Version: 3.x

withTiming

`withTiming` lets you create an animation based on duration and easing.

Preview Code

Reference

```
import { withTiming } from 'react-native-reanimated';

function App() {
  sv.value = withTiming(0);
  // ...
}
```

Type definitions

Arguments

`toValue`

The value on which the animation will come at rest.

`config` Optional

The timing animation configuration.


```
withTiming(sv.value, {
  duration: 1000,
  easing: Easing.inOut(Easing.quad),
  reduceMotion: ReduceMotion.System,
```

```
    })
```

Available properties:

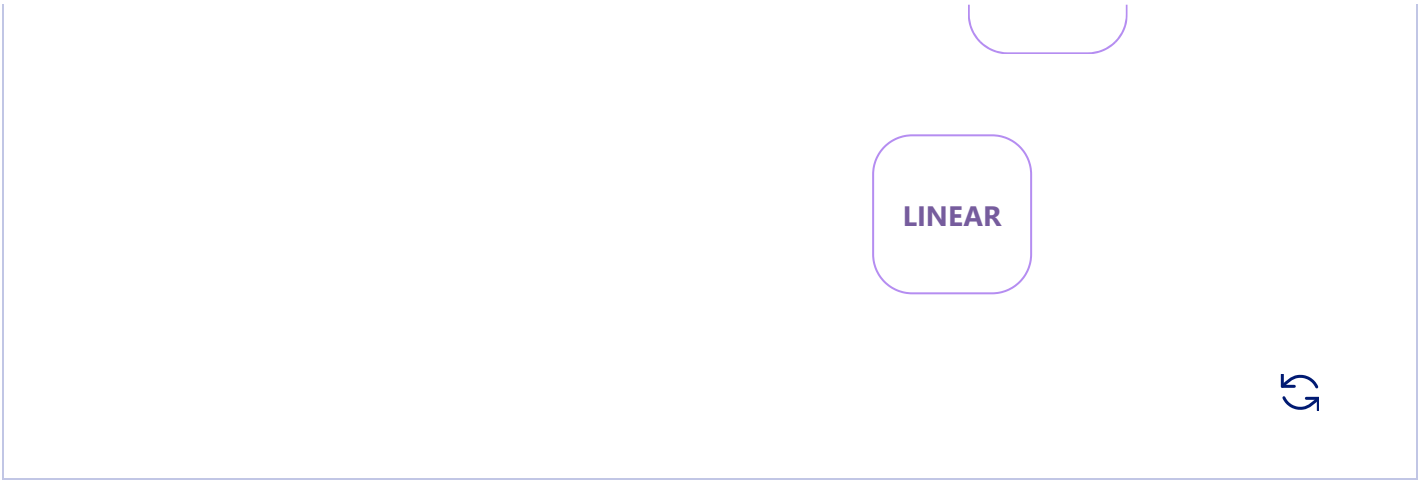
Name	Type	Default	Description
duration Optional	`number`	300	Length of the animation (in milliseconds).
easing Optional	`Easing`	`Easing.inOut(Easing.quad)`	An easing function which defines the animation curve.
reduceMotion Optional	`ReduceMotion`	`ReduceMotion.System`	A parameter that determines how the animation responds to the device's reduced motion accessibility setting.

`Easing`

The `easing` parameter lets you fine-tune the animation over the specified time duration. For example, you can make the animation begin with fast acceleration and then slow down towards the end, or start slowly, then pick up speed before slowing down again towards the end.

It will all start to make sense when you compare a `linear` easing side by side with the default `Easing.inOut(Easing.quad)` easing.

Preview Code



Reanimated provides a selection of ready-to-use easing functions in the `Easing` module. You can find a visualization of some common easing functions at http://easings.net/.

You can use our built-in easings by passing them as the `easing` property to the `withTiming` config:

```
import { Easing } from 'react-native-reanimated';

withTiming(sv.value, {
  easing: Easing.bounce,
});
```

Available functions:

- `back` provides a simple animation where the object goes slightly back before moving forward
- `bezier(x1: number, y1: number, x2: number, y2: number)` provides a cubic bezier curve
- `bounce` provides a bouncing animation
- `circle` provides a circular function
- `cubic` provides a cubic function
- `ease` provides a simple inertial animation
- `elastic(bounciness?: number)` provides a simple spring interaction
- `exp` provides an exponential function
- `linear` provides a linear function
- `poly(n: number)` can be used to implement quartic, quintic, and other higher power functions

- `quad` provides a quadratic function
- `sin` provides a sinusoidal function

The following helpers are used to modify other easing functions.

- `in(easing: EasingFunction)` runs an easing function forwards
- `inOut(easing: EasingFunction)` makes any easing function symmetrical
- `out(easing: EasingFunction)` runs an easing function backwards

`callback` Optional

A function called upon animation completion. If the animation is cancelled, the callback will receive `false` as the argument; otherwise, it will receive `true`.

Returns

`withTiming` returns an animation object. It can be either assigned directly to a shared value or can be used as a value for a style object returned from useAnimatedStyle.

Example

Preview Code

Remarks

- The callback passed to the 3rd argument is automatically workletized and ran on the UI thread.

Platform compatibility

Android	iOS	Web
✅	✅	✅

Edit this page