

Theoretic 0.3.0

Contents

1 Summary	1	Theorem	Only Named	A2
Proposition 1.2 Foo	1	A.5	show-theorem	A3
2 Features	1	A.6	solutions	A3
3 Styling / Customization	3	A.7	theorem	A3
Lemma 3.2 Title	3	Theorem 2	Pythagoras	A4
Lesson 3.5 Foo	3	Theorem A.12	AAAAA / ZZZZZZ	A5
Lesson 3.6 Important Lesson	3	Theorem A.14	Positional	A6
3.1 Preset Styles	4	Theorem A.15	Named	A6
		Theorem A.16	Mixed	A6
A Detailed Documentation of all Exported Symbols	A1	A.8	toc	A6
A.1 qed	A1	Theorem A.17	Z	A7
A.2 restate	A1	Theorem A.18	A	A7
Proposition A.1 Funky!	A1	A.9	toc-entry	A7
A.3 show-entry-as	A1	A.10	thm-counter	A9
A.4 show-ref	A2	A.11	proof	A9
Theorem A.3 Pythagoras	A2			
		B Solutions		A9

1 Summary

This package provides opinionated functions to create theorems and similar environments.

```
#import "@preview/theoretic:0.3.0"
#import theoretic.presets.basic: *
#show ref: theoretic.show-ref

#theorem[This is a theorem.]
#proof[
    This is a proof. A QED symbol is placed correctly even
    after block equations.
    $ norm(x) = sqrt( sum_(k = 1)^d x_k ) . $
]
#proposition(<thm:foo>)[Foo][This is a named theorem.]
#proof[@thm:foo[-]][
    Proof with a list or enum?
    - No problem for QED.
]
```

Theorem 1.1. *This is a theorem.*

Proof. This is a proof. A QED symbol is placed correctly even after block equations.

$$\|x\| = \sqrt{\sum_{k=1}^d x_k}.$$

Proposition 1.2 (Foo). *This is a named theorem.*

Proof of Proposition 1.2. Proof with a list or enum?

- No problem for QED.

2 Features

- Except for `#show ref: theoretic.show-ref`, no “setup” is strictly necessary.

Customisation of the environments is achieved via parameters on the `#theoretic.theorem()` function. You can use e.g. `#let lemma = theoretic.theorem.with(kind: "lemma", supplement: "Lemma", /* ... */)`. → See Section 3

For convenience, the `theoretic.presets` module contains predefined styled environments. → See Section 3.1

- Flexible References via specific supplements. → `#theoretic.show-ref()`

```
@thm:foo vs @thm:foo[-] vs @thm:foo[--] vs @thm:foo[!]
vs @thm:foo[!] vs @thm:foo[!!!] vs @thm:foo[?] vs
@thm:foo[Statement]
```

Proposition 1.2 (Foo) vs Proposition 1.2 vs 1.2 vs
Foo (Proposition 1.2) vs Foo (1.2) vs Foo vs Propo-
sition vs Statement 1.2 (Foo)

- Any theorem can be restated.

→ `#theoretic.restate()`

```
theoretic.restate(<thm:foo>
// the head links to the original
```

Proposition 1.2 (Foo). This is a named theorem.

- Automatic numbering. If your headings are numbered, it will use top-level heading numbers as the first component, otherwise it will simply number your theorems starting with Theorem 1.

```
#theorem(number: "!!")[
    Number can be overridden per-theorem.
]
#theorem(number: 400)[
    If a `number` is passed (as opposed to a string or
    content),
]
#theorem[
    ...subsequent theorems will pick it up.
]
```

Theorem !!. Number can be overridden per-theorem.

Theorem 2.400. If a number is passed (as opposed to a string or content),

Theorem 2.401. ...subsequent theorems will pick it up.

- Custom outlines: Outline for headings *and/or* theorems.

