ) -----
) -----)

```

```

753 \coffin_join:NnnNnnnn
754 \l_ducksay_msg_coffin { hc } { t }
755 \l_ducksay_bubble_top_coffin { hc } { b }
756 { \c_zero_dim } { \l_ducksay_bubble_top_kern_dim }
757 \coffin_join:NnnNnnnn
758 \l_ducksay_msg_coffin { hc } { b }
759 \l_ducksay_bubble_top_coffin { hc } { t }
760 { \c_zero_dim } { \l_ducksay_bubble_bottom_kern_dim }
761 }

```

(End definition for \ducksay\_join\_bubble\_to\_msg\_coffin:.)

\ducksay\_shipout:

```

762 \cs_new_protected:Npn \ducksay_shipout:
763 {
764   \hcoffin_set:Nn \l_ducksay_msg_coffin { \box_use:N \l_ducksay_msg_box }
765   \bool_if:NF \l_ducksay_no_bubble_bool
766   {
767     \ducksay_measure_msg:
768     \ducksay_set_bubble_coffins:
769     \ducksay_set_bubble_top_kern:
770     \ducksay_set_bubble_bottom_kern:
771     \ducksay_join_bubble_to_msg_coffin:
772   }
773   \bool_if:NF \l_ducksay_no_body_bool
774   {
775     \bool_if:NT \l_ducksay_mirrored_body_bool
776     {
777       \coffin_scale:Nnn \l_ducksay_body_coffin
778       { -\c_one_int } { \c_one_int }
779       \str_case:Vn \l_ducksay_body_to_msg_align_body_tl
780       {
781         { l } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { r } }
782         { r } { \tl_set:Nn \l_ducksay_body_to_msg_align_body_tl { l } }
783       }
784     }
785     \bool_if:NTF \l_ducksay_ignored_body_bool
786     { \coffin_attach:NVnNVnnn }
787     { \coffin_join:NVnNVnnn }
788     \l_ducksay_msg_coffin \l_ducksay_body_to_msg_align_msg_tl { b }
789     \l_ducksay_body_coffin \l_ducksay_body_to_msg_align_body_tl { t }
790     { \l_ducksay_body_x_offset_dim } { \l_ducksay_body_y_offset_dim }
791   }
792   \coffin_typeset:NVVnn \l_ducksay_msg_coffin
793   \l_ducksay_output_h_pole_tl \l_ducksay_output_v_pole_tl
794   { \l_ducksay_output_x_offset_dim } { \l_ducksay_output_y_offset_dim }
795   \group_end:
796 }

```

(End definition for \ducksay\_shipout:.)

**3.3.4.1.1 Message Reading Functions** Version 2 has different ways of reading the message argument of \ducksay and \duckthink. They all should allow almost arbitrary content and the height and width are set based on the dimensions.

```

( 35 )
\ .-'\
" \
----- ( .)-----
)-----)

```

\ducksay\_eat\_argument\_tabular:w

```

797 \cs_new:Npn \ducksay_eat_argument_tabular:w
798 {
799   \bool_if:NTF \l_ducksay_eat_arg_tab_verb_bool
800   { \ducksay_eat_argument_tabular_verb:w }
801   { \ducksay_eat_argument_tabular_normal:w }
802 }

```

(End definition for \ducksay\_eat\_argument\_tabular:w.)

\ducksay\_eat\_argument\_tabular\_inner:w

```

803 \cs_new:Npn \ducksay_eat_argument_tabular_inner:w #1
804 {
805   \hbox_set:Nn \l_ducksay_msg_box
806   {
807     \l_ducksay_msg_fount_tl
808     \ducksay_msg_tabular_begin:
809     #1
810     \ducksay_msg_tabular_end:
811   }
812   \ducksay_shipout:
813 }

```

(End definition for \ducksay\_eat\_argument\_tabular\_inner:w.)

\ducksay\_eat\_argument\_tabular\_verb:w

```

814 \NewDocumentCommand \ducksay_eat_argument_tabular_verb:w
815 { >{ \ducksay_process_verb_newline:nnn { ~ } { ~ \par } } +v }
816 {
817   \ducksay_eat_argument_tabular_inner:w
818   {
819     \group_begin:
820     \__ducksay_everyeof:w { \exp_not:N }
821     \exp_after:wN
822     \group_end:
823     \__ducksay_scantokens:w { #1 }
824   }
825 }

```

(End definition for \ducksay\_eat\_argument\_tabular\_verb:w.)

\ducksay\_eat\_argument\_tabular\_normal:w

