Version: 3.x

Layout transitions

> INFO
>
> This page was ported from an old version of the documentation.
>
> As we're rewriting the documentation some of the pages might be a little outdated.

The document explains how can you animate all layout changes for a specific view just by adding a single property to the view. To be precise how to animate positions and dimensions of components. What's important it will all happen entirely on UI thread without any communication through the bridge. There are plenty of ways in which you can animated layout changes however in contrast to entering and exiting animations they are not so regular. We've prepared a few predefined layout transitions, however if you want to create more custom and complex transition you can create your own.

How to use predefined layout transition?

1. Import chosen transition

```
// Transition is just an example and should be replaced by real animation. For Instance Layout
import { Transition } from 'react-native-reanimated';
```

2. Choose Animated Component which layout you want to animate

```
    // AnimatedComponent - component created by createAnimatedComponent or imported from Reanimated
    <AnimatedComponent layout={Transition} >
```

3. Customize the animation

Different type of layout transitions can be customized differently. For the complete list of option please refer to the paragraph specific to the particular animation type.

```
    <AnimatedComponent layout={Transition.duration(3000).otherModifier()} >
```

Predefined Transitions

Below we listed all of the currently available predefined layout transitions. Each transition contains all of its modifiers and a video presenting what it looks like when applied to a waterfall grid.

If you cannot find an transition that suits you then you can create a custom one. If you think that the animation should be here, please open an issue or create a pull request.

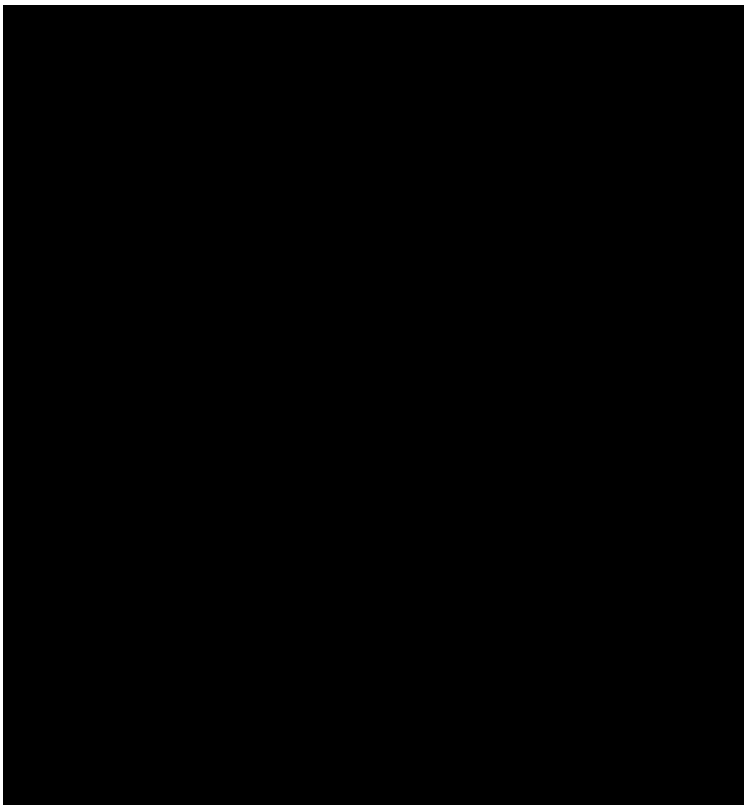
Layout

Linear transition, animates both position and dimension in the same way.

Modifiers

- `duration` (in ms) default: 300
- `delay` (in ms) default: 0
- `easing` same easing worklet as with `withTiming`
- `springify` change animation to spring
- `damping` default: 10
- `mass` default: 1
- `stiffness` default: 100
- `overshootClamping` default: false
- `restDisplacementThreshold` default: 0.001
- `restSpeedThreshold` default: 0.001
- `withCallback` callback that will fire after the exit animation ends
- `randomDelay` randomize delay of the animation between 0 and provided delay (or 1000ms if delay not provided)
- `reduceMotion` determines how the animation responds to the device's reduced motion accessibility setting.