→ `#theoretic.toc()`

- Filter for specific kinds of theorem to create e.g. a list of definitions.
- Optionally sorted alphabetically!
- Theorems can have a different title for outlines (`theorem(toctitle: ...)`) and can even have multiple entries in a sorted outline.
- Highly customizable!
 - (And this customization can be reused for regular outlines)

→ `#theoretic.toc-entry()`
→ `#theoretic.show-entry-as()`

- Automatic QED placement!

→ `theorem(fmt-suffix: ...) & #theoretic.qed()`

In most cases, it should place the QED symbol appropriately automatically:

```
#proof[This is a proof. $x=y$]
#proof[
    This is a proof.
    $ x = y $
]
#proof[
    #set math.equation(numbering: "(1)")
    This is a proof.
    $ x = y $
]
```

Proof. This is a proof. $x = y$ ■

Proof. This is a proof.

$$x = y$$
 ■

Proof. This is a proof.

$$x = y \quad (1)$$
 ■

```
#proof[
    This is a proof.
    - #lorem(3)
]
#proof[
    This is a proof.
    - #lorem(3) $ x = y $
]
#proof[
    This is a proof.
    + #lorem(3)
    + #lorem(3)
    + #lorem(3)
    + #lorem(3)
]
```

Proof. This is a proof.

► Lorem ipsum dolor. ■

Proof. This is a proof.

► Lorem ipsum dolor.

$$x = y$$
 ■

Proof. This is a proof.

1. Lorem ipsum dolor.

 1. Lorem ipsum dolor.

 1. Lorem ipsum dolor.

Specifically, it works for lists, enums, and unnumbered block equations, which may be nested. If your proof ends with some other block, you should might want to place a `#qed()` manually. For proper alignment with a block equation, use

```
#set math.equation(numbering:(..) => {qed()}, number-align: bottom)
```

placed directly in front of the equation.

- Exercise solutions: → `#theoretic.solutions()`
 - Every theorem environment can have a solution, which is shown in a separate section.
 - Solutions section automatically hides itself if there are no solutions to show.

```
#exercise(solution: [/*******/])[  
  Go look for the solution of this exercise at the end of  
  this document.  
]
```

Exercise 2.402. Go look for the solution of this exercise at the end of this document.

3 Styling / Customization

For basic customization, you can override the supplement, kind, and options parameters of `#theoretic.theorem()`.

You can start completely fresh:

```
#import "@preview/theoretic:0.3.0"  
#show ref: theoretic.show-ref  
  
// simply setting `variant` to one of "plain", "remark", "definition" or "important" changes the style:  
#let theorem = theoretic.theorem.with(supplement: "Theorem", kind: "theorem", variant: "important")  
// you can also set the other style options directly:  
#let lemma = theoretic.theorem.with(  
  supplement: "Lemma", kind: "lemma",  
  options: (  
    head-font: (style: "normal", weight: "bold"),  
    title-font: (style: "italic", weight: "regular", fill: oklch(40%, 0.2, 12deg)),  
    body-font: (style: "normal", weight: "regular"),  
    block-args: (outset: 4pt, fill: oklch(95%, 0.06, 12deg)),  
    head-punct: none,  
    head-sep: h(1em),  
  ),  
)  
  
#lorem(5)  
#theorem[#lorem(5)]  
#lemma[Title][#lorem(5)]
```

Lorem ipsum dolor sit amet.

Theorem 3.1. Lorem ipsum dolor sit amet.

Lemma 3.2 (Title) Lorem ipsum dolor sit amet.

See the documentation in the appendix for more details on the options parameter.

Alternatively, you can also build upon a preset style:

```
#import "@preview/theoretic:0.3.0"  
#import theoretic.presets.fancy: *  
#show ref: theoretic.show-ref  
  
// this is immediately useful for translations:  
#let satz = theorem.with(supplement: "Satz")  
#satz[Eine deutsche Aussage.]  
#theorem[An English theorem.]  
  
// or to add new kinds:  
#let lesson = theorem.with(supplement: "Lesson", kind:  
  "lesson", options: (hue: 12deg), variant: "plain")  
#lesson[Foo][Bar]  
#lesson(variant: "important")[Important Lesson][...]
```

Satz 3.3 Eine deutsche Aussage.

