



made with
Animo

regions.typ

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

reproducible-reporting.github.io/animo/regions/

Reflow Inside a Region

*Inside a **region**, typst lays out each epoch afresh.*

A short claim.

A caveat that goes away.

This paragraph comes after both tags.

Bare tags, each in a box for its largest epoch.

A short claim.

A caveat that goes away.

This paragraph comes after both tags.

The tinted frame shows the footprint of the region, which is the same in every state. The content after the region therefore never moves.

The same three paragraphs, inside a region on the left, and as bare tags on the right.

Reflow Inside a Region

*Inside a **region**, typst lays out each epoch afresh.*

A short claim.

This paragraph comes after both tags.

Bare tags, each in a box for its largest epoch.

A short claim.

This paragraph comes after both tags.

The tinted frame shows the footprint of the region, which is the same in every state. The content after the region therefore never moves.

remove frees the space of the caveat in the region, and the paragraph after it moves up.

Reflow Inside a Region

*Inside a **region**, typst lays out each epoch afresh.*

A claim that grew long enough to wrap onto a second line.

This paragraph comes after both tags.

Bare tags, each in a box for its largest epoch.

A claim that grew long enough to wrap onto a second line.

This paragraph comes after both tags.

The tinted frame shows the footprint of the region, which is the same in every state. The content after the region therefore never moves.

replace lays out a longer claim, which wraps and pushes the paragraph down again.

Where a Smaller State Sits

`align: top`, the default

`align: bottom`

This line comes right after both regions.

With `align: bottom`, the gap sits above a short state, where it reads as the spacing after the heading.

Both regions reserve the room of their tallest state, so this short state leaves a gap.

Where a Smaller State Sits

align: top, the default

This paragraph arrives in state 1. Both regions reserve its room from the start, so the line after them stays where it is.

align: bottom

This paragraph arrives in state 1. Both regions reserve its room from the start, so the line after them stays where it is.

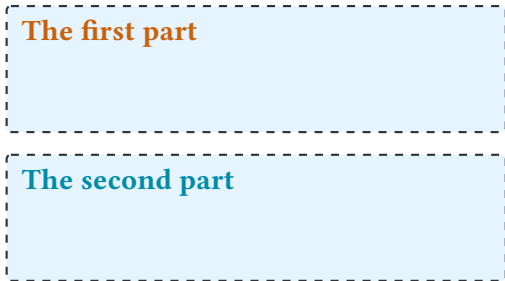
This line comes right after both regions.

*With **align: bottom**, the gap sits above a short state, where it reads as the spacing after the heading.*

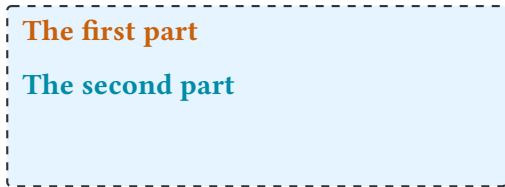
***reset** brings in the paragraph, which fills the footprint of both regions.*

Splitting and Merging Regions

Two regions, one per part.



One region around both parts.



Separate regions reserve the sum of their tallest states. One region around both parts reserves less room, but a change in one part moves the other.

On the left, each part has a region. On the right, one region holds both parts.

Splitting and Merging Regions

Two regions, one per part.

The first part

This paragraph arrives in state 1 and leaves again in state 2.

The second part

One region around both parts.

The first part

This paragraph arrives in state 1 and leaves again in state 2.

The second part

Separate regions reserve the sum of their tallest states. One region around both parts reserves less room, but a change in one part moves the other.

reset brings in the first paragraph. On the right, the second title moves down.

Splitting and Merging Regions

Two regions, one per part.

The first part

The second part

This paragraph arrives in state 2,
when the first one leaves.

One region around both parts.

The first part

The second part

This paragraph arrives in state 2,
when the first one leaves.

Separate regions reserve the sum of their tallest states. One region around both parts reserves less room, but a change in one part moves the other.

The paragraphs swap. On the left, each region changes without moving the other.

A Given Size

`region[...]` measures every epoch.



`region(height: 2cm)[...]` measures nothing.



A region of a given size compiles faster, because it skips the measurement of its epochs. It clips by default, so check every subslide for content that is cut off.

The measured region reserves room for the long text. The other region is 2cm tall.

A Given Size

`region[...]` measures every epoch.

This text arrives in state 1 and needs more room than 2cm. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam.

`region(height: 2cm)[...]` measures nothing.

