



made with
Animo

structural.typ

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

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

Replace, Apply and Reset

A short line.

This paragraph stays where it is on every state, because a tag whose content changes reserves the room of its largest epoch.

The body lays out a short line, in a tag that reserves room for two lines.

state 0 ●○○○

Replace, Apply and Reset

A short line.

This paragraph stays where it is on every state, because a tag whose content changes reserves the room of its largest epoch.

`apply` wraps the line in `text.with(fill: ..)`, which colors it.

Replace, Apply and Reset

This replacement is a good deal longer than the short line it replaces, so that it wraps onto a second line inside the room that the tag reserves.

This paragraph stays where it is on every state, because a tag whose content changes reserves the room of its largest epoch.

`replace` lays out a longer text, which arrives colored because the wrapper stays.

state 2 ○○○●

Replace, Apply and Reset

A short line.

This paragraph stays where it is on every state, because a tag whose content changes reserves the room of its largest epoch.

reset *lays out the body as written again, without the wrapper.*

state 3 ○○○●

Inserting Content

An empty tag receives  a replacement.

A word in the body can  with a **reset**.

*An empty tag in the body and **replace** in the timeline insert content. A tag whose first content operation is **reset** starts removed.*

The dashed frames mark the room that each tag reserves. The second tag starts removed.

Inserting Content

An empty tag receives **inserted by the timeline** a replacement.

A word in the body can `-----` with a **reset**.

*An empty tag in the body and **replace** in the timeline insert content. A tag whose first content operation is **reset** starts removed.*

replace on an empty tag inserts content that the body does not hold.

Inserting Content

An empty tag receives **inserted by the timeline** a replacement.

A word in the body can **arrive later** with a **reset**.

*An empty tag in the body and **replace** in the timeline insert content. A tag whose first content operation is **reset** starts removed.*

reset brings in the body of a tag that started removed.

How They Compose



laid out **B**
wrappers none

replace and *remove* set what is laid out and keep the wrappers, *apply* adds wrappers, and *reset* sets both.

The body lays out B, with no wrappers.

state 0 ● ○ ○ ○ ○ ○ ○ ○

How They Compose

B

laid out **B**
wrappers **emph**

replace and *remove* set what is laid out and keep the wrappers, *apply* adds wrappers, and *reset* sets both.

apply("x", emph) adds a wrapper around *B*.

state 1 ○●○○○○○○

How They Compose

R

laid out **R**

wrappers **emph**

`replace` and `remove` set what is laid out and keep the wrappers, `apply` adds wrappers, and `reset` sets both.

`replace("x")[R]` lays out *R*, and the wrapper stays.

state 2 ○●○○○○○

How They Compose



laid out **R**
wrappers **emph** inside **strong**

replace and **remove** set what is laid out and keep the wrappers, **apply** adds wrappers, and **reset** sets both.

apply("x", strong) adds a second wrapper, outside the first one.

state 3 ○○○●○○○

How They Compose



laid out **nothing**
wrappers **emph** inside **strong**

replace and **remove** set what is laid out and keep the wrappers, **apply** adds wrappers, and **reset** sets both.

remove("x") lays out nothing, and both wrappers are kept.

How They Compose



laid out **S**
wrappers **emph** inside **strong**

replace and **remove** set what is laid out and keep the wrappers, **apply** adds wrappers, and **reset** sets both.

replace("x")[S] lays out *S* inside the two kept wrappers.

How They Compose



laid out **B**
wrappers none

`replace` and `remove` set what is laid out and keep the wrappers, `apply` adds wrappers, and `reset` sets both.

`reset("x")` lays out the body again, without wrappers.

How They Compose



laid out **B**
wrappers none

`replace` and `remove` set what is laid out and keep the wrappers, `apply` adds wrappers, and `reset` sets both.

`apply("x", emph)`, `reset("x")` applies in order, so the reset wins.

state 7 ○○○○○○●

Several Sites and a Move

$$f(x) = x + 1$$

Both sites of `var` change together.

A short line

The name `var` tags both sites of x in the equation.

Several Sites and a Move

$$f(y) = y + 1$$

Both sites of `var` change together.

A short line

`replace` gives every site of `var` the same replacement.

Several Sites and a Move

$$f(y) = y + 1$$

Both sites of `var` change together.

A different line

A `replace` and a `move` in one subslide read as one movement.

state 2 ○○○●

Several Sites and a Move

$$f(y) = y + 1$$

Both sites of `var` change together.

A different line

`apply` recolors the line, which stays where the `move` put it.

state 3 ○○○●

Epochs

A line removed in the last epoch.



A new epoch begins at every subslide that holds a structural primitive.

The body as written is the first layout of the slide, which is epoch 0.

Epochs

A line revealed, then replaced and moved.

A line removed in the last epoch.



A new epoch begins at every subslide that holds a structural primitive.

reveal changes no content, so state 1 stays in epoch 0.

Epochs

This replacement starts epoch 1.

A line removed in the last epoch.



A new epoch begins at every subslide that holds a structural primitive.

`replace` starts epoch 1, so `typst` lays the slide out a second time.

Epochs

This replacement starts epoch 1.

A line removed in the last epoch.



A new epoch begins at every subslide that holds a structural primitive.

move changes no content, so state 3 shares epoch 1 with state 2.

Epochs

This replacement starts epoch 1.



A new epoch begins at every subslide that holds a structural primitive.

remove starts epoch 2, so typst lays this slide out three times in all.

Space That Is Reserved

This sentence contains a word and reserves space for it.

A short phrase on a line reserves the width of its longest epoch.

*The rest of the slide does not move, because a tag reserves room for its largest content.
Starting removed and starting hidden look the same inside a reserved box.*

The dashed frames mark the room each tag reserves. The word starts removed.

Space That Is Reserved

This sentence contains a **hidden** word and reserves space for it.

A **short** phrase on a line reserves the width of its longest epoch.

*The rest of the slide does not move, because a tag reserves room for its largest content.
Starting removed and starting hidden look the same inside a reserved box.*

reset *lays out the word in the room that was kept for it.*

Space That Is Reserved

This sentence contains a **hidden** word and reserves space for it.

A **a much longer phrase** phrase on a line reserves the width of its longest epoch.

*The rest of the slide does not move, because a tag reserves room for its largest content.
Starting removed and starting hidden look the same inside a reserved box.*

replace *lays out a longer phrase, as wide as the room that was kept for it.*

What a Structural Step Looks Like

The first answer.

a slow dissolve, duration: 2

The first answer.

a hard cut, duration: 0

Neither content travels to meet the other. The old one fades out where the new one fades in.

Each of the next two states replaces one answer, which the browser animates.

state 0 ●○○○

What a Structural Step Looks Like

The second answer.

a slow dissolve, duration: 2

The first answer.

a hard cut, duration: 0

Neither content travels to meet the other. The old one fades out where the new one fades in.

duration: 2 dissolves the first answer slowly into the second.

state 1 ○●○

What a Structural Step Looks Like

The second answer.

a slow dissolve, duration: 2

The second answer.

a hard cut, duration: 0

Neither content travels to meet the other. The old one fades out where the new one fades in.

duration: 0 swaps the content at once, which is a hard cut.