Theorem 3.4 An English theorem.

Lesson 3.5 **Foo** Bar

Lesson 3.6 Important Lesson The variant parameter is also intended to be called for single theorems. E.g. in this fancy preset, "important" adds a line above the supplement.

Note that not all preset styles respect the same options. More details are given in the examples in Section 3.1.

If you want to go in a completely new direction, you can also provide your own `show-theorem` function to fully control styling. For how this can look, I recommend looking at how the predefined styles are made: See the code on GitHub.

3.1 Preset Styles

Preset “basic”

Use with: `#import theoretic.presets.basic: *`

This style uses the built-in `#theoretic.show-theorem()` and accepts all its options.

Theorem 3.7. This is an example theorem created using `#theorem[...]`.

Proposition 3.8. Using `#proposition[...]`.

Lemma 3.9. Using `#lemma[...]`.

Corollary 3.10. Using `#corollary[...]`.

Algorithm 3.11. Using `#algorithm[...]`.

Axiom 3.12. Using `#axiom[...]`.

Definition 3.13. Using `#definiton[...]`.

Exercise 3.14. Using `#exercise[...]`.

Example. Using `#example[...]`.

Counter-Example. Using `#counter-example[...]`.

Remark. Using `#remark[...]`.

Note. Using `#note[...]`.

Claim. Using `#claim[...]`.

Proof. Using `#proof[...]`.

Theorem 3.15 (Title). This is an example theorem created using `#theorem(toctitle: none)[Title][...]`.

Proposition 3.16 (Title). *Lore ipsum dolor.*

Lemma 3.17 (Title). *Lore ipsum dolor.*

Corollary 3.18 (Title). *Lore ipsum dolor.*

Algorithm 3.19 (Title). *Lore ipsum dolor.*

Axiom 3.20 (Title). *Lore ipsum dolor.*

Definition 3.21 (Title). *Lore ipsum dolor.*

Exercise 3.22 (Title). *Lore ipsum dolor.*

Example (Title). *Lore ipsum dolor.*

Counter-Example (Title). *Lore ipsum dolor.*

Remark (Title). *Lore ipsum dolor.*

Note (Title). *Lore ipsum dolor.*

Claim (Title). *Lore ipsum dolor.*

Proof of Title. *Lore ipsum dolor.*

Preset “fancy”

Use with: `#import theoretic.presets.fancy: *`

This style is intended¹ for use with a font that supports stretch: 85% and weight: "semibold". It is here shown using the font "Besley*", which you can download on GitHub.

This style ignores most options, except for a new hue option to set the oklch hue. The variant can be "muted", "remark", "plain", or "important".

Theorem 3.24 This is an example theorem created using `#theorem[...]`.

Proposition 3.25 Using `#proposition[...]`.

Lemma 3.26 Using `#lemma[...]`.

Corollary 3.27 Using `#corollary[...]`.

Algorithm 3.28 Using `#algorithm[...]`.

Axiom 3.29 Using `#axiom[...]`.

Definition 3.30 Using `#definiton[...]`.

Exercise 3.31 Using `#exercise[...]`.

Example 3.32 Using `#example[...]`.

Counter-Example 3.33 Using `#counter-example[...]`.

Remark Using `#remark[...]`.

Note Using `#note[...]`.

Claim Using `#claim[...]`.

Proof. Using `#proof[...]`.

Theorem 3.34 **Title** This is an example theorem created using `#theorem(toctitle: none)[Title][...]`.

