Version: 3.x

Entering/Exiting animations

Entering/Exiting animations let you animate elements when they are added to or removed from the view hierarchy.

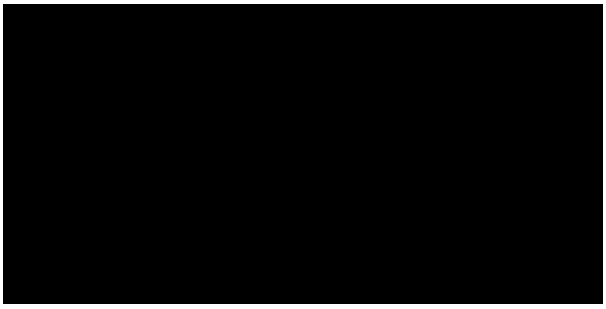
Reanimated comes with a bunch of predefined animations you can customize. For more advanced use-cases, you can use Keyframes or create your own custom entering/exiting animations.

Fade

`FadeX` lets you create a fading animation.

```
import { FadeIn, FadeOut } from 'react-native-reanimated';

function App() {
  return <Animated.View entering={FadeIn} exiting={FadeOut} />;
}
```



Available fade animations:

Entering

- `FadeIn`
- `FadeInRight`
- `FadeInLeft`

Exiting

- `FadeOut`
- `FadeOutRight`
- `FadeOutLeft`

- `FadeInUp`
- `FadeInDown`
- `FadeOutUp`
- `FadeOutDown`

Initial values

Modifiers

Time-based Optional

Time-based modifiers relay on `withTiming` function.

```
FadeOutLeft.duration(500).easing(Easing.ease);
```

- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `300`.
- `.easing(easingFunction: EasingFunction)` is an easing function which defines the animation curve. Defaults to `Easing.inOut(Easing.quad)`

NOTE

Time-based modifiers have no effect when `.springify()` is used.

Spring-based Optional

Spring-based modifiers relay on `withSpring` function.

```
FadeInUp.springify()
  .damping(30)
  .mass(5)
  .stiffness(10)
  .overshootClamping(false)
```

```
  .restDisplacementThreshold(0.1)
  .restSpeedThreshold(5);
```

- `.springify()` enables the spring-based animation configuration.
- `.damping(value: number)` decides how quickly a spring stops moving. Higher damping means the spring will come to rest faster. Defaults to `10`.
- `.mass(value: number)` is the weight of the spring. Reducing this value makes the animation faster. Defaults to `1`.
- `.stiffness(value: number)` decides how bouncy the spring is. Defaults to `100`.
- `.overshootClamping(value: boolean)` decides whether a spring can bounce over the designated position. Defaults to `false`.
- `.restDisplacementThreshold(value: number)` is the displacement below which the spring will snap to the designated position without further oscillations. Defaults to `0.001`.
- `.restSpeedThreshold(value: number)` is the speed in pixels per second from which the spring will snap to the designated position without further oscillations. Defaults to `2`.

Common Optional

```
FadeInDown.delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({ transform: [{ translateY: 420 }] })
  .withCallback((finished) => {
    console.log(`finished without interruptions: ${finished}`);
  });
```

- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.

- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

Bounce

`BounceX` lets you create a bouncing animation.

```
import { BounceIn, BounceOut } from 'react-native-reanimated';

function App() {
  return <Animated.View entering={BounceIn} exiting={BounceOut} />;
}
```

Available bounce animations:

Entering

- `BounceIn`
- `BounceInRight`
- `BounceInLeft`
- `BounceInUp`
- `BounceInDown`

Exiting

- `BounceOut`
- `BounceOutRight`
- `BounceOutLeft`
- `BounceOutUp`
- `BounceOutDown`

Initial values

Modifiers Optional

```
BounceInDown.duration(500)
  .delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({ transform: [{ translateY: -420 }] })
  .withCallback((finished) => {
    console.log(`finished without interruptions: ${finished}`);
  });
```

