Custom-numbered-blocks Example

1 Custom blocks and crossreferencing

With this filter, you can define custom div classes (environments) that come with numbering, such as theorems, examples, exercises. The filter currently supports output formats pdf and html.

It is possible to define shared counters for multiple classes of blocks, similar to LaTeX's amsthm, by defining `groups`.

New feature 1.1: New options for block appearance

The default appearance for custom divs is **foldbox**: a collapsible similar to quarto's callouts, with a collapse button at the bottom that makes it easier collapse long boxes, and box open to the right. It comes with the variant `foldbox.simple`, with closed box and no additional close button.
Two new ways a custom block can be rendered are available now:

- **quartothmlike** mimics the appearance of theorem blocks in quarto, however they look alike in both `html` and `pdf` formats
- **textbox** a colored text box

Feature 1.2: Numbering: cross reference overhaul with slightly changed behaviour

Numbering is by default within document and without section prefix for single documents, or within chapter and with chapter prefix for books. Grouped classes share the same counter, and the same default style.
The default behaviour can be changed by setting the item `chapters` in yaml-key `crossreference` to `true` or `false`, to comply with quarto's (no longer so) new crossreferencing options.
In the present single document, yaml contains

```
crossref:
```

```
chapters: true
```

Numbered custom blocks can be cross-referenced with `\ref`.
Default numbering can be switched off for the whole class by setting the `numbered` to false, or for an individual block by adding the class `unnumbered`.
Crossreferences may need a re-run to update.

New feature: Nested blocks may be numbered

With the new version, nested blocks are numbered. This does not always make sense – you can and probably should alter this behaviour by specifying class `.unnumbered`, because inner blocks come first in numbering sequence, not necessarily logical from a reader perspective.

New feature 1.3: Custom *styles*

yaml now allows for a new key group `styles`. Here, you can define general appearance of blocks and optional properties such as color.
This way, appearance of multiple classes can be unified even if they do not belong to the same numbering `group`

Change: Change in color specification

Colors should now be specified as hex strings starting with `#`. This is supported by most major editors, and the canonical way in quarto `brand` (the old syntax is still working)

New feature 1.4: `longref` for cross references that include block name

Cross references have previously only been possible in LaTeX style, via `\ref{thelabel}`. Now a new variant, `\longref{thelabel}`, has been added, that adds the name of the environment as a reference prefix to the resolved link. By default, the reference prefix is equal to the environment name, which can be changed as attribute `label`. The reference prefix can be redefined via the attribute `reflabel`.

To do 1.1: future plans for cross references

- ☐ should I change `label` to `blockname`? It would be breaking for those who have used it.
- ☐ consider making crossreferences using `@thelabel` as in plain quarto. Use the pandoc facilities for that

Feature 1.5: Custom lists-of blocks

Generate `.qmd` files that contains a list of selected block classes, intermitted with headers from the document for easy customization and commenting.
This way, one can make a list of all definitions, examples, or {theorems, propositions and lemmas} etc., edit it later and attach to the main document. If you edit, make sure to rename the autogenerated list first, otherwise it will get overwritten in the next run and all work is lost …
Currently, you need to give a key `listin` for any class or group of classes that should appear in a list of things. The value of this key is an array of names, also if only one list is to be generated. These names are turned into files `list-of-`name`.qmd`. I am considering replacing the yaml interface by a sub-key to `custom-numbered-classes`. This would allow to define arbitrary classes that can be attached to any custom div block, such as `.important`.

Tip: automatic inclusion of lists-of
You can include generated lists-of using a quarto shortcut. To do this, you need to compile the document twice, in order to get the list-of file.

> **Warning**
>
> You need to comment out the include shortcode until the first version of the "list-of" qmd file is generated.
> If your custom block has headers inside, it may break the div structure. To avoid this, enclose the header n another div

New features mentioned in this document:

2 Pseudomath examples

The blocks in this section are mostly `quartothmlike` blocks. The section also demonstrates that the section prefix can be changed by specifying the attribute `numberprefix` in the section heading. So here, section number"2" is replaced by "P".

Definition P.1 (Fαncybox)

A box is called *fαncybox* if it looks quite fancy.

In this context, by *fancy* we mean that the title of the box appears as a clickable button when rendered as html, where *clickable* implies that it throws a small shadow that becomes bigger when hovering over it.

Corollary P.2 (fancy is fancy)

By Definition P.1, `foldboxes` are fancyboxes.

Conjecture P.3

Students are lured into clicking on the title and unfolding the fancybox.

Theorem P.4

This extension has been written by a teacher who hopes that students read the course notes…

Theorem P.4 is suggested by Conjecture P.3, but it cannot be proven theoretically. It does give rise to more conjectures, though. Here a reference **??** to an unknown object.

Conjecture

The teacher mentioned in Theorem P.4 is a statistician who got addicted to quarto due to James J Balamuta's fantastic quarto-webr extension, and desparately wanted to have a common counter for theorems and alike. She got also convinced that everything is possible in quarto by the many nice extensions from Shafayet Khan Shafee.