```

826 \NewDocumentCommand \ducksay_eat_argument_tabular_normal:w { +m }
827 { \ducksay_eat_argument_tabular_inner:w { #1 } }

```

(End definition for \ducksay\_eat\_argument\_tabular\_normal:w.)

\ducksay\_eat\_argument\_hbox:w

```

828 \cs_new_protected_nopar:Npn \ducksay_eat_argument_hbox:w
829 {
830   \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
831   { \@grabbox }
832   { \@grabbox* }
833   {} \l_ducksay_msg_box \l_ducksay_msg_fount_tl \hbox {} \ducksay_shipout:
834 }

```

(End definition for \ducksay\_eat\_argument\_hbox:w.)

\ducksay\_eat\_argument\_vbox:w

```

835 \cs_new_protected_nopar:Npn \ducksay_eat_argument_vbox:w
836 {
837   \ducksay_evaluate_message_alignment_fixed_width_vbox:
838   \bool_if:NTF \l_ducksay_msg_strip_spaces_bool
839     { \@grabbox }
840     { \@grabbox* }
841     {
842       \hsize \l_ducksay_msg_width_dim
843       \linewidth \hsize
844       \l_ducksay_msg_align_vbox_tl
845       \@afterindentfalse
846       \@afterheading
847     }
848     \l_ducksay_msg_box \l_ducksay_msg_fount_tl \vbox {} \ducksay_shipout:
849   }

```

(End definition for \ducksay\_eat\_argument\_vbox:w.)

### 3.3.4.1.2 Generating Variants of External Functions

```

850 \cs_generate_variant:Nn \coffin_join:NnnNnnnn { NVnNVnnn }
851 \cs_generate_variant:Nn \coffin_attach:NnnNnnnn { NVnNVnnn }
852 \cs_generate_variant:Nn \coffin_typeset:Nnnnn { NVVnn }
853 \cs_generate_variant:Nn \str_case:nn { Vn }

```

### 3.3.4.2 Document level

**\ducksay**

```

854 \NewDocumentCommand \ducksay { 0{ } }
855 {
856   \ducksay_digest_options:n { #1 }
857 }

```

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

**\duckthink**

```

858 \NewDocumentCommand \duckthink { 0{ } }
859 {
860   \ducksay_digest_options:n { think, #1 }
861 }

```

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

862 </code.v2>

### 3.4 Definition of the Animals

```

863 <*animals>
864 \ProvidesFile{ducksay.animals.tex}
865 [\ducksay@date\space v\ducksay@version\space ducksay animals]
866 %^~A some of the below are from http://ascii.co.uk/art/
867 \AddAnimal{duck}%>>=
868 { \
869   \
870   >(')
871   )/
872   /(
873   / '----/
874   \ ~-- /
875   ~~~~~~}%<<
876 \AddAnimal{small-duck}%>>=
877 { \
878   \
879   >()_
880   (__)__ _}%<<
881 \AddAnimal{duck-family}%>>=
882 { \
883   \
884   >(')
885   )/
886   /(
887   / '----/ -()_ >()_
888   __\__~--/_/_ _(_)_(_)_ _}%<<
889 \AddAnimal{cow}%>>=
890 { \
891   \ ^__^
892   \ (oo)\_____
893   (__) \          )\/\
894   ||----w |
895   ||     ||}%<<
896 \AddAnimal{head-in}%>>=
897 { \
898   \ ^__^
899   \ (oo)\_____
900   (__) \          )= ( ____| \_____
901   ||----w | \ \ \ \ \_____
902   ||     || ||     || }%<<
903 \AddAnimal{sodomized}%>>=
904 { \
905   \
906   \ ^__^
907   \ (oo)\_____
908   (__) \          ) /
909   ||----w ( (
910   ||     ||>>}%<<
911 \AddAnimal{tux}%>>=
912 { \
913   \ .--.
914   |o_o |
915   |/_/ |

```