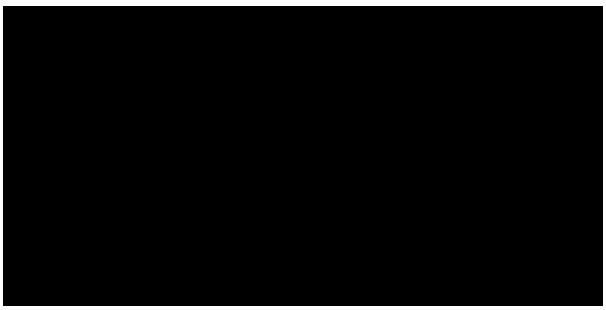
- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `600`.
- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.
- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.
- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

Flip

`FlipX` lets you create animation based on rotation over specific axis.

```
import { FlipInEasyX, FlipOutEasyX } from 'react-native-reanimated';

function App() {
  return <Animated.View entering={FlipInEasyX} exiting={FlipOutEasyX} />;
}
```



Available flip animations:

Entering

- `FlipInEasyX`
- `FlipInEasyY`
- `FlipInXDown`
- `FlipInXUp`
- `FlipInYLeft`
- `FlipInYRight`

Exiting

- `FlipOutEasyX`
- `FlipOutEasyY`
- `FlipOutXDown`
- `FlipOutXUp`
- `FlipOutYLeft`
- `FlipOutYRight`

Initial values

Modifiers

Time-based Optional

Time-based modifiers relay on `withTiming` function.

```
FlipOutYLeft.duration(500).easing(Easing.ease);
```

- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `300`.
- `.easing(easingFunction: EasingFunction)` is an easing function which defines the animation curve. Defaults to `Easing.inOut(Easing.quad)`

> **NOTE**
>
> Time-based modifiers have no effect when `.springify()` is used.

Spring-based Optional

Spring-based modifiers relay on `withSpring` function.

```
FlipInXUp.springify()
  .damping(2)
  .mass(3)
  .stiffness(10)
  .overshootClamping(false)
  .restDisplacementThreshold(0.1)
  .restSpeedThreshold(5);
```

- `.springify()` enables the spring-based animation configuration.
- `.damping(value: number)` decides how quickly a spring stops moving. Higher damping means the spring will come to rest faster. Defaults to `10`.
- `.mass(value: number)` is the weight of the spring. Reducing this value makes the animation faster. Defaults to `1`.
- `.stiffness(value: number)` decides how bouncy the spring is. Defaults to `100`.
- `.overshootClamping(value: boolean)` decides whether a spring can bounce over the designated position. Defaults to `false`.
- `.restDisplacementThreshold(value: number)` is the displacement below which the spring will snap to the designated position without further oscillations. Defaults to `0.001`.
- `.restSpeedThreshold(value: number)` is the speed in pixels per second from which the spring will snap to the designated position without further oscillations. Defaults to `2`.

Common Optional

```
FlipInEasyY.delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({
```

```
    transform: [{ perspective: 100 }, { rotateY: '123deg' }],
  })
  .withCallback((finished) => {
    console.log(`finished without interruptions: ${finished}`);
  });
```

- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.
- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.
- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

LightSpeed

`LightSpeedX` lets you create an animation of a horizontally moving object with a change of opacity and skew.

```
import { LightSpeedInRight, LightSpeedOutLeft } from 'react-native-reanimated';

function App() {
  return (
    <Animated.View entering={LightSpeedInRight} exiting={LightSpeedOutLeft} />
  );
}
```

Available lightspeed animations:

Entering

- `LightSpeedInRight`
- `LightSpeedInLeft`

Exiting

- `LightSpeedOutRight`
- `LightSpeedOutLeft`

Initial values

Modifiers

Time-based Optional

Time-based modifiers relay on `withTiming` function.

```
LightSpeedOutLeft.duration(500).easing(Easing.ease);
```

- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `300`.
- `.easing(easingFunction: EasingFunction)` is an easing function which defines the animation curve. Defaults to `Easing.inOut(Easing.quad)`

NOTE

Time-based modifiers have no effect when `.springify()` is used.

