

# lemmify

Lemmify is a library for typesetting mathematical theorems in typst. It aims to be easy to use while trying to be as flexible and idiomatic as possible. This means that the interface might change with updates to typst (for example if user-defined element functions are introduced). But no functionality should be lost.

If you are encountering any bugs, have questions or are missing features, feel free to open an issue on GitHub.

## Basic usage

1. Import lemmify:

```
#import "@preview/lemmify:0.2.0": default-theorems, select-kind
```

2. Generate some common theorem kinds with pre-defined style:

```
#let (  
  theorem, lemma, corollary,  
  remark, proposition, example,  
  proof, theorem-rules  
) = default-theorems(lang: "en")
```

3. Apply the generated style:

```
#show: theorem-rules
```

4. Customize the theorems using show rules. For example, to add a block around proofs:

```
#show select-kind(proof): block.with(  
  breakable: true,  
  width: 100%,  
  fill: gray,  
  inset: 1em,  
  radius: 5pt  
)
```

5. Create theorems, lemmas, and proofs:

```
#theorem(name: "Some theorem") [  
  Theorem content goes here.  
]<thm>  
  
#theorem(numbering: none) [  
  Another theorem.  
]  
  
#proof(link-to: <thm>)[  
  Complicated proof.  
]<proof>  
  
@proof and @thm[theorem]
```

The result should now look something like this:

**Theorem 1** (*Some theorem*) Theorem content goes here.

**Theorem** Another theorem.

**Proof** Complicated proof. □

Proof 1 and theorem 1

## Examples

By default theorems are reset on every heading. This can be changed with the `max-reset-level` parameter of `default-theorems()`. This also changes which heading levels will be used in the numbering. To not reset them at all `max-reset-level` can be set to 0.

```
#import "@preview/lemmify:0.2.0": default-theorems

#let (
  theorem, theorem-rules
) = default-theorems(max-reset-level: 2)
#show: theorem-rules
#set heading(numbering: "1.1")

== Heading
#theorem(lorem(5))
== Heading
#theorem(lorem(5))
=== Heading
#theorem(lorem(5))
```

## 1 Heading

**Theorem 1.1** Lorem ipsum dolor sit amet.

### 1.1 Heading

**Theorem 1.1.1** Lorem ipsum dolor sit amet.

#### 1.1.1 Heading

**Theorem 1.1.2** Lorem ipsum dolor sit amet.

Each theorem belongs to a group and every group shares one counter. The theorems created by `default-theorems()` all belong to the same group, except for proofs. You can create separate groups by passing a group parameter to `default-theorems()`. The next example shows how to create separately numbered examples.

```
#import "@preview/lemmify:0.2.0": default-theorems

#let (theorem, theorem-rules) = default-theorems()
#show: theorem-rules
#let (
  example, theorem-rules
) = default-theorems(group: "example-group")
```

```
#show: theorem-rules
```

```
#theorem(lorem(5))  
#example(lorem(5))  
#example(lorem(5))  
#theorem(lorem(5))
```

**Theorem 1** Lorem ipsum dolor sit amet.

**Example 1** Lorem ipsum dolor sit amet.

**Example 2** Lorem ipsum dolor sit amet.

**Theorem 2** Lorem ipsum dolor sit amet.

The link-to parameter can be used to link theorems to other content. By default theorems are linked to the last heading and proofs are linked to the last theorem. This example shows how corollaries can be linked to the last theorem. Note that it's fine to only apply the theorem-rules once here since both theorem-kinds belong to the same group.

```
#import "@preview/lemmify:0.2.0": default-theorems, select-kind, reset-counter  
  
#let (theorem, theorem-rules) = default-theorems()  
#let (corollary,) = default-theorems(  
  group: "corollary-group",  
  link-to: select-kind(theorem)  
)  
#show: theorem-rules  
#show select-kind(theorem): it => {it; reset-counter(corollary)}  
  
#theorem(lorem(5))  
#corollary(lorem(5))  
#corollary(lorem(5))  
#theorem(lorem(5))  
#corollary(lorem(5))
```

**Theorem 1** Lorem ipsum dolor sit amet.

**Corollary 1.1** Lorem ipsum dolor sit amet.

**Corollary 1.2** Lorem ipsum dolor sit amet.

**Theorem 2** Lorem ipsum dolor sit amet.

**Corollary 2.1** Lorem ipsum dolor sit amet.

The best and easiest way to change the look of theorems is to use show-rules. The next example shows another way how the appearance of theorems can be changed.

```
#import "@preview/lemmify:0.2.0": default-theorems, style-simple  
  
#let (  
  theorem, proof, theorem-rules
```

```

) = default-theorems(
  lang: "en",
  style: style-simple.with(seperator: ". "),
  proof-style: style-simple.with(kind-name-style: emph, seperator: ". ")
)
#show: theorem-rules

#theorem(lorem(5))
#proof(lorem(5))

```

**Theorem 1.** Lorem ipsum dolor sit amet.

*Proof.* Lorem ipsum dolor sit amet.

