Eggs

Linguistic examples with minimalist syntax

November 2025

https://github.com/retroflexivity/typst-eggs

Eggs (/ɛgz/ or /ɪgz/) is a Typst package for typesetting linguistic examples. Its aim is to provide a Typst analogue to LaTeX `gb4e`, `expex`, `linguex`, etc. Additionally, it ships with `leipzig`-style gloss abbreviations.

This is a documentation on Eggs. It begins with a review of features, then provides an extensive list of functions and their arguments.

1. Initializing

To use Eggs, first import it, and then set its config via a global show rule.[1]

```
#import "@preview/eggs:0.1.0"
#show: eggs
```

This same function is used to configure Eggs' settings. See below.

2. Examples

The primary fuction of Eggs is `example`. It acceps any `content` and typesets it as a single top-level linguistic example.

```
The sentence below demonstrates the reading known as specificational.
#example[
  The primary function of Eggs is `example`.
]
The semantics of the DP to the left of _is_ has been an object of much debate.
```

The sentence below demonstrates the reading known as specificational.

(1) The primary function of Eggs is `example`.

The semantics of the DP to the left of *is* has been an object of much debate.

Examples are automatically numbered continuously. To override automatic numbering, pass the `number` argument. Examples with a custom number do not increment the counter, so numbering continues where it was left off. Passing things that are not numbers is not supported yet.

```
The idea of Predicate Inversion stems from the seeming parallelism of the following
sentences.
#example[
  `example` is the primary function of Eggs.
]
#example(number: 1)[
  The primary function of Eggs is `example`.
]
```

The idea of Predicate Inversion stems from the seeming parallelism of the following sentences.

(2) `example` is the primary function of Eggs.

(1) The primary function of Eggs is `example`.

The counter can also be set explicitly with `#counter("example").update(n)`. This way, following examples will be numbered starting with $n + 1$. See counter for more info.

[1] This package follows Typst's tradition of not abbreviating function names too much. Naturally, to achieve more effortless (or TeX-like) experience, you can set some aliases on import, e.g.

```
#import "@preview/eggs:0.1.0": example as ex, subexample as subex, judge as j, ex-label as el, ex-ref as er
```

3. Subexamples

Numbered lists inside examples (lines that begin with +) are automatically typeset as subexamples.

```
Compare the following two sentences, of which only the former exhibits the Definiteness
Effect.
#example[
  + There is a/\*the subexample.
  + Here is a/the subexample.
]
```

Compare the following two sentences, of which only the former exhibits the Definiteness Effect.

(3) a. There is a/*the subexample.
b. Here is a/the subexample.

In case you prefer it manual, the function `subexample` is defined. It is intended to only be used inside `example`. Automatic conversion of numbered lists can be toggled off by setting `auto-subexamples: false` in the config (see Section 8). To suspend it for a single example, pass `auto-subexamples: false` to the subexample directly.

```
Compare the following two sentences, of which only the former exhibits the Definiteness
Effect.
#example(auto-subexamples: false)[
  #subexample[There is a/\*the subexample.]
  #subexample[Here is a/the subexample.]
]
```

Compare the following two sentences, of which only the former exhibits the Definiteness Effect.

(3) a. There is a/*the subexample.
b. Here is a/the subexample.

4. Glosses

Bullet lists in examples (lines that begin with `-`) are automatically treated as gloss lines.

For words to split, you need to ensure that there is a `space` element between. The easiest way to do it is to separate words with **more than one** space (e.g. by pressing TAB in most editors). There are exceptions, so use ~ (non-breaking space) for spaces you don't want to be treated as separators.

Translations and preambles are written as lines below and above glosses, respectively.

```
The following example presents the Russian _eto_-construction.
#example[
  Russian
  - Jajca      eto   vkusno.
  - eggs  this           tasty
  'Eggs are tasty.'
]
```

The following example presents the Russian *eto*-construction.

(4) Russian
Jajca eto vkusno.
eggs this tasty
'Eggs are tasty.'

Glosses can by typeset manually with `gloss`. It accepts either a content that it splits automatically or a list. Automatic bullet list conversion can be toggled off by setting `auto-glosses: false` in the config (see Section 8). To suspend it for a single example, pass `auto-glosses: false` to the (sub)example directly.

```
The following example presents the Russian _eto_-construction.
#example(auto-glosses: false)[
  Russian
  #gloss(
    [Jajca      eto   vkusno.],
    ([eggs], [this], [tasty]),
  )
  'Eggs are tasty.'
]
```

The following example presents the Russian *eto*-construction.

(4) Russian
Jajca eto vkusno.
eggs this tasty
'Eggs are tasty.'