Proposition 3.35 **Title** *Lore ipsum dolor.*

Lemma 3.36 **Title** *Lore ipsum dolor.*

Corollary 3.37 **Title** *Lore ipsum dolor.*

Algorithm 3.38 **Title** *Lore ipsum dolor.*

Axiom 3.39 **Title** *Lore ipsum dolor.*

Definition 3.40 **Title** *Lore ipsum dolor.*

Exercise 3.41 **Title** *Lore ipsum dolor.*

Example 3.42 **Title** *Lore ipsum dolor.*

Counter-Example 3.43 **Title** *Lore ipsum dolor.*

Remark **Title** *Lore ipsum dolor.*

Note **Title** *Lore ipsum dolor.*

Claim **Title** *Lore ipsum dolor.*

¹It still looks okay in other fonts, but it does not reach full potential. Compare: **Lemma 3.23** **Lore** ipsum.

Preset “bar”

Use with: `#import theoretic.presets.bar: *`

This style ignores most options, except for head-font, and it adds a color option. The variant can be “plain” or “important”. (Not all environments shown.)

Theorem 3.44

This is an example theorem created using `#theorem[...]`.

Lemma 3.45

Using `#lemma[...]`.

Example

Using `#example(options: (color: red))[...]`.

Proposition 3.46

This is an example theorem created using `#proposition(toctitle: none)[Title][...]`.

Lemma 3.47

Using `#lemma(variant: "important")[...]`.

Proof of Title: Using `#proof[...]`. ■

Preset “corners”

Use with: `#import theoretic.presets.corners: *`

This style is almost identical to the basic one, it just wraps the environments in a block with corners. It adds a color option. (Not all environments shown.)

└ **Theorem 3.48.** This is an example theorem created using `#theorem[...]`.

└ **Lemma 3.49.** Using `#lemma[...]`.

└ **Example.** Using `#example(options: (color: red))[...]`.

└ **Proposition 3.50 (Title).** This is an example theorem created using `#proposition(toctitle: none)[Title][...]`.

└ **Lemma 3.51.** Using `#lemma(variant: "important")[...]`.

└ *Proof of Title.* Using `#proof[...]`. ■

A Detailed Documentation of all Exported Symbols

A.1 qed

Place a QED mark and clear the `_thm_needs_qed` flag, so that the theorem environment itself won't place one.

Pass this as `theorem fmt-suffix` for proof environments.

Parameters

```
qed(suffix: sym.qed content, force: boolean) → content
```

`force: true` `boolean` Whether to place suffix no matter the `_thm_needs_qed` flag.

A.2 restate

Re-state a theorem.

It will reuse the original kind, supplement, number, title, body, and styling. It will *not* re-emit the solution or label, and it will use `toctitle: none` to avoid duplicate toc entries.

```
#let proposition = theorem.with(
  kind: "proposition",
  supplement: "Proposition",
  options: (
    head-font: (fill: blue),
  )
)
#proposition(<funky>)[Funky!][Blah _blah_ blah.]
Restated:
#restate("funky")
Restated with added customizations:
#restate(<funky>, options: (body-font: (fill: red)))
```

Proposition A.1 (Funky!). Blah blah blah.

Restated:

Proposition A.1 (Funky!). Blah blah blah.

Restated with added customizations:

Proposition A.1 (Funky!). Blah blah blah.

Parameters

```
restate(<label>: label string, <..args>) → content
```

`<label>` `label` or `string` Positional Label of the theorem to restate.

`<..args>` Positional

Override arguments for the theorem function.

(I don't recommend changing anything here except possibly `show-theorem` and `options`.)

A.3 show-entry-as

Helper function to adapt actual outlines to look the same as those made with `toc()`. This is useful if you want to have e.g. a list of figures and a list of definitions and want them to share their style.