Spring-based Optional

Spring-based modifiers relay on `withSpring` function.

```
LightSpeedInLeft.springify()
  .damping(30)
  .mass(5)
  .stiffness(10)
  .overshootClamping(false)
  .restDisplacementThreshold(0.1)
  .restSpeedThreshold(5);
```

- `.springify()` enables the spring-based animation configuration.
- `.damping(value: number)` decides how quickly a spring stops moving. Higher damping means the spring will come to rest faster. Defaults to `10`.
- `.mass(value: number)` is the weight of the spring. Reducing this value makes the animation faster. Defaults to `1`.
- `.stiffness(value: number)` decides how bouncy the spring is. Defaults to `100`.
- `.overshootClamping(value: boolean)` decides whether a spring can bounce over the designated position. Defaults to `false`.
- `.restDisplacementThreshold(value: number)` is the displacement below which the spring will snap to the designated position without further oscillations. Defaults to `0.001`.
- `.restSpeedThreshold(value: number)` is the speed in pixels per second from which the spring will snap to the designated position without further oscillations. Defaults to `2`.

Common Optional

```
LightSpeedInRight.delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({
    transform: [{ translateX: -100 }, { skewX: '-10deg' }],
  })
  .withCallback((finished) => {
    console.log(`finished without interruptions: ${finished}`);
  });
```

- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.
- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.
- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

Pinwheel

`PinwheelX` lets you create an animation based on rotation, scale, and opacity.

```
import { PinwheelIn, PinwheelOut } from 'react-native-reanimated';

function App() {
  return <Animated.View entering={PinwheelIn} exiting={PinwheelOut} />;
}
```

Available pinwheel animations:

Entering

- `PinwheelIn`

Exiting

- `PinwheelOut`

Initial values

Modifiers

Time-based Optional

Time-based modifiers relay on `withTiming` function.

```
PinwheelOut.duration(500).easing(Easing.ease);
```

- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `300`.
- `.easing(easingFunction: EasingFunction)` is an easing function which defines the animation curve. Defaults to `Easing.inOut(Easing.quad)`

> **NOTE**
>
> Time-based modifiers have no effect when `.springify()` is used.

Spring-based Optional

Spring-based modifiers relay on `withSpring` function.

```
PinwheelIn.springify()
  .damping(2)
  .mass(5)
  .stiffness(10)
  .overshootClamping(false)
  .restDisplacementThreshold(0.1)
  .restSpeedThreshold(5);
```

- `.springify()` enables the spring-based animation configuration.

- `.damping(value: number)` decides how quickly a spring stops moving. Higher damping means the spring will come to rest faster. Defaults to `10`.
- `.mass(value: number)` is the weight of the spring. Reducing this value makes the animation faster. Defaults to `1`.
- `.stiffness(value: number)` decides how bouncy the spring is. Defaults to `100`.
- `.overshootClamping(value: boolean)` decides whether a spring can bounce over the designated position. Defaults to `false`.
- `.restDisplacementThreshold(value: number)` is the displacement below which the spring will snap to the designated position without further oscillations. Defaults to `0.001`.
- `.restSpeedThreshold(value: number)` is the speed in pixels per second from which the spring will snap to the designated position without further oscillations. Defaults to `2`.

Common Optional

```
PinwheelIn.delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({
    transform: [{ scale: 0.8 }, { rotate: '3' }],
  })
  .withCallback((finished) => {
    console.log(`finished without interruptions: ${finished}`);
  });
```

- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.
- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.
- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

Roll

`RollX` lets you create an animation of a horizontally moving object with a rotation.

```
import { RollInRight, RollOutLeft } from 'react-native-reanimated';

function App() {
  return <Animated.View entering={RollInRight} exiting={RollOutLeft} />;
}
```

Available roll animations:

Entering

- `RollInRight`
- `RollInLeft`

Exiting

- `RollOutRight`
- `RollOutLeft`

Initial values

Modifiers

Time-based Optional

Time-based modifiers relay on `withTiming` function.

```
RollOutLeft.duration(500).easing(Easing.ease);
```

- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `300`.
- `.easing(easingFunction: EasingFunction)` is an easing function which defines the animation curve. Defaults to `Easing.inOut(Easing.quad)`

> **NOTE**
>
> Time-based modifiers have no effect when `.springify()` is used.