5. Judges

Certain strings at the beginning of a line or a gloss word (see Section 4) are automatically transformed into left judges.[2] Left judges are negatively padded strings that take up no space.

By default, such strings are `*`, `#`, `?`, `OK`, and all combinations of these. All except the asterisk are also superscripted. Spaces after a judge are omitted for readability.

```
The following pair shows the information-structural rigidity of specificational sentences.
#example[
  + OK SMITH is the killer.
  +  \*THE KILLER is Smith.
]
```

The following pair shows the information-structural rigidity of specificational sentences.

(5) a. ^{OK}SMITH is the killer.
b. *THE KILLER is Smith.

This can be modified by tweaking `auto-judges` in the config (see Section 8) or by passing the `auto-judges` argument to an example directly. `auto-judges` takes a dictionary where keys are the strings and values indicate whether to additionally superscript them. For instance, if you would like to make hashes and question marks non-superscripted, try `auto-judges: ("\*": false, "\#": false, "?": false, "OK": true)`. Use `auto-judges: ()` to disable the automatic conversion completely.

Again, a function `judge` is provided to typeset judges manually. Following spaces are **not** omitted.

```
The following pair shows the information-structural rigidity of specificational sentences.
#example(auto-judges: ())[
  + #judge(super[OK])SMITH is the killer.
  +  #judge[\*]THE KILLER is Smith.
]
```

The following pair shows the information-structural rigidity of specificational sentences.

(5) a. ^{OK}SMITH is the killer.
b. *THE KILLER is Smith.

N.B.: Avoid using `judge` with strings that are already in `auto-judges`, as this can lead to superscript doubling.

6. Abbreviations

Eggs provides the whole set of Leipzig Standard Abbreviations as commands, in the style of TeX's `leipzig`. They are imported from a subpackage `abbreviations`.

All abbreviations are in lowercase. All abbreviations' names are as they appear, except for:

- **1, 2, 3** are `p1`, `p2`, `p3`
- **N** (as the non- prefix) is `non`

[2] In fact, due to the way show rules work, this happens at the beginning of any elements inside examples, including inline ones. This might overgenerate left judges in certain narrow cases like in `this one _?here_`. Disable auto judges if it is a problem.

```
#import abbreviations: ins, pst, n

Note that the famous temperature examples forbid Instrumental in Russian.
#example[
  - \*Temperatur-oj  byl-o        10~gradusov.
  - weather-#ins             be-#pst\-#n  10~degrees
]
```

Note that the famous temperature examples forbid Instrumental in Russian.

(6) *Temperatur-oj byl-o 10 gradusov.
weather-INS be-PST-N 10 degrees

Custom abbreviations can be created by defining a new `abbreviation`.

```
#let eto = abbreviation("eto", "an extremely polysemous expression")

Due to the polysemy _eto_ exhibits, I will gloss it simply as #eto.
#example[
  - Eto   čto   za   primer?
  - #eto  what  for  example
  'What's this example?'
]
```

Due to the polysemy *eto* exhibits, I will gloss it simply as ETO.

(7) Eto čto za primer?
ETO what for example
'What's this example?'

The list of currently used abbreviations is stored as a dictionary of the form `abbr: description` inside the `used-abbreviations` state. The final list of abbreviations can be accessed as `#context state("used-abbreviations").final()`. Additionally, there is a convenient function to print it as a term list.

```
The list of glossing abbreviations used in this paper:
#print-abbreviations()
```

The list of glossing abbreviations used in this paper:
INS instrumental
PST past
N neuter
ETO an extremely polysemous expression

7. Labels and refs

There are two ways of labelling an example (or a subexample). The first is passing the `label` argument to `example` (or `subexample`). The second is placing an `ex-ref` anywhere in the body of the example (or a subexample), preferably the beginning or the end. Typst's built-in `ref` does not work for now.

If a subexample doesn't have a label but its parent does, an automatic codly-style label of the form `<top-label:subex-letter>` is added automatically.

```
The curious minimal pair in terms of scope in (@pair) is taken Arregi et al. (2021). Only
(@pred) has a narrow scope reading of the indefinite.
#example[
  #ex-label(<pair>)
  + OK Kim is not one of the judges. #ex-label(<pred>)
  + \#One of the judges is not Kim.
]
(@pair:b) is pragmatically odd, but becomes better if Kim is replaced with something that
can exist in multiple places at once (@ok). The argument loses in convincibility, though.
#example(label: <ok>)[
  OK One of the judges is not "OK".
]
```

The curious minimal pair in terms of scope in (8) is taken Arregi et al. (2021). Only (8a) has a narrow scope reading of the indefinite.