```

915         //  \ \
916         (|    | )
917         /'\_  _/'\
918         \___)=(___/}%=<<
919 \AddAnimal{pig}%>>=
920 + \    _//| .-~-.
921     \ _/oo }      }-@
922     ('')_ }      |
923     '---| { }--{ }
924         //_/_/_/+}%=<<
925 \AddAnimal{frog}%>>=
926 { \
927     \ (.)_(.)
928     _ ( _ ) _
929     / \/'-----'\ \
930 --\ ( ( ) ) /--
931 )   /\ \._./ /\ (
932 )_/_/ /\ \ _/ \_}%=<<
933 \AddAnimal{snowman}%>>=
934 { \
935     \[_]_
936     (")
937     >-( : )-<
938     ( _ : _ )}%=<<
939 \AddAnimal[tail-symbol=s]{hedgehog}%>>=
940 { s     .\|//||\||.
941     s  |/\|//|/\|//|
942     /. '|\|//|/\|//|
943     o__._|//|//|//|'}%=<<
944 \AddAnimal{kangaroo}%>>=
945 { \
946     \ _,'
947     <--\__/_ \
948     \_ / _ \
949     \, / \ \
950     //  \ \
951     ,/'      '\_,}%=<<
952 %^~A http://chris.com/ascii/index.php?art=animals/rabbits
953 \AddAnimal[tail-symbol=s,tail-count=3]{rabbit}%>>=
954 { s
955     s  / \'\
956     s | \ \ ' \ /' / \
957     \_/' \ \ -"-/' / \ \
958         |         | \ \ |
959         (d      b) \_/_/
960         /         \
961         ,".|.'.\_/.'.|.|"
962         /  /\' _|_ ' /\ \
963         | /  ' ' " ' _ \ |
964         | |         | |
965         | \ \ \ / / |
966         \ \ \ / / /
967         " " \ : /' " '
968         ' " " " " '}%=<<

```







```

1077 ;' \ ; : . , ;' _ _ / ;
1078 : "-. "-: ; , :/." .' :
1079 \ \ : : ;/_ _ :
1080 \ \ .-'\ \ /t=" " ":-+. :
1081 ' . -" '1 _ _ / /' : ; ; \ ;
1082 \ \ .- " .- "-.-" .' .' j \ / ;/
1083 \ / .- " / . .' .' ;- ' ;
1084 :-"-.-' ./.-' / '._ _.'
1085 \ 't . _ /
1086 "-.t-.-:'}%=<<
1087 \AddAnimal[tail-count=3]{yoda-head}%>>=
1088 { \
1089 \ \
1090 \ \
1091 .-.' ; '._
1092 / : _ _ \ ; / _ _ ; \
1093 ,'- " _ _ .: _ _ ; "-.": _ _ ; _ _ " _ ' ,
1094 : ' 't" _ _ . >@.' ; _ , @' ' _ _ "j.' ' ;
1095 ' : _ _ J ' _ _ 'L _ _ ' _ _ ' L _ _ .- ;'
1096 "-. _ _ ; .- " "-. : _ _ -"
1097 L ' / . _ _ _ _ _ . \ ' J
1098 "-. " _ _ " _ _ -"
1099 _ _ .l"-: _ JL _ ; -" ; _ _
1100 .-j/' : ; ; " " " / .' \ "-.
1101 .' / : ' : : / ." .' ; ' .
1102 .- " / ; ' ." : . " " : "-.
1103 .+"- . : : " . " . " " ; - _ _ \}%=<<
1104 %~^A from https://www.ascii-code.com/ascii-art/movies/star-wars.php
1105 \AddAnimal{small-yoda}%>>=
1106 { \
1107 \
1108 _ _ .- _
1109 ' _ _ "7'
1110 /' .-c
1111 | /T
1112 _ ) _ /LI}%=<<
1113 \AddAnimal{r2d2}%>>=
1114 { \
1115 \ , _ _ _ _ .
1116 ,'_/_|_|_\'
1117 /<<:8[0]:>\
1118 _ | _ _ _ _ _ _ _ | _
1119 | | ==-== | |
1120 | | --== | |
1121 \ | ::::|()|| /
1122 | | ....|()|| |
1123 | | _ _ _ _ _ _ | |
1124 | | \ _ _ _ _ _ / | |
1125 / \ \ / \ / \ \
1126 ' _ _ , ' _ _ , ' _ _ }%=<<
1127 \AddAnimal{vader}%>>=
1128 { \ \ .-.' ~~~~~ ' _ _
1129 \ / \ || \
1130 / \ || \

```