Spring-based Optional

Spring-based modifiers relay on `withSpring` function.

```
RollInLeft.springify()
  .damping(30)
  .mass(5)
  .stiffness(10)
  .overshootClamping(false)
  .restDisplacementThreshold(0.1)
  .restSpeedThreshold(5);
```

- `.springify()` enables the spring-based animation configuration.
- `.damping(value: number)` decides how quickly a spring stops moving. Higher damping means the spring will come to rest faster. Defaults to `10`.
- `.mass(value: number)` is the weight of the spring. Reducing this value makes the animation faster. Defaults to `1`.
- `.stiffness(value: number)` decides how bouncy the spring is. Defaults to `100`.
- `.overshootClamping(value: boolean)` decides whether a spring can bounce over the designated position. Defaults to `false`.
- `.restDisplacementThreshold(value: number)` is the displacement below which the spring will snap to the designated position without further oscillations. Defaults to `0.001`.
- `.restSpeedThreshold(value: number)` is the speed in pixels per second from which the spring will snap to the designated position without further oscillations. Defaults to `2`.

Common Optional

```
RollInRight.delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({ transform: [{ translateX: 100 }, { rotate: '-45deg' }] })
  .withCallback((finished) => {
    console.log(`finished without interruptions: ${finished}`);
  });
```

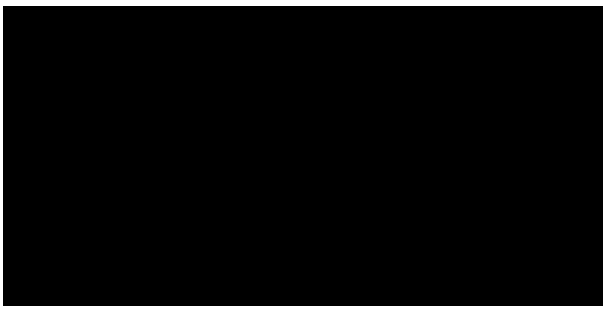
- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.
- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.
- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

Rotate

`RotateX` lets you create a rotation animation.

```
import { RotateInDownLeft, RotateOutDownLeft } from 'react-native-reanimated';

function App() {
  return (
    <Animated.View entering={RotateInDownLeft} exiting={RotateOutDownLeft} />
  );
}
```



Available rotate animations:

Entering

- `RotateInDownLeft`
- `RotateInDownRight`
- `RotateInUpLeft`
- `RotateInUpRight`

Exiting

- `RotateOutDownLeft`
- `RotateOutDownRight`
- `RotateOutUpLeft`
- `RotateOutUpRight`

Initial values

Modifiers

Time-based Optional

Time-based modifiers relay on `withTiming` function.

```
RotateOutDownRight.duration(500).easing(Easing.ease);
```

- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `300`.
- `.easing(easingFunction: EasingFunction)` is an easing function which defines the animation curve. Defaults to `Easing.inOut(Easing.quad)`

NOTE

> Time-based modifiers have no effect when `.springify()` is used.

Spring-based Optional

Spring-based modifiers relay on `withSpring` function.

```
RotateInUpLeft.springify()
  .damping(30)
  .mass(5)
  .stiffness(10)
  .overshootClamping(false)
  .restDisplacementThreshold(0.1)
  .restSpeedThreshold(5);
```

- `.springify()` enables the spring-based animation configuration.
- `.damping(value: number)` decides how quickly a spring stops moving. Higher damping means the spring will come to rest faster. Defaults to `10`.
- `.mass(value: number)` is the weight of the spring. Reducing this value makes the animation faster. Defaults to `1`.
- `.stiffness(value: number)` decides how bouncy the spring is. Defaults to `100`.
- `.overshootClamping(value: boolean)` decides whether a spring can bounce over the designated position. Defaults to `false`.
- `.restDisplacementThreshold(value: number)` is the displacement below which the spring will snap to the designated position without further oscillations. Defaults to `0.001`.
- `.restSpeedThreshold(value: number)` is the speed in pixels per second from which the spring will snap to the designated position without further oscillations. Defaults to `2`.

Common Optional

```
RotateInDownLeft.delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({
    transform: [{ rotate: '-90deg' }, { translateX: 100 }, { translateY: 100 }],
  })
  .withCallback((finished) => {
```

```
    console.log(`finished without interruptions: ${finished}`);
  });
```

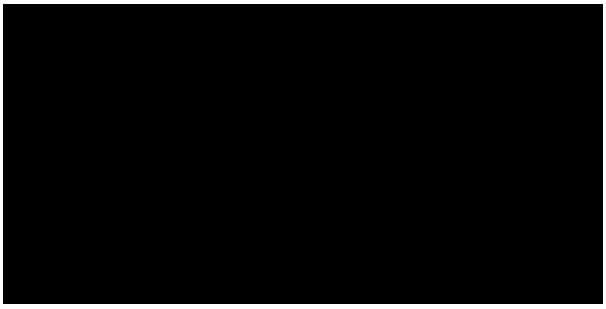
- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.
- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.
- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