Note: For Typst versions ≤ 0.12 , this function is a bit "hacky" and might not always work. (It deconstructs the `outline.entry` based on heuristics.)

```
#import theoretic: show-entry-as, toc-entry

#outline(target: figure, title: [Typst Default])

#show outline.entry: show-entry-as(toc-entry.with(hanging-indent: 60pt, /*...*/))
#outline(target: figure, title: [Using `theoretic.toc-entry`])

#figure(
```

```

caption: [Example Figure],
block(height: 2em, width: 100%, fill: gradient.linear(..color.map.viridis))
)

```

Typst Default

Figure 1 Example Figure A2

Using `theoretic.toc-entry`

Figure 1 Example Figure A2



Figure 1: Example Figure

Parameters

```
show-entry-as(<toc-entry>: function)
```

<toc-entry> function Positional

Customize `toc-entry()` used.

Expects a function taking five positional arguments (level, target, prefix, body, page).

A.4 show-ref

Show-rule-function to be able to @ labelled theorems.

Use via `#show ref: show-ref` at the beginning of your document.

```

@show ref: theoretic.show-ref
#theorem(label: <fact>, supplement: "Fact")[#lorem(2)]
#theorem(label: <pythagoras>, "Pythagoras")[#lorem(2)]
#theorem(label: <z1>, title: "Only Named", number: none)
[#lorem(2)]
#theorem(label: <y>, number: "Y")[#lorem(2)]
#theorem(label: "5", number: none)[#lorem(2)]

As a consequence of @fact and @pythagoras[!]...

```

Fact A.2. *Loem ipsum.*

Theorem A.3 (Pythagoras). *Loem ipsum.*

Theorem (Only Named). *Loem ipsum.*

Theorem Y. *Loem ipsum.*

Theorem. *Loem ipsum.*

As a consequence of Fact A.2 and Pythagoras (A.3)...

The reference can be controlled via the supplement passed:

	BOTH	WITHOUT TITLE	WITHOUT NUMBER	NEITHER
@ref (Full)	Theorem A.3 (Pythagoras)	Fact A.2 / Theorem Y	Theorem (Only Named)	Theorem
@ref[-] (Compact)	Theorem A.3	Fact A.2 / Theorem Y	Theorem (Only Named)	Theorem
@ref[--] (Number)	A.3	A.2 / Y	(Only Named)	Theorem
@ref[!] (Inverted)	Pythagoras (Theorem A.3)	Fact A.2 / Theorem Y	Only Named (Theorem)	Theorem
@ref[!] (Compact Inverted)	Pythagoras (A.3)	Fact A.2 / Theorem Y	Only Named	Theorem
@ref[!!!] (Name)	Pythagoras	Fact A.2 / Theorem Y	Only Named	Theorem
@ref[?] (Kind)	Theorem	Fact / Theorem	Theorem	Theorem
@ref[Custom] (Custom Supplement)	Custom A.3 (Pythagoras)	Custom A.2 / Custom Y	Custom (Only Named)	Custom

Note: the fact that references and links in this document are underlined in gray is achieved with a separate `@show` link: `it => underline(..)` rule, and not because of this function.

Parameters

```
show-ref(<it>: ref)
```

A.5 show-theorem

Default “show” function for theorems. Note that in your versions of this, you cannot use `it` to generate the default options, but you can fall back to `theoretic.show.theorem(it)`.

Parameters

```
show-theorem(<it>: dictionary)
```

`<it>` `dictionary` Positional

A dictionary with keys:

- `supplement`: `content`
- `number`: `content` | `none`
- `title`: `content` | `none`
- `body`: `content`
- `variant`: `str`
- `options`: `dictionary`

Note that the suffix is already added to the body at this point.

Also, note that the variant is already used when filling the options dictionary with defaults. For the expected keys of options, see `theorem.options`

A.6 solutions

List all solutions, if any. Should not be used more than once in a document, as this might break links.

See Section B for how it looks.

Parameters

```
solutions(title: content, theorem-function: function, supplement: content) → content
```

`title: "Solutions"` `content` Title/heading to use.

`theorem-function: theorem` `function` Which theorem function to use for the solutions.

`supplement: "Solution"` `content`

Supplement to use for the solution theorems. Change this e.g. for localization.