(8) a. $^{\text{OK}}$Kim is not one of the judges.
b. $^{\#}$One of the judges is not Kim.

(8b) is pragmatically odd, but becomes better if Kim is replaced with something that can exist in multiple places at once (9). The argument loses in convincibility, though.

(9) $^{\text{OK}}$One of the judges is not "OK".

Until Typst's reference customization is more powerful (read: existent), Eggs provides a custom function `ex-ref`. It

- accepts from one to two arguments;
- accepts example labels and integers: integers are used for relative references;
- collapses the number of the second argument if it refers to a subexample;
- additionally accepts `left` and `right`, which it prints to the left and to the right of the number;
- encloses this all in parentheses.

```
Recall the discussion concerning examples #ex-ref(<pair>, <ok>). Despite the Russian data,
we should not ignore the minimal pair in #ex-ref(<pred>, <pair:b>). There is much more to
the story #ex-ref(left: "e.g. ", 1, right: " etc.").

#example[
  + Only "?" is one of the judges.
  + One of the judges is only "?".
]
```

Recall the discussion concerning examples (8-9). Despite the Russian data, we should not ignore the minimal pair in (8a-b). There is much more to the story (e.g. 10 etc.).

(10) a. Only "?" is one of the judges.
b. One of the judges is only "?".

Thus, modulo parentheses, `#ex-ref(1)` is `\nextx`, `#ex-ref(0)` is `\lastx`, etc.

8. Customization

Eggs offers some layout and styling customization and several behaviour toggles. The complete list is given in Section 9. Customization is done via setting the parameters of `eggs` in a global show rule.

```
#show: eggs.with(
  indent: 2.5em,
  body-indent: 2.5em,
  sub-indent: -0.3em,
  sub-body-indent: 1.7em
)

The examples in this part all imitate Higgins' (1973) original layout.
#example[
  + What John also is is enviable.
  + What John also is is also enviable.
]

#example[
  What I am pointing at is a kangaroo.
]
```

The examples in this part all imitate Higgins' (1973) original layout.

(11) a. What John also is is enviable.
b. What John also is is also enviable.

(12) What I am pointing at is a kangaroo.

To change Eggs' config temporarely, you can also simply pass the content to `eggs`. Note, however, that all parameters not passed are set to default. If you change your config frequently, you may want to store your defaults in a separate variable.

```
#eggs(indent: 2.5em, body-indent: 2.5em, sub-indent: -0.3em, sub-body-indent: 1.7em)[
  The following examples imitate Higgins' (1973) original layout.
  #example[
    + What John also is is enviable.
    + What John also is is also enviable.
  ]

  #example[
    What I am pointing at is a kangaroo.
  ]
] // this ends the scope of eggs
Compare #ex-ref(0) with itself but in the default layout.
#example(number: 12)[
  What I am pointing at is a kangaroo.
]
```

The following examples imitate Higgins' (1973) original layout.

(11) a. What John also is is enviable.
b. What John also is is also enviable.

(12) What I am pointing at is a kangaroo.

Compare (12) with itself but in the default layout.

(12) What I am pointing at is a kangaroo.

9. Complete function documentation

9.1. eggs

Sets the default config with optional overrides. Primarily intended for use in a global show rule: `#show: eggs()`

Parameters

```
eggs(
  it: content,
  auto-subexamples: bool,
  auto-glosses: bool,
  auto-labels: bool,
  auto-judges: array,
  indent: length,
  body-indent: length,
  sub-indent: length,
  sub-body-indent: length,
  spacing: length,
  sub-spacing: length,
  num-pattern: str function,
  sub-num-pattern: str function,
  label-supplement: str none,
  sub-label-supplement: str none,
  breakable: bool,
  sub-breakable: bool,
  gloss-word-spacing: length,
  gloss-line-spacing: length,
  gloss-before-spacing: length,
  gloss-after-spacing: length,
  gloss-styles: array
) -> content
```

it `content`

The content.

auto-subexamples `bool`

Whether to treat numbered lists in examples as subexamples.

Default: `true`

auto-glosses `bool`

Whether to treat bullet lists in examples as glosses.

Default: `true`

auto-labels `bool`

Whether to insert subexample labels of the form ex-label:a.

Default: `true`

auto-judges `array`

A dictionary of characters to convert into judges (keys) and whether to superscript them (values).

Default: `(`
```
    "\*": false,
    "\#": true,
    "?": true,
    "OK": true,
  )
```

indent `length`

Distance between the left margin and the left edge of the example number.