Slide

`SlideX` lets you create an animation of horizontal or vertical moving object.

```
import { SlideInRight, SlideOutLeft } from 'react-native-reanimated';

function App() {
  return <Animated.View entering={SlideInRight} exiting={SlideOutLeft} />;
}
```



Available slide animations:

Entering

Exiting

- `SlideInRight`
- `SlideInLeft`
- `SlideInUp`
- `SlideInDown`
- `SlideOutRight`
- `SlideOutLeft`
- `SlideOutUp`
- `SlideOutDown`

Initial values

Modifiers

Time-based Optional

Time-based modifiers relay on `withTiming` function.

```
SlideOutLeft.duration(500).easing(Easing.ease);
```

- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `300`.
- `.easing(easingFunction: EasingFunction)` is an easing function which defines the animation curve. Defaults to `Easing.inOut(Easing.quad)`

NOTE

Time-based modifiers have no effect when `.springify()` is used.

Spring-based Optional

Spring-based modifiers relay on `withSpring` function.

```
SlideInUp.springify()
  .damping(30)
```

```
  .mass(5)
  .stiffness(10)
  .overshootClamping(false)
  .restDisplacementThreshold(0.1)
  .restSpeedThreshold(5);
```

- `.springify()` enables the spring-based animation configuration.
- `.damping(value: number)` decides how quickly a spring stops moving. Higher damping means the spring will come to rest faster. Defaults to `10`.
- `.mass(value: number)` is the weight of the spring. Reducing this value makes the animation faster. Defaults to `1`.
- `.stiffness(value: number)` decides how bouncy the spring is. Defaults to `100`.
- `.overshootClamping(value: boolean)` decides whether a spring can bounce over the designated position. Defaults to `false`.
- `.restDisplacementThreshold(value: number)` is the displacement below which the spring will snap to the designated position without further oscillations. Defaults to `0.001`.
- `.restSpeedThreshold(value: number)` is the speed in pixels per second from which the spring will snap to the designated position without further oscillations. Defaults to `2`.

Common Optional

```
SlideInDown.delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({ transform: [{ translateY: 420 }] })
  .withCallback((finished) => {
    console.log(`finished without interruptions: ${finished}`);
  });
```

- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.

- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.
- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

Stretch

`StretchX` lets you create an animation based on scaling in X or Y axis.

```
import { StretchInX, StretchOutY } from 'react-native-reanimated';

function App() {
  return <Animated.View entering={StretchInX} exiting={StretchOutY} />;
}
```

Available stretch animations:

Entering

- `StretchInX`
- `StretchInY`

Exiting

- `StretchOutX`
- `StretchOutY`

Initial values

Modifiers

Time-based Optional

Time-based modifiers relay on `withTiming` function.

```
StretchOutX.duration(500).easing(Easing.ease);
```

- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `300`.
- `.easing(easingFunction: EasingFunction)` is an easing function which defines the animation curve. Defaults to `Easing.inOut(Easing.quad)`

> **NOTE**
>
> Time-based modifiers have no effect when `.springify()` is used.

Spring-based Optional

Spring-based modifiers relay on `withSpring` function.

```
StretchInX.springify()
  .damping(30)
  .mass(5)
  .stiffness(10)
  .overshootClamping(false)
  .restDisplacementThreshold(0.1)
  .restSpeedThreshold(5);
```

- `.springify()` enables the spring-based animation configuration.
- `.damping(value: number)` decides how quickly a spring stops moving. Higher damping means the spring will come to rest faster. Defaults to `10`.
- `.mass(value: number)` is the weight of the spring. Reducing this value makes the animation faster. Defaults to `1`.
- `.stiffness(value: number)` decides how bouncy the spring is. Defaults to `100`.
- `.overshootClamping(value: boolean)` decides whether a spring can bounce over the designated position. Defaults to `false`.

- `.restDisplacementThreshold(value: number)` is the displacement below which the spring will snap to the designated position without further oscillations. Defaults to `0.001`.
- `.restSpeedThreshold(value: number)` is the speed in pixels per second from which the spring will snap to the designated position without further oscillations. Defaults to `2`.