Example



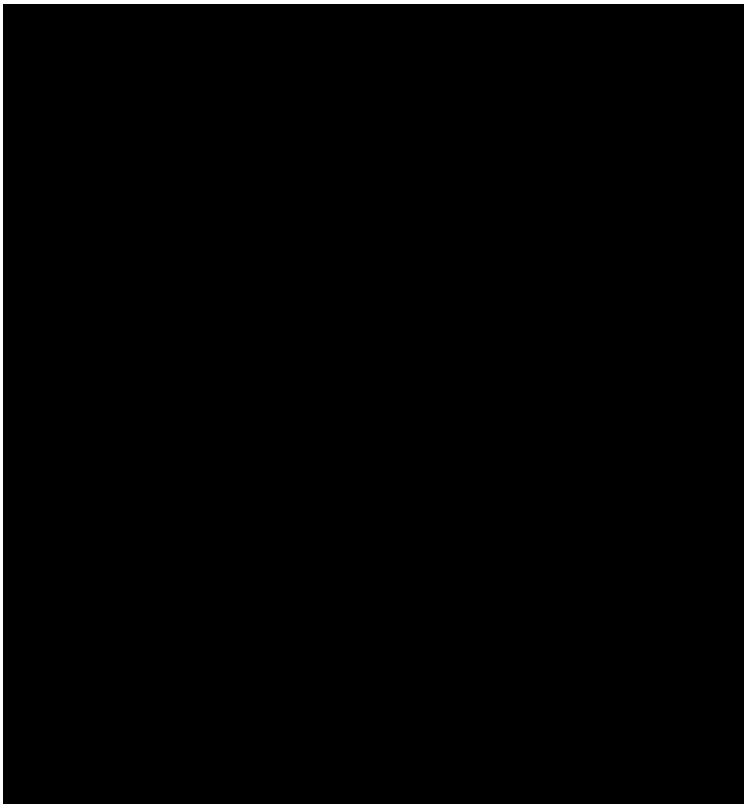
Sequenced Transition

Sequenced transition, animates firstly x-position and width, then later y-position and height.

Modifiers

- `duration` (in ms) default: 300
- `delay` (in ms) default: 0
- `withCallback` callback that will fire after the exit animation ends
- `randomDelay` randomize delay of the animation between 0 and provided delay (or 1000ms if delay not provided)
- `reverse` reverse order of the animation (first animates y-dimension and height)
- `reduceMotion` determines how the animation responds to the device's reduced motion accessibility setting.

Example



Fading Transition

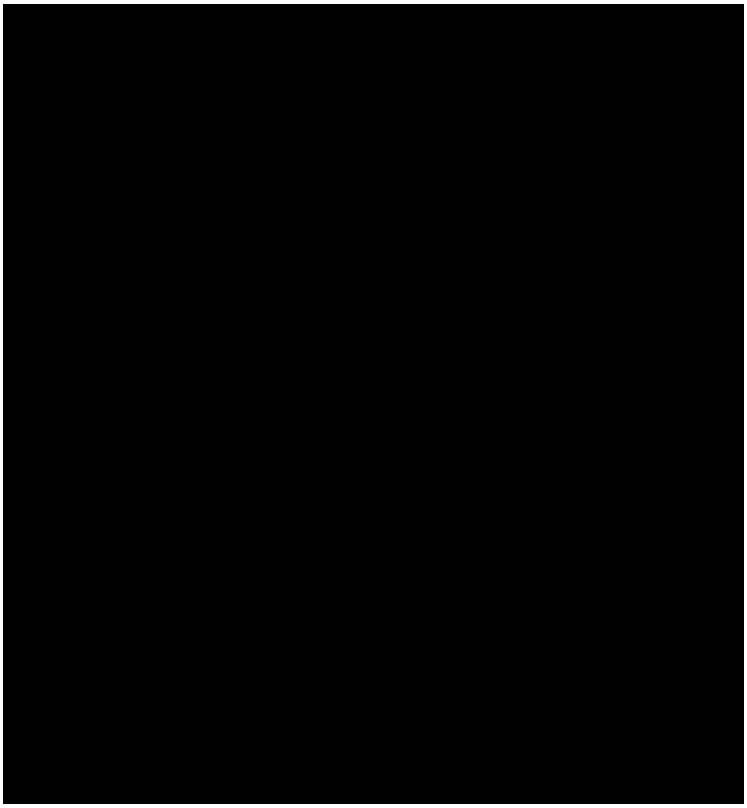
Fading transition, animates the opacity of component, so it will disappear with previous position and dimensions and appear with new ones.