Doing the same thing to remarks is a bit more complicated since the style parameter applies to both theorems and remarks.

```

#import "@preview/lemmify:0.2.0": default-theorems, style-simple

#let (
  theorem, theorem-rules
) = default-theorems(
  lang: "en",
  style: style-simple.with(seperator: ". ")
)
#let (remark,) = default-theorems(
  style: style-simple.with(kind-name-style: emph, seperator: ". "),
  numbering: none
)
#show: theorem-rules

#theorem(lorem(5))
#remark(lorem(5))

```

**Theorem 1.** Lorem ipsum dolor sit amet.

*Remark.* Lorem ipsum dolor sit amet.

If the pre-defined styles are not customizable enough you can also provide your own style.

```

#import "@preview/lemmify:0.2.0": default-theorems, get-theorem-parameters

#let custom-style(thm) = {
  let params = get-theorem-parameters(thm)
  let number = (params.numbering)(thm, false)
  block(
    inset: .5em,
    fill: gray,
    {
      params.kind-name + " "
      number
      if params.name != none { ": " + params.name }
    }
  )
}

```

```

    }
  )
  v(0pt, weak: true)
  block(
    width: 100%,
    inset: 1em,
    stroke: gray + 1pt,
    params.body
  )
}

#let (
  theorem, theorem-rules
) = default-theorems(lang: "en", style: custom-style)
#show: theorem-rules

#theorem(name: "Some theorem") [
  #lorem(40)
]

```

#### Theorem 1: Some theorem

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua quaerat voluptatem. Ut enim aequi doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere.

There is one other way to create `theorem-functions`: the `theorem-kind()` function. It is used to create the theorem-functions returned by `default-theorems()` so it behaves almost the same. The only difference is that there is no `max-reset-level` parameter and that no `theorem-rules` are returned. A default rule which does not reset any theorem counters can be imported.

```

#import "@preview/lemmify:0.2.0": theorem-kind, theorem-rules

#let note = theorem-kind("Note")
#show: theorem-rules

#note(lorem(5))

```

**Note 1** Lorem ipsum dolor sit amet.

## Documentation

### `theorem-kind`

Creates a new `theorem-function`.

## Parameters

```
theorem-kind(  
  kind-name: str,  
  group: str,  
  link-to: label selector selector-function none,  
  numbering: theorem-numbering-function none,  
  subnumbering: numbering-function str none,  
  style: style-function  
) -> theorem-function
```

### kind-name str

The name of the theorem kind. It also acts as an identifier together with group when using `select-kind`, so it should be unique.

### group str

The group identifier. Each theorem group shares one counter.

Default: `LEMMIFY-DEFAULT-THEOREM-GROUP`

### link-to label or selector or selector-function or none

This parameter sets what the `theorems` created by the `theorem-function` will be linked to by default.

Default: `last-heading`

### numbering theorem-numbering-function or none

Specify a default value for the numbering parameter of the `theorem-function`.

Default: `numbering-concat`

### subnumbering numbering-function or str or none

The subnumbering is needed to convert the `theorems` counter to content, which is then used in the `theorem-numbering-function`.

Default: `"1"`

### style style-function

Specifies how the `theorems` will look. This will only be visible once the `theorem-rules()` have been applied.

Default: `style-simple`

## theorem-rules

Apply the style of every `theorem` and handle references to `theorems`.

### Parameters

```
theorem-rules(content: content) -> content
```

## default-theorems

Generate a few common theorem kinds in the specified language.

Returns a dictionary of the form (theorem, lemma, corollary, remark, proposition, example, definition, proof, theorem-rules). The theorem-rules can be applied using a show statement. Additionally to showing the theorem styles and handling references they also reset the counters according to `max-reset-level`

This function accepts all parameters of `theorem-kind()` once for proofs and once for all kinds except for proofs.

### Parameters

```
default-theorems(  
  group: str,  
  proof-group: str,  
  lang: str,  
  style: style-function,  
  proof-style: style-function,  
  numbering: theorem-numbering-function none,  
  proof-numbering: theorem-numbering-function none,  
  link-to: label selector selector-function none,  
  proof-link-to: label selector selector-function none,  
  subnumbering: numbering-function str none,  
  max-reset-level: int none  
) -> dictionary
```

**lang**    `str`

The language in which the theorem kinds are generated.

Default: `"en"`

**max-reset-level**    `int` or `none`

If it is not none the theorem counter will be reset on headings below `max-reset-level`. And if `link-to` is set to `last-heading` higher levels will not be displayed in the numbering.

Default: `none`

## Function types

### theorem-function

The functions that actually create the theorems. The default values of the arguments are set to the values provided in `theorem-kind()` or `default-theorems()`.

## Parameters

```
theorem-function(  
  name: content str,  
  link-to: label selector selector-function none,  
  numbering: theorem-numbering-function none,  
  body: content  
) -> theorem
```

**name**    content or str

The name of the theorem.

Default: none

**link-to**    label or selector or selector-function or none

Link the theorem to some other content. For labels and selectors the last match before the theorem is used.

Default: theorem-kind.link-to

**numbering**    theorem-numbering-function or none

See theorem-numbering-function for more information. Can be set to none for unnumbered theorems.