This text arrives in state 1 and needs more room than 2cm. Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt

A region of a given size compiles faster, because it skips the measurement of its epochs. It clips by default, so check every subslide for content that is cut off.

The long text fits in the measured region, and is clipped in the region of a given size.

Nesting and Names

The outer region, named **box**

An inner region

The outer region continues here, and stays where it is.

The timeline can address a region with a name like a tag. Regions nest at most three deep.

*The outer region is named **box**, and holds an inner region.*

Nesting and Names

The outer region, named **box**

An inner region

A sentence that arrives inside the inner region, which reflows alone.

The outer region continues here, and stays where it is.

The timeline can address a region with a name like a tag. Regions nest at most three deep.

reset brings in a sentence, which reflows only the inner region.

Nesting and Names

The outer region, named box

An inner region

A sentence that arrives inside the inner region, which reflows alone.

The outer region continues here, and stays where it is.

The timeline can address a region with a name like a tag. Regions nest at most three deep.

`scale` shrinks the whole outer footprint, because the region has a name.

Nesting and Names

The timeline can address a region with a name like a tag. Regions nest at most three deep.

`move` shifts the footprint, and `hide` makes it invisible half a second later.

Regions in Containers

In a `1fr` grid column.

In `#columns(2)`.

In a placed box with `width: 5cm`.

In an `auto` grid column, a `stack(dir: ltr)` or a bare `#place`, a region takes the whole body width unless it is given a `width`.

Each region is as wide as its container, which is the width it lays out its body at.

Regions in Containers

In a `1fr` grid column.

This sentence arrives in state 1 and wraps at the width of its region.

In a placed box with `width: 5cm`.

This sentence arrives in state 1 and wraps at the width of its region.

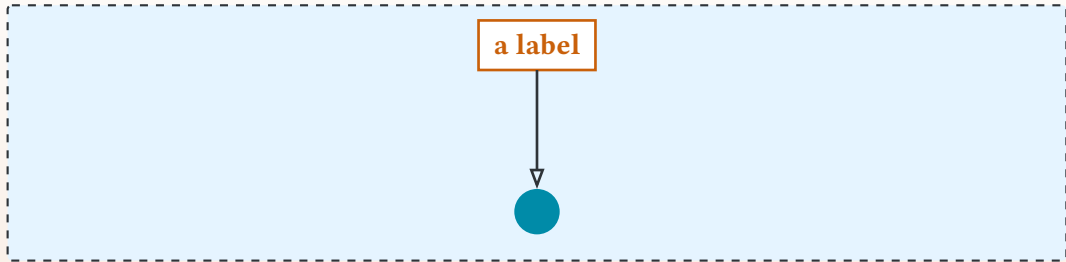
In `#columns(2)`.

This sentence arrives in state 1 and wraps at the width of its region.

In an `auto` grid column, a `stack(dir: ltr)` or a bare `#place`, a region takes the whole body width unless it is given a `width:`.

`reset` brings in the same sentence in all three regions, which wrap it at three widths.

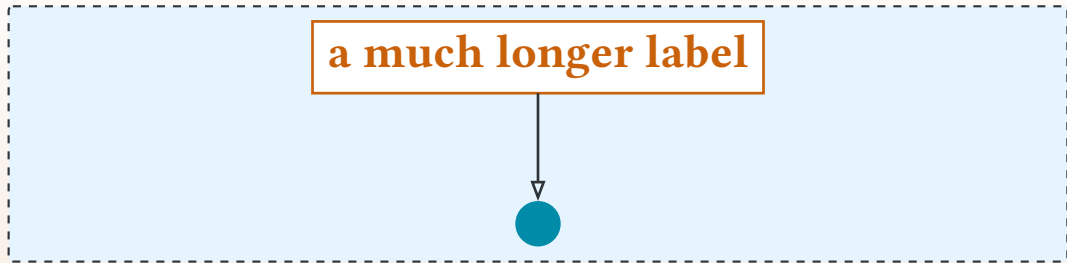
A Region Around a Cetz Canvas



The arrow stays attached to the frame of the label, because the whole figure is drawn again. A region around a canvas is the most expensive construct in a deck.

The label is a tag on a `content()` element of the cetz canvas.

A Region Around a Cetz Canvas



The arrow stays attached to the frame of the label, because the whole figure is drawn again. A region around a canvas is the most expensive construct in a deck.

replace lays out a larger label, and *typst* draws the whole figure again.