Modifiers

- `duration` (in ms) default: 300
- `delay` (in ms) default: 0
- `withCallback` callback that will fire after the exit animation ends
- `randomDelay` randomize delay of the animation between 0 and provided delay (or 1000ms if delay not provided)
- `reduceMotion` determines how the animation responds to the device's reduced motion accessibility setting.

Example

{" "}



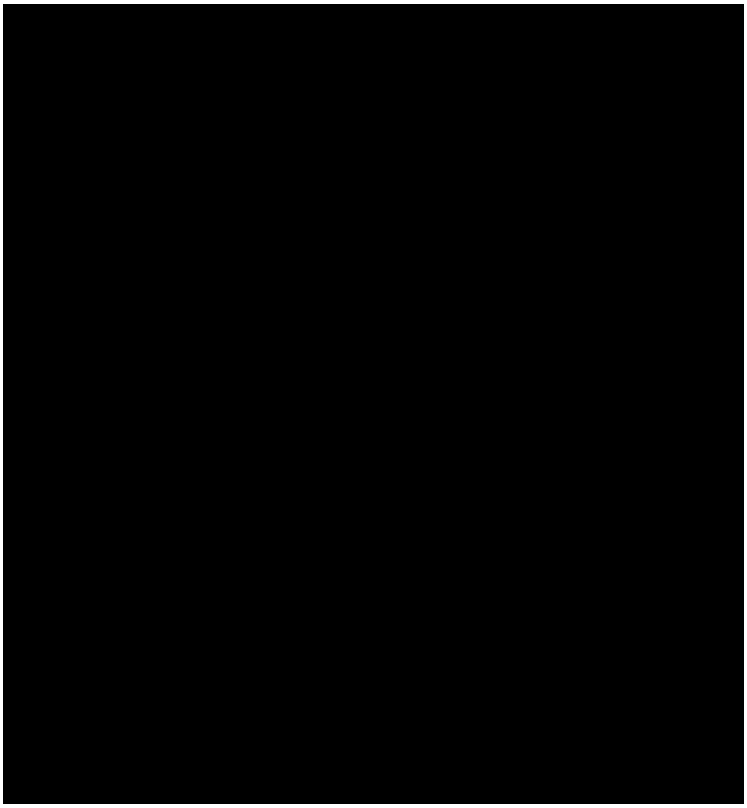
Jumping Transition

Jumping transition, component "jumps" to the new position.

Modifiers

- `duration` (in ms) default: 300
- `delay` (in ms) default: 0
- `withCallback` callback that will fire after the exit animation ends
- `randomDelay` randomize delay of the animation between 0 and provided delay (or 1000ms if delay not provided)
- `reduceMotion` determines how the animation responds to the device's reduced motion accessibility setting.

Example



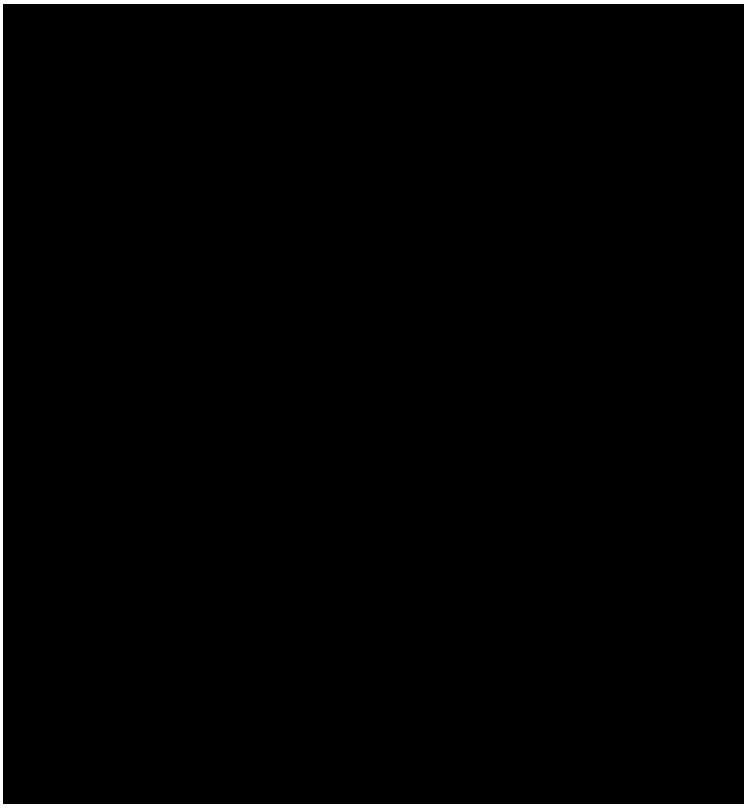
Curved Transition

Curved transition, enables to animate each position and dimension with different easing which makes components animation curved.