A.7 theorem

Theorem Environment

```
#set heading(numbering: none)

#theorem[If the headings are not numbered, theorem
numbering starts at 1.]
```

```
= Heading
#theorem(title: "Pythagoras")[
    Given a right-angled triangle, the length
    of the hypotenuse squared is equal to the
    sum of the squares of the remainig sides'
    lengths.
]
```

Theorem 1. If the headings are not numbered, theorem numbering starts at 1.

Heading

Theorem 2 (Pythagoras). Given a right-angled triangle, the length of the hypotenuse squared is equal to the sum of the squares of the remainig sides' lengths.

Parameters

```
theorem(
    show-theorem: function ,
    options: dictionary ,
    variant: str ,
    fmt-suffix: function none ,
    kind: string ,
    supplement: content ,
    number: auto none integer content ,
    title: none content ,
    totitle: auto none content array ,
    label: none label string ,
    solution: none content ,
    <..unnamed-and-body>: arguments
) → content
```

show-theorem: show-theorem function

This function is used to show the actual theorem. I recommend looking at the code and documentation for the default `show-theorem()` to see how this would look.

options: (:) dictionary Additional options that are passed to `show-theorem`.

The default `show-theorem` expects the following keys:

- head-font: dict // options for the head text
- title-font: dict // title is placed inside the head
- body-font: dict // options for the body text
- block-args: dict // options for the block
- head-punct: content // placed at the end of the head
- head-sep: content // placed after the head
- link: none | (link target) // The target the head should link to.

If you are using a custom `show-theorem`, you can also add more fields here.

This will be filled with defaults depending on the `variant`.

variant: "plain" str This controls the defaults for `options`. It is also passed to `show-theorem`.

fmt-suffix: none function or none

Will be called at the end of the theorem if `_thm_needs_qed` hasn't been cleared. (E.g. by `qed()`)

kind: "theorem" string Used for filtering e.g. when creating table of theorems.

supplement: "Theorem" content

What to label the environment.

It is recommended to keep kind and supplement matching (except for “subtypes”, e.g. one might have the kind of “Example” and “Counter-Example” both as “example”)

`number: auto` `auto` or `none` or `integer` or `content`

- If `auto`, will continue numbering from last numbered theorem.
- If `integer`, it will continue the numbering of later theorems from the given number.
- If `content`, it is shown as-is, with no side-effects.

```
#let corollary = theorem.with(
  kind: "corollary",
  supplement: "Corollary")

#corollary[#lorem(2)]

#corollary(number: none)[Skip number]
#corollary[Resume numbering]

#corollary(number: "X")[Custom "number"]
#corollary[Resume numbering]

#corollary(number: 10)[Set number]
#corollary[Continue from set number]
```

Corollary A.3. *Lorem ipsum.*

Corollary. *Skip number*

Corollary A.4. *Resume numbering*

Corollary X. *Custom “number”*

Corollary A.5. *Resume numbering*

Corollary A.10. *Set number*

Corollary A.11. *Continue from set number*

`title: none` `none` or `content` Title of the Theorem. Usually shown in parentheses after the number.

This can also be passed as a positional argument.

`toctitle: auto` `auto` or `none` or `content` or `array`

Title of the Theorem to be used in outlines.

- `auto` to use the title.
- `none` to hide it from the outlines.

If you pass an array, in `sorted` outlines (`toc.sort`) it will be split into multiple entries. All but the first one are marked as secondary.

```
#theorem(
  title: [A to Z],
  toctitle: ([AAAAA], [ZZZZZZ])
)[
  Compare how this appears in different outlines!
]
```

Theorem A.12 (A to Z). *Compare how this appears in different outlines!*

`label: none` `none` or `label` or `string`

Label (for references)

This can also be passed as a positional argument. In that case it must be a label and not a string.

NB: Simply putting a `<label>` after the `#theorem[]` does not work for referencing.