Default: `0em`

body-indent `length`

Distance between the left edge of the example marker and the left edge of the example body.

Default: `2.5em`

sub-indent `length`

Distance between the left edge of the top-level example body and the left edge of the subexample number. Can be negative.

Default: `0em`

sub-body-indent `length`

Distance between the left edge of the subexample marker and the subexample body.

Default: `1.5em`

spacing `length`

Vertical spacing around example. Currently, there is no way to modify spacing between two subexamples specifically. Defaults to `par.spacing`.

Default: `auto`

sub-spacing `length`

Vertical spacing around subexamples. Defaults to `par.leading`.

Default: `auto`

num-pattern `str` or `function`

Example number format. A numbering pattern.

Default: `"(1)"`

sub-num-pattern `str` or `function`

Subexample number format. A numbering pattern.

Default: `"a."`

label-supplement `str` or `none`

The example figure supplement used in references.

Default: `none`

sub-label-supplement `str` or `none`

The subexample figure supplement used in references.

Default: `none`

breakable `bool`

Whether the example figure is breakable.

Default: `false`

sub-breakable `bool`

Whether the subexample figure is breakable.

Default: `false`

gloss-word-spacing `length`

Horizontal spacing between words in glosses.

Default: `1em`

gloss-line-spacing `length`

Vertical spacing between lines in glosses. Defaults to `par.leading`.

Default: `auto`

gloss-before-spacing `length`

Vertical spacing above glosses (i.e. after the preamble). Defaults to `par.leading`.

Default: `auto`

gloss-after-spacing `length`

Vertical spacing below glosses (i.e. before the translation). Defaults to paragraph leading.

Default: `auto`

gloss-styles `array`

List of functions to be applied to each line of glosses. Can be of any length. `gloss-styles[0]` is applied to the first line, `gloss-styles[1]` — to the second, etc.

Default: `()`

9.2. example

Typesets a top-level example as a figure of kind “example”.

Parameters

```
example(
  content: content,
  label: label none,
  number: int none,
  auto-subexamples: bool,
  auto-glosses: bool,
  auto-judges: array
) -> content
```

content `content`

The body of the example.

label `label` or `none`

The label to use for reference.

Default: `none`

number `int` or `none`

Overrides automatic numbering of the example. If not none, the counter does not increment.

Default: `none`

auto-subexamples `bool`

Override config preset for automatic numbered linst conversion.

Default: `auto`

auto-glosses `bool`

Override config preset for automatic bullet list conversion.

Default: `auto`

auto-judges `array`

Override config preset for automatic judge conversion.

Default: `auto`

9.3. subexample

Explicitly typesets a subexample as a figure of type “subexample”. Only intended for use inside an example.

Parameters

```
subexample(
  content: content,
  label: label none,
  number: int none,
  auto-glosses: bool
) -> content
```

content `content`

The body of the subexample.

label `label` or `none`

The label to use for reference.

Default: `none`

number `int` or `none`

Overrides automatic numbering of the subexample. If not none, the counter does not increment.

Default: `none`

auto-glosses `bool`

Override config preset for automatic bullet list conversion.

Default: `auto`

9.4. judge

Typesets the content, then adds negative space of the length of this content. Used to add a judge to the left of a text.

Parameters

```
judge(j: content) -> content
```

j `content`

The content to typeset.

9.5. gloss

Typesets a block of interlinear glosses.

Parameters

```
gloss(..args: content array)
```

..args `content` or `array`

Any number of rows of equal length. Rows can be either contents where elements are separated by more than one space or lists.

9.6. abbreviation

Prints the symbol in smallcaps and adds it to the list of used abbreviations.

Parameters

```
abbreviation(
  symbol: str,
  description: str
) -> content
```

symbol `str`

The abbreviation.

description `str`

The abbreviation description for the list of used abbreviations.

9.7. print-abbreviations

Prints the list of abbreviations used in the document as a term list.

Parameters

```
print-abbreviations() -> content
```

9.8. ex-label

Used inside an example to give it a label. Effectively an alias of metadata.

Parameters

```
ex-label(l: label) -> content
```

l `label`

The label.

9.9. ex-ref

Typesets an example reference in parentheses. Accepts labels and integers for relative references. Automatically handles plural references such as (1-3) and (1a-c)

Parameters

```
ex-ref(
  ..args: label int,
  left: content,
  right: content
) -> content
```

..args `label` or `int`

One to two arguments. Can be either labels or integers. Integers are used for relative reference: 0 means the last example, 1 means the next, etc.

left `content`

Text on the left, e.g. “e.g. “

Default: none

right `content`

Text on the right, e.g. “ etc.”

Default: `none`