Common Optional

```
StretchInY.delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({ transform: [{ scaleY: 0.5 }] })
  .withCallback((finished) => {
    console.log(`finished without interruptions: ${finished}`);
  });
```

- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.
- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.
- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

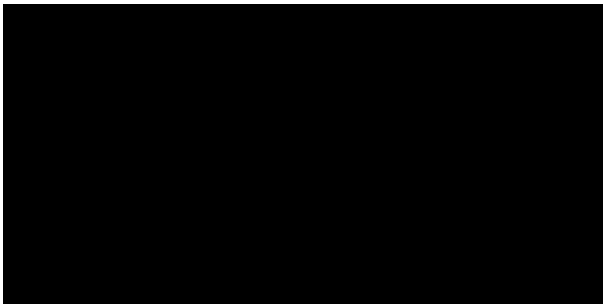
Zoom

`ZoomX` lets you create an animation based on scale.

```
import { ZoomIn, ZoomOut } from 'react-native-reanimated';

function App() {
```

```
  return <Animated.View entering={ZoomIn} exiting={ZoomOut} />;
}
```



Available zoom animations:

Entering

- `ZoomIn`
- `ZoomInDown`
- `ZoomInEasyDown`
- `ZoomInEasyUp`
- `ZoomInLeft`
- `ZoomInRight`
- `ZoomInRotate`
- `ZoomInUp`

Exiting

- `ZoomOut`
- `ZoomOutDown`
- `ZoomOutEasyDown`
- `ZoomOutEasyUp`
- `ZoomOutLeft`
- `ZoomOutRight`
- `ZoomOutRotate`
- `ZoomOutUp`

Initial values

Modifiers

Time-based Optional

Time-based modifiers relay on `withTiming` function.

```
ZoomOutLeft.duration(500).easing(Easing.ease);
```

- `.duration(durationMs: number)` is the length of the animation (in milliseconds). Defaults to `300`.
- `.easing(easingFunction: EasingFunction)` is an easing function which defines the animation curve. Defaults to `Easing.inOut(Easing.quad)`

> **NOTE**
>
> Time-based modifiers have no effect when `.springify()` is used.

Spring-based Optional

Spring-based modifiers relay on `withSpring` function.

```
ZoomInRotate.springify()
  .damping(30)
  .mass(5)
  .stiffness(10)
  .overshootClamping(false)
  .restDisplacementThreshold(0.1)
  .restSpeedThreshold(0.1);
```

- `.springify()` enables the spring-based animation configuration.
- `.damping(value: number)` decides how quickly a spring stops moving. Higher damping means the spring will come to rest faster. Defaults to `10`.
- `.mass(value: number)` is the weight of the spring. Reducing this value makes the animation faster. Defaults to `1`.
- `.stiffness(value: number)` decides how bouncy the spring is. Defaults to `100`.
- `.overshootClamping(value: boolean)` decides whether a spring can bounce over the designated position. Defaults to `false`.
- `.restDisplacementThreshold(value: number)` is the displacement below which the spring will snap to the designated position without further oscillations. Defaults to `0.001`.

- `.restSpeedThreshold(value: number)` is the speed in pixels per second from which the spring will snap to the designated position without further oscillations. Defaults to `2`.

Common Optional

```
ZoomIn.delay(500)
  .randomDelay()
  .reduceMotion(ReduceMotion.Never)
  .withInitialValues({ transform: [{ scale: 0.5 }] })
  .withCallback((finished) => {
    console.log(`finished without interruptions: ${finished}`);
  });
```

- `.delay(durationMs: number)` is the delay before the animation starts (in milliseconds). Defaults to `0`.
- `.randomDelay()` randomizes the delay of the animation between `0` and the provided delay. Uses 1000 ms if delay wasn't provided.
- `.reduceMotion(reduceMotion: ReduceMotion)` determines how the animation responds to the device's reduced motion accessibility setting.
- `.withInitialValues(values: StyleProps)` allows to override the initial config of the animation.
- `.withCallback(callback: (finished: boolean) => void)` is the callback that will fire after the animation ends. Sets `finished` to `true` when animation ends without interruptions, and `false` otherwise.

Platform compatibility

Android	iOS	Web
✅	✅	❌

Edit this page