`solution: none` `none` or `content`

Optional Solution. See also `solutions()`.

```
#theorem(solution: [This will show up wherever
`#theoretic.solutions()` is placed.])[#lorem(5)]
```

Theorem A.13. *Lorem ipsum dolor sit amet.*

```
<..unnamed-and-body>    arguments    Positional
```

The last positional argument given is used as the theorem body.

Other positional arguments are used for the title and label, depending on their type.

```
// Any of these work:  
#theorem(<positional>)[Positional][#lorem(4)]  
#theorem(label: <named>, title: [Named])[#lorem(4)]  
#theorem([Mixed], label: <mixed>)[#lorem(4)]
```

Theorem A.14 (Positional). *Lorem ipsum dolor sit.*

Theorem A.15 (Named). *Lorem ipsum dolor sit.*

Theorem A.16 (Mixed). *Lorem ipsum dolor sit.*

A.8 toc

Create an outline that includes named theorems.

Can be styled with show rules for `outline.entry()`. See the source code of this manual for an example.

```
#heading(outlined: false, level: 3)[  
  Contents  
]  
#toc(depth: 1)
```

Contents

1	Summary	1
	Proposition 1.2 Foo	1
2	Features	1
3	Styling / Customization	3
	Lemma 3.2 Title	3
	Lesson 3.5 Foo	3
	Lesson 3.6 Important Lesson	3
A	Detailed Documentation of all Exported Symbols	A1
	Proposition A.1 Funky!	A1
	Theorem A.3 Pythagoras	A2
	Theorem Only Named	A2
	Theorem 2 Pythagoras	A4
	Theorem A.12 AAAA / ZZZZZ	A5
	Theorem A.14 Positional	A6
	Theorem A.15 Named	A6
	Theorem A.16 Mixed	A6
	Theorem A.17 Z	A7
	Theorem A.18 A	A7
B	Solutions	A9

Parameters

```
toc(  
  depth: integer,  
  exclude: array (string),  
  level: integer auto,  
  toc-entry: function,  
  sort: bool  
) → content
```

depth: 2 **integer** Maximum depth of headings to consider

```
exclude: ("proof", "solution")    array (string)    list of theorem.kinds to ignore.
```

```
#heading(outlined: false, level: 3)[
  Table of Examples
]
#toc(
  depth: 0,
  exclude: ("proof", "solution", "theorem")
)
```

Table of Examples

Proposition 1.2 Foo	1
Lemma 3.2 Title	3
Lesson 3.5 Foo	3
Lesson 3.6 Important Lesson	3
Proposition A.1 Funky!	A1

level: auto integer or auto Fake level to use for theorems. If auto, it will use depth + 1.

```
toc-entry: toc-entry function
```

Customize toc-entry() used.

Expects a function taking five positional arguments (level, target, prefix, body, page).

```
sort: false bool    Whether to sort the entries alphabetically.
```

Only respected if depth is 0.

If true, this will also split entries where toctitle is an array into separate entries.

```
#theorem("Z")[Blah blah.]
#theorem("A")[Blah blah.]
#heading(outlined: false, level: 3)[
  Sorted Table of Theorems
]
#set text(size: 9pt)
#toc(
  depth: 0,
  sort: true,
  toc-entry: toc-entry.with(hanging-indent: 60pt),
)
```

Theorem A.17 (Z). Blah blah.

Theorem A.18 (A). Blah blah.

Sorted Table of Theorems

Theorem A.18 A	A7
Theorem A.12 AAAAA	A5
Proposition 1.2 Foo	1
Lesson 3.5 Foo	3
Proposition A.1 Funky!	A1
Lesson 3.6 Important Lesson	3
Theorem A.16 Mixed	A6
Theorem A.15 Named	A6
Theorem Only Named	A2
Theorem A.14 Positional	A6
Theorem A.3 Pythagoras	A2
Theorem 2 Pythagoras	A4
Lemma 3.2 Title	3
Theorem A.17 Z	A7
Theorem A.12 (ZZZZZZ)	A5

A.9 toc-entry

Create a toc entry.

