



made with
Animo

continuous.typ

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

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

Subslides

State 2 moves this line down by a centimeter.

A slide with S sub calls has $S + 1$ states, numbered from 0 to S .

Subslides

State 1 reveals this line, and state 2 hides it again.

State 2 moves this line down by a centimeter.

A slide with S sub calls has $S + 1$ states, numbered from 0 to S .

Subslides

State 2 moves this line down by a centimeter.

A slide with S sub calls has $S + 1$ states, numbered from 0 to S .

The Block-Scoped Import

In the body, `move` and `scale` are the elements of `typst`, which change the layout.

`typst`'s `move` lifts a word, and `typst`'s `scale` stretches one.

In the timeline, `scale` is the primitive of `Animo`, which scales a tag.

This phrase grows.

`typst`'s own `hide` hides the framed sentence, and no timeline can reveal it.



The Block-Scoped Import

In the body, `move` and `scale` are the elements of `typst`, which change the layout.

`typst`'s `move` lifts a word, and `typst`'s `scale` stretches one.

In the timeline, `scale` is the primitive of `Animo`, which scales a tag.

This phrase grows.

`typst`'s own `hide` hides the framed sentence, and no timeline can reveal it.



Without the Import

A star import of the package also brings in `anim`, so this timeline needs no `import`:

```
animation: anim.sub(anim.reveal("short"))
```

Without the Import

A star import of the package also brings in `anim`, so this timeline needs no `import`:

```
animation: anim.sub(anim.reveal("short"))
```

This line is the tag that the timeline reveals.

Move and Scale



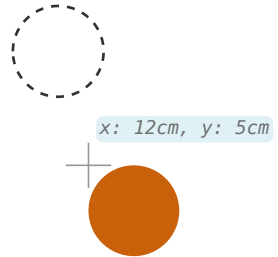
x: 12cm, y: 5cm



The body puts the ball at the top left, and the dashed target on the right.

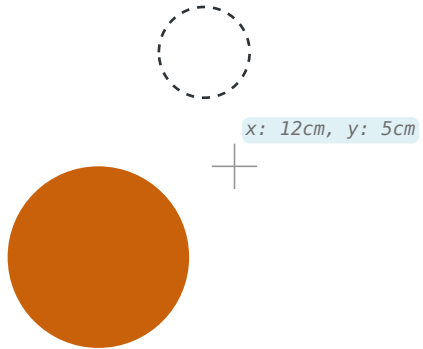
state 0 ● ○ ○ ○ ○

Move and Scale



`x: 12cm, y: 5cm` puts the top-left corner of the ball on the cross.

Move and Scale



$dx: -3\text{cm}$ shifts the ball to the left, and $f: 2$ doubles it about its centre.

Move and Scale



`x: 12cm, y: 5cm`



`relto: "target"` puts the ball on the target, and `f: 1` restores its size.

state 3 ○○○●○

Move and Scale



`x: 12cm, y: 5cm`



`hide` makes the target invisible, and the ball stays where it is.

state 4 ○○○○●

Where **move** Puts Things

label

first and **second**



*The label and the two sites of **pair** are where the body put them.*

state 0 ●○○○○

Where `move` Puts Things

`first` and `second`



`relto: "node"` puts the top-left corner of the label on that of the node.

Where **move** Puts Things

first and **second**



dy: -5mm lifts the label from where it is.

Where **move** Puts Things

first and **second**



label

x: 1cm, y: 6cm is measured from the corner of the slide, so x: 1cm is the margin.

state 3 ○○○●○

Where **move** Puts Things

and



label

*The first site of **pair** lands on the node, and the second site moves along.*

state 4 ○○○○●

Moved and Scaled



The square is 1cm wide, and the dashed target is 2cm wide.

Moved and Scaled



f: 2 and relto in one subslide put the corner of the doubled square on the target.

Moved and Scaled



$f: 1$ shrinks the placed square towards the corner it was put on.

What **scale** Sets

stretch

f sets both axes, **fx** and **fy** one axis each. A factor is a number or a ratio, so **f: 2** and **f: 200%** are the same.

The dashed frame marks the size of the word as the body laid it out.

What **scale** Sets

stretch

f sets both axes, **fx** and **fy** one axis each. A factor is a number or a ratio, so **f: 2** and **f: 200%** are the same.

fx: 2 makes the word twice as wide.

What **scale** Sets

stretch

f sets both axes, **fx** and **fy** one axis each. A factor is a number or a ratio, so **f: 2** and **f: 200%** are the same.

fy: 200% makes it twice as tall, and the width keeps its factor of 2.

What **scale** Sets

stretch

f sets both axes, **f_x** and **f_y** one axis each. A factor is a number or a ratio, so **$f: 2$** and **$f: 200\%$** are the same.

$f: 1$ sets both factors back to 1.

The next slide plays on its own.

It comes up in six seconds, so there is no need to press a key.

Timing

A presenter can always step ahead of a timer.

*Stepping back plays the deck backwards, and **Space** stops it where it is.*

***init(wait: 6)** brings this state up six seconds after the slide before it.*

Timing

wait: and **hold:** are the two names of one gap between two subslides.

A presenter can always step ahead of a timer.

*Stepping back plays the deck backwards, and **Space** stops it where it is.*

***wait: 2** brings this state up two seconds later, and **hold: 2** keeps it for two.*

Timing

wait: and **hold:** are the two names of one gap between two subslides.

delay: staggers the operations of one subslide,
so they arrive in the order they are written,
and **duration:** says how long each of them takes.

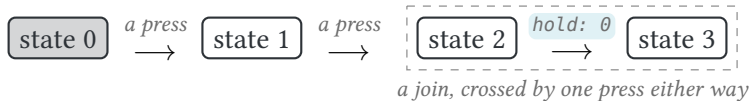
A presenter can always step ahead of a timer.

*Stepping back plays the deck backwards, and **Space** stops it where it is.*

delay: and **duration:** stagger the three lines, and a step back plays them in reverse.

Durations and Joins

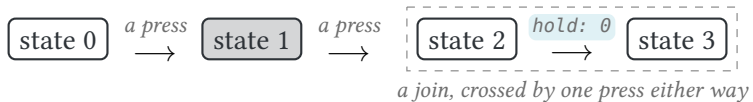
A slow move of four seconds



The next press starts a move that takes four seconds.

Durations and Joins

A slow move of four seconds



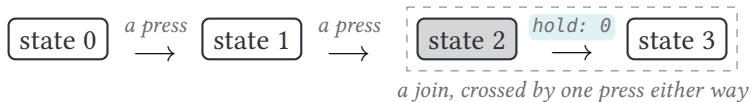
A press while the move is still running overtakes it rather than cutting it short.

state 1 ○●○○

Durations and Joins

A slow move of four seconds

State 2 comes up with a press,



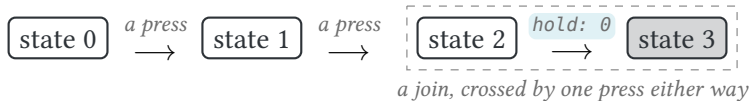
hold: 0 brings state 3 up at the same time as state 2, so the deck does not stop here.

state 2 ○○○●

Durations and Joins

A slow move of four seconds

State 2 comes up with a press, and state 3 comes up with it.



One press forward ran through the join, and one press back returns over it to state 1.

state 3 ○○○●