Default: theorem-kind.numbering

## theorem-numbering-function

Create combined numberings from theorem and the content linked to it.

There are two pre-defined theorem-numbering-functions: numbering-concat() and numbering-proof().

## Parameters

```
theorem-numbering-function(  
  thm: theorem,  
  referenced: bool  
) -> content
```

**thm**    theorem

The theorem for which the numbering should be generated. See also get-theorem-parameters().

**referenced**    bool

This is false if the numbering was requested from the theorem it belongs to. Otherwise it is false. See numbering-proof() as an example.



## style-function

Defines how the `theorem` will look. Use `get-theorem-parameters()` to get all information stored in the `theorem`.

There are two pre-defined `style-functions`: `style-simple()` and `style-reversed()`.

### Parameters

```
style-function(thm: theorem) -> content
```

## selector-function

Useful for more advanced queries. See `last-heading()` for an example.

### Parameters

```
selector-function(loc: location) -> content none
```

**loc**    `location`

When used in `link-to` parameter of some `theorem` its location will be passed when resolving the link with `resolve-link()`.

## numbering-function

A normal numbering function as described in the [typst documentation](#).

### Parameters

```
numbering-function(..state: int) -> content
```

## theorem

A `theorem` is a figure with some additional information stored in one of its parameters.

### is-theorem

Check if argument is `theorem`.

### Parameters

```
is-theorem(c: any) -> bool
```

## get-theorem-parameters

Extract theorem parameters from figure. Returns a dictionary of the form (body, group, kind-name, name, link-to, numbering, subnumbering, style).

### Parameters

```
get-theorem-parameters(thm: theorem) -> dictionary
```

## resolve-link

Return the content that is linked to the `theorem`.

## Parameters

`resolve-link(thm: theorem) -> content`

## numbered

A `numbered` is a heading, page, `math.equation` or figure that is already embedded in the document (that means it was obtained by a query). The numbering also has to be different from none.

## is-numbered

Check if argument is `numbered`.

## Parameters

`is-numbered(n: any) -> bool`

## display-numbered

Display the numbering of the argument at its location.

## Parameters

`display-numbered(n: numbered) -> content`

## Styles

### numbering-concat

If the linked content is numbered combine it with the numbering of the `theorem`.

## Parameters

```
numbering-concat(  
  thm: theorem,  
  referenced: bool,  
  seperator: content str  
)
```

**seperator**   `content` or `str`

The sepearator is put between both numberings.

Default: `"."`

### numbering-proof

Copy the numbering of a linked `theorem` if referenced. Otherwise no numbering is returned.

## Parameters

```
numbering-proof(  
  thm: theorem,  
  referenced: bool  
)
```

### qed-box

A box for convenience. (Not a function but a constant.)

## Parameters

`qed-box()`

## style-simple

Simple theorem style. The theorem gets represented as a breakable block of the form `kind-name-style(kind-name) number-style(numbering) name-style(name) seperator body`.

## Parameters

```
style-simple(  
  thm: theorem,  
  kind-name-style: function,  
  number-style: function,  
  name-style: function,  
  seperator: content str,  
  qed: content none  
)
```

### kind-name-style    function

A function `str -> content` to change the look of the kind-name.

Default: strong

### number-style    function

A function `content -> content` to change the look of the generated numbering.

Default: strong

### name-style    function

A function `content -> content` to change the look of the name.

Default: name => `emph("(" + name + ")")`

### seperator    content or str

How to seperate the theorem header and its body.

Default: " "

### qed    content or none

Select what content to show at the end of the theorem.

Default: none

## style-reversed

Reverses numbering and kind-name, otherwise the same as `style-simple()`.

### Parameters

```
style-reversed(  
  thm: theorem,  
  kind-name-style: function,  
  number-style: function,  
  name-style: function,  
  seperator: content str,  
  qed: content none  
)
```

## Selectors

The selectors can be used in show-rules to customize the `theorems` styling as well as with the `link-to` parameter.

### last-heading

Selector-function which selects the last heading.

#### Parameters

```
last-heading(  
  ignore-unnumbered: bool,  
  max-level: int none,  
  loc: location  
) -> heading none
```

**ignore-unnumbered**    `bool`

Use the last heading which is numbered.

Default: `false`

**max-level**    `int` or `none`

Do not select headings above this level.

Default: `none`

### select-group

Generate selector that selects all theorems of the same group as the argument.

#### Parameters

```
select-group(thm-func: theorem-function) -> selector
```

### select-kind

Generate selector that selects only theorems that were create from the `theorem-function`.

#### Parameters

```
select-kind(thm-func: theorem-function) -> selector
```

## Resetting counters

### reset-counter

Reset theorem group counter to zero. The result needs to be added to the document.

#### Parameters

```
reset-counter(thm-func: theorem-function) -> content
```

**thm-func**    theorem-function

The group is obtained from this argument.

### reset-counter-heading

Reset counter of theorem group on headings with at most the specified level.

#### Parameters

```
reset-counter-heading(  
  thm-func: theorem-function,  
  max-level: int none,  
  content: content  
) -> content
```

**thm-func**    theorem-function

The group is obtained from this argument.

**max-level**    int or none

Should be at least 1.