anki-typst

Create anki cards from typst.

v0.2.0 2024-06-08

Contents

1. Examples

Via theorem environment:

```
#import anki.theorems: item
// Don't forget this! v
#show: anki.setup.with(enable_theorems:
true)

// create item kinds
#let example = item("Example", initial_tags:
("example",))
#let theorem = item("Theorem", proof_name:
"\"Proof\"")

// create item
#example("Pythagoras")[
 $ a^2 + b^2 = c^2 $
]

// use secondary numbering
#example("triangle", secondary: auto)[
  #sym.triangle.tr.filled
]
#example("another triangle", secondary:
auto)[
  #sym.triangle.t.stroked
]

// and a theorem, with a custom number
#theorem("Triangular numbers", number: "42")
[
  The triangular numbers are given by:
  $ T_n = sum_(k=1)^n k = (n(n+1))/2 $
][
  Induction over n.
]
```

Example 1.0.1 (Pythagoras).

$$a^2 + b^2 = c^2$$

Example 1.0.1a (triangle). ◥

Example 1.0.1b (another triangle). ▵

Theorem 42 (Triangular numbers). The triangular numbers are given by:

$$T_n = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

"Proof". Induction over n. □

Via raw function:

```
#import anki: anki_export

#anki_export(
  id: "id 29579",
  tags: ("Perfect", ),
  deck: "beauty",
  model: "simple",
  question: "Are you beautiful?",
  answer: "Yes!",
)
```

2. Function reference

2.1. lib

setup

Setup the document

This is crucial for displaying everything correctly!

Example:

```
show: anki.setup.with(enable_theorems: true)
```

Parameters

```
setup(
  doc: content,
  export: bool auto,
  enable_theorems: bool,
  title: str none,
  prefix_deck_names_with_numbers: bool
) -> content
```

doc `content`

The document to wrap.

export `bool` or `auto`

Whether to enable export mode. If `export` is `auto`, anki-typst will try to read from `sys.inputs`.

Default: `auto`

enable_theorems `bool`

Whether to enable theorem support (via `ctheorems`)

Default: `false`

title `str` or `none`

The top-level deck name of all cards.

Default: `none`

prefix_deck_names_with_numbers `bool`

Whether to prefix all deck names with the corresponding heading number.

Default: `false`

2.2. raw

anki_export

Create an anki card.

Even though the default values of `id`, `deck` and `model` are `none`, they are required!
This does not create the card on its own, you have to use the command line interface!

Example

```
#import anki: anki_export

#anki_export(
  id: "id 29579",
  tags: ("Perfect", ),
  deck: "beauty",
  model: "simple",
  question: "Are you beautiful?",
  answer: "Yes!",
)
```

Parameters

```
anki_export(
  id: str,
  tags: array,
  deck: str,
  model: str,
  number: int str none,
  ..fields: arguments
)
```

id `str`

The id of the card. Used to update the card later on.

Default: `none`

tags `array`

Tags to add to the card.

Default: `()`

deck `str`

Name of the card deck. Anki nests decks with `::`, so you can try `Deck::Subdeck`.

Default: `none`

model `str`

Name of the card model.

Default: `none`

number `int` or `str` or `none`

The number of the card. Not really special but passed differently to the command line interface.

Default: `none`

..fields `arguments`

Additional fields for the anki card.

2.3. theorems

anki_thm

Create an anki card.

Example

```
#import anki.theorems: anki_thm

#anki_thm(
  "id 29579",
  tags: ("Perfect", ),
  deck: "beauty",
  question: "Are you beautiful?",
  answer: "Yes!",
)
```

Parameters

```
anki_thm(
  id: str,
  tags: array,
  deck: none str,
  model: none str,
  numbering: str function none,
  number: auto function array,
  secondary: none auto true function array,
  secondary_numbering: str function none,
  ..fields: arguments
)
```

id `str`

The id of the card. Used to update the card later on.

tags `array`

Tags to add to the card.

Default: `()`

deck `none` or `str`

Name of the deck. Anki nests decks with `::`, so you can try `Deck::Subdeck`. If `deck` is `none` it will be read from state.

Default: `none`

model `none` or `str`

Name of the model. If `model` is `none` it will be read from state.

Default: `none`

numbering `str` or `function` or `none`

The pattern for the primary number.

Default: `"1.1"`

number `auto` or `function` or `array`

The primary number of the card.

Default: `auto`

secondary `none` or `auto` or `true` or `function` or `array`

The secondary number of the card.

Default: `none`

secondary_numbering `str` or `function` or `none`

The pattern for the secondary number.

Default: `"a"`

..fields `arguments`

Additional fields for the anki card.

deck

Set the current deck name.

Parameters

`deck(name: str)`

name `str`

New name.

inner

Inner function to create anki items.

