Version: 3.x

Keyframe animations

> ⓘ 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 As we're rewriting the documentation some of the pages might be a little outdated.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 you can define complex animation using simple and popular animation definitions schema - Keyframes.

How to define Keyframe animation?

1. Import Keyframe

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

2. Create Keyframe object, define initial and final state

In Keyframe's constructor pass object with definitions of your animation. Object keys should be within range `0-100` and correspond to animation progress, so to `0` assign the style, you want for your object at the beginning of the animation and to `100` assign the style you want for your object to have at the end of the animation.

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

const keyframe = new Keyframe({
  0: {
    transform: [{ rotate: '0deg' }],
  },
  100: {
    transform: [{ rotate: '45deg' }],
  },
});
```

Instead of using `0` and `100`, you can define edge points using `from` and `to` keywords. The result will be the same.

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

const keyframe = new Keyframe({
  from: {
    transform: [{ rotate: '0deg' }],
  },
  to: {
    transform: [{ rotate: '45deg' }],
  },
});
```

Providing keyframe `0` or `from` is required as it contains the initial state of the object you want to animate. Make sure you provided the initial value for all style properties you want to animate in other keyframes. Remember not to provide both `0` and `from`, or `100` and `to` keyframe as it will result in parsing conflict.

3. Add middle points

Between edge points, you can define middle points in which you want your object to have certain style properties. Remember that you can specify style only for those properties that you set the initial value in `0` or `from` keyframe. If you want to animate transform style, make sure that all properties in the transformation array are in the same order in all keyframes.

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

const keyframe = new Keyframe({
  0: {
    transform: [{ rotate: '0deg' }],
  },
  45: {
    transform: [{ rotate: '100deg' }],
  },
  100: {
    transform: [{ rotate: '45deg' }],
  },
});
```

4. Customize transitions using an easing function

If easing property is not provided, it defaults to linear easing function.

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

const keyframe = new Keyframe({
  0: {
    transform: [{ rotate: '0deg' }],
  },
  45: {
    transform: [{ rotate: '100deg' }],
    easing: Easing.exp,
  },
  100: {
    transform: [{ rotate: '45deg' }],
  },
});
```

How to use keyframe animations?

Currently, you can define animations using keyframes only for entry and exit animations.

1. Choose Animated Component which entering or exiting you want to animate

```
// AnimatedComponent - component created by createAnimatedComponent or imported from
Reanimated
// keyframe - Keyframe object
<AnimatedComponent exiting={keyframe} />
```

2. Customize the animation

```
<AnimatedComponent exiting={keyframe.duration(3000).delay(200)} />
```

Available modifiers

The order of modifiers doesn't matter.

duration

default: 500 How long the animation should last.

delay

default: 0 Allows to start with a specified delay.

reduceMotion

default: ReduceMotion.System Determines how the animation responds to the device's reduced motion accessibility setting.

withCallback

Allows to execute code when keyframe animation ends.

Example

```
export function KeyframeAnimation() {
  const [show, setShow] = useState(false);

  const enteringAnimation = new Keyframe({
    0: {
      originX: 50,
      transform: [{ rotate: '45deg' }],
    },
    30: {
      originX: 10,
      transform: [{ rotate: '-90deg' }],
    },
    100: {
      originX: 0,
      transform: [{ rotate: '0deg' }],
      easing: Easing.quad,
    },
  }).duration(2000);

  const exitingAnimation = new Keyframe({
    0: {
      opacity: 1,
      transform: [{ skewX: '0deg' }],
    },
    30: {
      opacity: 0.5,
      transform: [{ skewX: '40deg' }],
      easing: Easing.exp,
    },
    100: {
      opacity: 0,
      transform: [{ skewX: '-10deg' }],
    },
  }).duration(2000);

  return (
    <View style={{ flexDirection: 'column-reverse' }}>
      <Button
        title="animate"
        onPress={() => {
          setShow((last) => !last);
        }}
      />
      <View
        style={{ height: 400, alignItems: 'center', justifyContent: 'center' }}>
        {show && (
          <Animated.View
            entering={enteringAnimation}
            exiting={exitingAnimation}
            style={{
              height: 100,
              width: 200,
              backgroundColor: 'blue',
              alignItems: 'center',
              justifyContent: 'center',
            }}
          />
        )}
      </View>
    </View>
  );
}
```

0:00 / 0:06

Edit this page