



made with  
**Animo**

numbering.typ

*A demo of the features explained on the page “Numbering”  
of the Presentation Author Guide*

[reproducible-reporting.github.io/animo/numbering/](https://reproducible-reporting.github.io/animo/numbering/)

# Numbering

*Slide numbers, subslide numbers and a progress bar*

`slide-number()`

`per-subslide: number and count`

`per-subslide: step and steps`

*numbered: false leaves this slide out of the count, so it has no footer.*

# The Slide Number

`slide-number()` and `slide-count()` are context functions that return integers.

**slide 1 of 5**

```
#context [slide #slide-number() of #slide-count()]
```

The title slide and the slide after it have `numbered: false`, so the counter skips both, and the third slide of the deck is slide 1. The footer in the corner shows the same number.

This slide has a single subslide, so the footer writes the slide number alone.

# The Subslide Number

`it.number` is the position of the subslide on screen, counting from one.

sub 1

sub 2

sub 3

sub 4

`it.number` is 1 and `it.count` is 4

`per-subslide(it => ..)` calls its callback once per subslide, and the browser shows the rendering of the subslide on screen.

The orange label in the footer counts the subslides of this slide.

# The Subslide Number

`it.number` is the position of the subslide on screen, counting from one.

`it.count` is how many subslides the slide has.

sub 1

sub 2

sub 3

sub 4

`it.number` is 2 and `it.count` is 4

*`per-subslide(it => ..)` calls its callback once per subslide, and the browser shows the rendering of the subslide on screen.*

`it.number` grows by one with every step, and `it.count` stays the same.

# The Subslide Number

`it.number` is the position of the subslide on screen, counting from one.

`it.count` is how many subslides the slide has.

The footer writes both only when `it.count > 1`.

sub 1

sub 2

sub 3

sub 4

`it.number` is 3 and `it.count` is 4

`per-subslide(it => ..)` calls its callback once per subslide, and the browser shows the rendering of the subslide on screen.

A question about “slide 2, sub 3” points at one subslide.

# The Subslide Number

`it.number` is the position of the subslide on screen, counting from one.

`it.count` is how many subslides the slide has.

The footer writes both only when `it.count > 1`.

Every output type shows the same number on the same subslide.

sub 1

sub 2

sub 3

sub 4

`it.number` is 4 and `it.count` is 4

`per-subslide(it => ..)` calls its callback once per subslide, and the browser shows the rendering of the subslide on screen.

*A page of the static presentation shows the number of its subslide.*

# A Progress Bar

## step 8 of 14



*The bar is  $100\% * (it.step - 1) / (it.steps - 1)$  wide, which is 54% here.*

*`it.step` counts the subslides of every slide, including the title slide.*

slide 3 of 5

sub 1 of 3

# A Progress Bar

## step 9 of 14



*The bar is  $100\% * (it.step - 1) / (it.steps - 1)$  wide, which is 62% here.*

**A slide that animates and a slide that does not move the bar alike.**

*Each step advances the bar by the same amount.*

# A Progress Bar

## step 10 of 14



*The bar is  $100\% * (it.step - 1) / (it.steps - 1)$  wide, which is 69% here.*

**A slide that animates and a slide that does not move the bar alike.**

**The same bar runs along the bottom edge of every slide, 3pt thin.**

*The bar on the slide and the thin one at the bottom edge are drawn by one function.*

## The `wrap:` Argument

Both frames hold `per-subslide(it => box(width: 100% * .., ..))`.

`wrap: block`



`wrap: auto`



`wrap: auto` chooses a box for inline content, and a box is only as wide as what it holds. A `100%` width inside such a box resolves to zero, so the lower bar is invisible. `wrap: block` gives the stack a block that fills the width of the slide.

The two frames hold the same inline bar, and only the upper one shows it.

# Where a Number Goes

*Typst lays out the overlay once per slide and the body once per epoch. A `per-subslide` in the overlay therefore costs its renderings only once, and the overlay stays in place when a `pan` moves the canvas.*

`[slide 5 in the body]`

*The footer is in the overlay, and the dashed label is a copy in the body.*

slide 5 of 5

sub 1 of 3

per epoch. A *per-subslide* in  
e overlay stays in place when a

The canvas has moved 10cm to the left, and the  
copy in the body has moved with it. The footer is  
still in its corner.

`[ slide 5 in the body ]`

`pan(dx: 10cm)` moves the canvas with the copy, and the footer stays in its corner.

# Where a Number Goes

*Typst lays out the overlay once per slide and the body once per epoch. A `per-subslide` in the overlay therefore costs its renderings only once, and the overlay stays in place when a `pan` moves the canvas.*

[ slide 5 in the body ]

`pan(x: 0cm)` returns to the origin of the canvas, and the copy returns with it.

slide 5 of 5

sub 3 of 3