```

1131      |      ||      |
1132      |-----||-----|
1133      | / ----- \ / ----- \ |
1134      / ( ) ( ) \
1135      / \ ----- () ----- / \
1136      / \      /||\
1137      / \      /||||\
1138      / \      /|||||\
1139      /- \0=====0/ \-
1140      '---..._|'-'-'-'|_---...-'
1141      |      '      |}%=<<
1142      \AddAnimal[tail-symbol=|,tail-count=1]{crusader}%>>=
1143      { |
1144      \[T]/}
1145      \csname bool_if:cT\endcsname {l_ducksay_version_one_bool}
1146      {\AnimalOptions{crusader}{tail-1=|,rel-align=c}}
1147      \csname bool_if:cT\endcsname {l_ducksay_version_two_bool}
1148      {\AnimalOptions{crusader}{tail-1=|,body-align=c}}}%=<<
1149      %^A http://ascii.co.uk/art/knights
1150      \AddAnimal[tail-count=3]{knight}%>>=
1151      {
1152      \      ,-"-".
1153      \      | === |
1154      )      | (
1155      .==\' " "/\'==.
1156      .\' \ (\' :) /\'
1157      _/_ |_-\' : \'_|_ \_
1158      <--->\' \ : / \'<--->
1159      / / >===== < / /
1160      _/_ .\' / ,--:-- \/=,\'
1161      /_/_ |_/v^v^v^v\_) \
1162      \(\) |V^V^V^V^V|\_/
1163      (\ \ \'---|---\'/
1164      \ \ \-_|_-,-/
1165      \ \ |__|_|_|
1166      \ \ <---X--->
1167      \ \ \..|..|/
1168      \ \ \ | /
1169      \ \ /V|V\
1170      \ / | \
1171      ,--\' --\'}%=<<
1172      %^A https://www.asciiart.eu/mythology/ghosts
1173      \AddAnimal{ghost}%>>=
1174      {
1175      \ .-.
1176      (o o)
1177      | 0 \
1178      \ \ \
1179      \'~~~\'}%=<<
1180      %^Ahttps://asciiart.website/index.php?art=creatures/fairies
1181      \AddAnimal{fairy}%>>=
1182      {
1183      \ .o00b
1184      .. .o0 0

```

```

1185      ';; d      0
1186      ;;;d      ..o0
1187      *      :0;;;'0oo0
1188      ~"\. dp'(0.o.
1189      \op      'o0b
1190              obU
1191              dop
1192              dop
1193              P0
1194              0 'b
1195              1 P.
1196              / ;
1197              '}%=<<
1198      \AddAnimal[tail-symbol=s]{only-tail}%>=
1199      { s
1200          s}%=<<
1201      \AddAnimal[tail-symbol=s,tail-count=3]{only-tail3}%>=
1202      { s
1203          s
1204          s}%=<<
1205      %^A head taken from https://www.asciiart.eu/animals/reptiles/snakes
1206      \AddAnimal[tail-symbol=s,tail-count=3]{snake}
1207      { s
1208          s      /\^/\
1209          s _|_| 0 |
1210          /'      \_/ \
1211      \ / |-----/ \
1212      \_/ \----- \
1213          ' | |
1214          / / / _ _ _ | |
1215          / / / _ _ " _ , " |
1216          | " _ _ / " _ _ _ , "
1217          " _ _ _ _ " " _ _ _ _ }
1218      %^A http://www.ascii-art.de/ascii/c/cat.txt
1219      \AddAnimal{cat}
1220      + \
1221          \
1222          \ ' . | \ . . . . . ' _ _ _ ' _ . . ' _ .
1223          / ' ' ' _ _ _ _ _ ' _ . . '
1224          ) /' _ / \ ' _ _ /
1225          ' _ " ' " \ _ , _ . ; _ . \ _ ,
1226          _ . - ' _ / { _ . ' ; /
1227          { _ . ' ' _ { _ / +
1228      %^A https://www.asciiart.eu/animals/cats
1229      \AddAnimal{sleepy-cat}
1230      { \
1231          \ | \
1232          / , ' _ , ' _ _ _ _ _ ' _ . ' _ . ,
1233          | , 4 - ) ) - , _ , \ ( ' _ . - '
1234          , _ _ _ , ( _ / _ _ ' _ _ \ _ )
1235      \AddAnimal{schroedinger-dead}
1236      { \
1237          \ _ . - - " " - - _
1238          |

```



Who's gonna use it anyway?

0

o

>( ' )

) /

/(

/ '-----/

\ ~=- /

~ ^ ~ ^ ~ ^ ~ ^ ~ ^ ~ ^ ~ ^

Hosted at  
[https://github.com/Skillmon/ltx\\_ducksay](https://github.com/Skillmon/ltx_ducksay)  
it is.

--.-.-  
'-.\_"7'  
/'.-c  
| /T  
\_)\_/LI