Modifiers

- `duration` (in ms) default: 300
- `delay` (in ms) default: 0
- `withCallback` callback that will fire after the exit animation ends
- `randomDelay` randomize delay of the animation between 0 and provided delay (or 1000ms if delay not provided)
- `easingX` provides Easing for x-position (default: `Easing.in(Easing.ease)`)
- `easingY` provides Easing for y-position (default: `Easing.out(Easing.ease)`)
- `easingWidth` provides Easing for width (default: `Easing.in(Easing.exp)`)
- `easingHeight` provides Easing for height (default: `Easing.out(Easing.exp)`)
- `reduceMotion` determines how the animation responds to the device's reduced motion accessibility setting.

Example



Entry/Exit Transition

Entry/Exit transition, lets you specify different animations for exiting from the current position and different animations for entering the new position with new dimensions. You can use all available predefined entering/entering animation or create your own one. Its duration equals the duration sum of entering and exiting. Also, be aware that you cannot use spring animations as entering or exiting as they don't have a fixed duration.

Modifiers

- `delay` (in ms) default: 0
- `withCallback` callback that will fire after the exit animation ends
- `randomDelay` randomize delay of the animation between 0 and provided delay (or 1000ms if delay not provided)
- `entering` animation that will be used for component entering (default: `FadeIn`)
- `exiting` animation that will be used for component exiting (default: `FadeOut`)
- `reduceMotion` determines how the animation responds to the device's reduced motion accessibility setting.

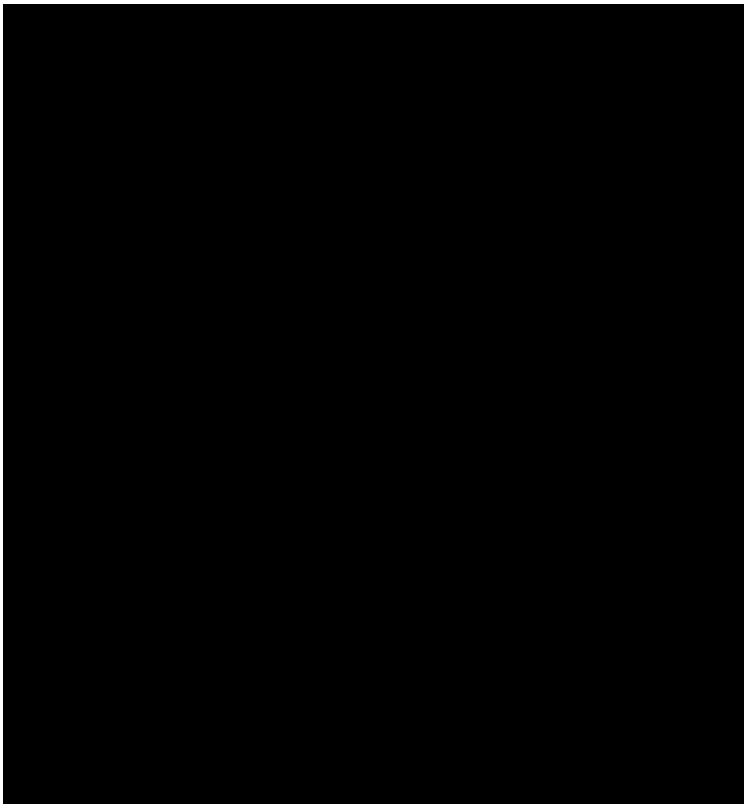
Combine Transition

To make usage of that transition simplier we have prepared function `combineTransition` that will make your code look cleaner and shorter.

Usage

```
// you can change ExitingAnimation and EnteringAnimation for any predefined animation you would like
// you can apply modifier (i.ex. delay()) on the object that this function returns
combineTransition(ExitingAnimation, EnteringAnimation).modifier();
```

Example



0:00

Edit this page