Parameters

```
inner(
  front: content str,
  content: content,
  tags: array,
  deck: none str,
  model: none str,
  clear_tags: bool,
  number: auto function array,
  secondary: none auto true function array,
  ..maybe_proof: none content
)
```

front `content` or `str`

Front content for the card.

content `content`

Main content for the card.

tags `array`

Tags to add to the card.

Default: `()`

deck `none` or `str`

Name of the deck. Anki nests decks with `::`, so you can try `Deck::Subdeck`. If `deck` is `none` it will be read from state.

Default: `none`

model `none` or `str`

Name of the model. If `model` is `none` it will be read from state.

Default: `none`

clear_tags `bool`

Remove `initial_tags` and use only `tags`.

Default: `false`

number `auto` or `function` or `array`

The primary number of the card.

Default: `auto`

secondary `none` or `auto` or `true` or `function` or `array`

The secondary number of the card.

Default: `none`

..maybe_proof `none` or `content`

The proof of the card if specified.

item

Main function to create anki items.

This function returns a function which represents an item kind. You can call the returned function multiple times to create multiple items.

Examples

```
#import anki.theorems: item
// Don't forget this!
#show: anki.setup.with(enable_theorems:
true)

// create item kinds
#let example = item("Example", initial_tags:
("example",))
#let theorem = item("Theorem", proof_name:
"\"Proof\"")

// create item
#example("Pythagoras")[
 $ a^2 + b^2 = c^2 $
]

// use secondary numbering
#example("triangle", secondary: auto)[
  #sym.triangle.tr.filled
]
#example("another triangle", secondary:
auto)[
  #sym.triangle.t.stroked
]

// and a theorem, with a custom number
#theorem("Triangular numbers", number: "42")
[
  The triangular numbers are given by:
  $ T_n = sum_(k=1)^n k = (n(n+1))/2 $
][
  Induction over n.
]
```

Example 2.3.1 (Pythagoras).

$$a^2 + b^2 = c^2$$

Example 2.3.1a (triangle). ◥

Example 2.3.1b (another triangle). △

Theorem 42 (Triangular numbers). The triangular numbers are given by:

$$T_n = \sum_{k=1}^{n} k = \frac{n(n+1)}{2}$$

"Proof". Induction over n. □

Parameters

```
item(
  name: str,
  initial_tags: array,
  base_level: none int,
  inset: relative dictionary,
  separator: content,
  numbering: str function none,
  secondary_numbering: str function,
  create_item_label: bool,
  item_label_prefix: str,
  item_args: dict,
  id: function,
  proof_name: str,
  proof_args: dict
)
```

name `str`

Name of the item kind.

initial_tags `array`

Tags to add to each item of this kind. To remove these tags pass `clear_tags: true` to the inner function.

Default: `()`

base_level `none` or `int`

The number of levels from headings to take for item numbering. `none` means "use all heading levels".

Default: `2`

inset `relative` or `dictionary`

How much to pad the block's content.

Default: `0em`

separator `content`

Separator between name and body.

Default: `[. #h(0.1em)]`

numbering `str` or `function` or `none`

The numbering pattern for the primary number.

Default: `"1.1"`

secondary_numbering `str` or `function`

The numbering pattern for the secondary number.

Default: `"a"`

create_item_label `bool`

Whether to create a new label for each item of this kind. The label will be `item_level_prefix + name`

Default: `true`

item_label_prefix `str`

Prefix for item labels.

Default: `""`

item_args `dict`

Arguments which will be passed to `ctheorems` for each item of this kind.

Default: `(:)`

id `function`

Function to create the id of the card. The id must be unique as it is used to update cards later on. The function will be called with `plain_front, deck, model, number, secondary, ..fields`.

Default: `fields => fields.at("plain_front")`

proof_name `str`

How the proof (or in general second argument) should be called.

Default: `"Proof"`

proof_args `dict`

Arguments which will be passed to `ctheorems` for each proof.

Default: `(:)`

model

Set the current model name.

Parameters

```
model(name: str)
```

name `str`

New name.

set_thmcounter

Set the counter for the theorems.

Example:

```
anki.theorems.set_thmcounter(items: (3, 5, 7))
// After this, the next item will have the number 1.0.2
anki.theorems.set_thmcounter(items: (1, 0, 1))
```

Parameters

```
set_thmcounter(
  heading: int array function,
  items: array
)
```

heading `int` or `array` or `function`

The new heading counter value.

Default: `none`

items `array`

The new item counter value. Each element in `items` corresponds to a level. The `base_level` argument of `item` corresponds to the number of `items`.

Default: `none`

setup

Setup the document

This is crucial for displaying everything correctly!

Example:

```
show: anki.theorems.setup
```

Parameters

```
setup(doc: content) -> content
```

doc `content`

The document to wrap.