Pass this to toc() using .with(..) to customize the fmt- parameters used.

This is used because since Typst 0.13, it is no longer possible to call outline.entry outside of an actual ourline element, and one “cannot outline metadata”.

This manual uses

```

set par(justify: false)
let indents = (0pt, 15pt, 37pt)
let hang-indents = (15pt, 22pt, 54pt)
let text-styles = ((weight: 700), (size: 10pt), (size: 9pt, weight: 500), (size: 9pt, fill: luma(20%)), )
theoretic.toc(toc-entry: theoretic.toc-entry.with(
  indent: (level) => { indents.at(level - 1) },
  hanging-indent: (level) => { hang-indents.at(level - 1) },
  fmt-prefix: (prefix, level, _s) => {
    set text(..text-styles.at(level - 1), number-width: "tabular")
    prefix
    h(4pt)
  },
  fmt-body: (body, level, _s) => {
    set text(..text-styles.at(level - 1))
    body
  },
  fmt-fill: (level, _s) => {
    if level == 2 {
      set text(..text-styles.at(2))
      box(width: 1fr, align(right, repeat(gap: 9pt, justify: false, [.])))
    }
  },
  fmt-page: (page, level, _s) => {
    set text(..text-styles.at(level - 1), number-width: "tabular")
    box(width: 18pt, align(right, [#page]))
  },
  above: (level) => {
    if level == 1 {
      auto // paragraph spacing
    } else {
      7pt
    }
  },
  below: auto,
))

```

Parameters

```

toc-entry(
  <level>: int,
  <target>: location,
  <prefix>: content |none|,
  <body>: content,
  <page>: content,
  secondary: boolean,
  indent: relative length |function|,
  hanging-indent: relative length |function| auto,
  above: relative length |function|,
  below: relative length |function|,
  fmt-prefix: (prefix, level, secondary) => if prefix != none {
    prefix
    h(0.5em, weak: false)
  } |function|,
  fmt-body: (body, level, secondary) => if secondary [(#body)] else [#body] |function|,
  fmt-fill: (level, secondary) => box(width: 1fr, repeat[.~]) |function|,
  fmt-page: (page, level, secondary) => page |function|
)

secondary: false  boolean

```

This is true for entries where the toc-title is an array, the entry was split and this is *not* the first one (in order specified).

`indent: 1em` relative length or function How much to indent each entry.

- If `relative length`, it will be multiplied with level - 1.
 - If `function`, will be called with the level as argument.

hanging-indent: `auto` | `relative length` or `function` or `auto`

How much more to indent subsequent lines (in addition to `toc-entry.indent`).

If the prefix is shorter than this, this will lead to a gap between prefix and body; If the prefix is longer, the body will start immediately after the prefix.

- If `function`, will be called with the level as argument.
 - If `auto`, will use the width of the prefix

```
#let example-entry = theoretic.toc-entry.with(1, here(),  
[Section 1.], lorem(6), [0])  
#let example-entry-2 = theoretic.toc-entry.with(2, here(),  
[Section 1.1.], lorem(6), [0])  
  
// aligned with end of prefix  
#example-entry(hanging-indent: auto)  
#example-entry-2(hanging-indent: auto)  
  
#example-entry(hanging-indent: 1em)  
#example-entry-2(hanging-indent: 1em)  
#example-entry(hanging-indent: 80pt)  
#example-entry-2(hanging-indent: 80pt)
```

If `function`, will be called with the level as argument.

below: 0.7em relative length or function If function, will be called with the level as argument.

A.10 thm-counter counter

Counts theorems.

In most cases, it is not necessary to reset this manually, it will get updated accordingly if you pass an integer to `theorem.number`.

A.11 proof function

This is simply `theorem()` with different options.

B Solutions

Solution (Exercise 2.402). *Yay! you found the solutions!*

Solution (Theorem A.13). This will show up wherever `#theoretic.solutions()` is placed.