

Easing

The `Easing` module implements common easing functions. This module is used by `Animated.timing()` to convey physically believable motion in animations.

You can find a visualization of some common easing functions at <http://easings.net/>

Predefined animations

The `Easing` module provides several predefined animations through the following methods:

- `back` provides a basic animation where the object goes slightly back before moving forward
- `bounce` provides a bouncing animation
- `ease` provides a basic inertial animation
- `elastic` provides a basic spring interaction

Standard functions

Three standard easing functions are provided:

- `linear`
- `quad`
- `cubic`

The `poly` function can be used to implement quartic, quintic, and other higher power functions.

Additional functions

Additional mathematical functions are provided by the following methods:

- `bezier` provides a cubic bezier curve

- `circle` provides a circular function
- `sin` provides a sinusoidal function
- `exp` provides an exponential function

The following helpers are used to modify other easing functions.

- `in` runs an easing function forwards
- `inOut` makes any easing function symmetrical
- `out` runs an easing function backwards

Example

TypeScript

JavaScript

Easing Demo

^ Expo

```
import React from 'react';
import {
  Animated,
  Easing,
  SectionList,
  StatusBar,
  StyleSheet,
  Text,
  TouchableOpacity,
  View,
} from 'react-native';
import type {EasingFunction} from 'react-native';

const App = () => {
  let opacity = new Animated.Value(0);

  const animate = (easing: EasingFunction) => {
    opacity.setValue(0);
    Animated.timing(opacity, {
      toValue: 1,
      duration: 1200,
      easing,
      useNativeDriver: true,
    }).start();
  };
};
```

Preview



My Device

iOS

Android

Web

Reference

Methods

step0()

```
static step0(n: number);
```

A stepping function, returns 1 for any positive value of n .

step1()

```
static step1(n: number);
```

A stepping function, returns 1 if n is greater than or equal to 1.

linear()

```
static linear(t: number);
```

A linear function, $f(t) = t$. Position correlates to elapsed time one to one.

<http://cubic-bezier.com/#0,0,1,1>

ease()

```
static ease(t: number);
```

A basic inertial interaction, similar to an object slowly accelerating to speed.

<http://cubic-bezier.com/#.42,0,1,1>

quad()

```
static quad(t: number);
```

A quadratic function, $f(t) = t * t$. Position equals the square of elapsed time.

<http://easings.net/#easeInQuad>

cubic()

```
static cubic(t: number);
```

A cubic function, $f(t) = t * t * t$. Position equals the cube of elapsed time.

<http://easings.net/#easeInCubic>

poly()

```
static poly(n: number);
```

A power function. Position is equal to the Nth power of elapsed time.

n = 4: <http://easings.net/#easeInQuart> n = 5: <http://easings.net/#easeInQuint>

sin()

```
static sin(t: number);
```

A sinusoidal function.

<http://easings.net/#easeInSine>

circle()

```
static circle(t: number);
```

A circular function.

<http://easings.net/#easeInCirc>

exp()

```
static exp(t: number);
```

An exponential function.

<http://easings.net/#easeInExpo>

elastic()

```
static elastic(bounciness: number);
```

A basic elastic interaction, similar to a spring oscillating back and forth.

Default bounciness is 1, which overshoots a little bit once. 0 bounciness doesn't overshoot at all, and bounciness of $N > 1$ will overshoot about N times.

<http://easings.net/#easeInElastic>

back()

```
static back(s)
```

Use with `Animated.parallel()` to create a basic effect where the object animates back slightly as the animation starts.

bounce()

```
static bounce(t: number);
```

Provides a basic bouncing effect.

<http://easings.net/#easeInBounce>

bezier()

```
static bezier(x1: number, y1: number, x2: number, y2: number);
```

Provides a cubic bezier curve, equivalent to CSS Transitions' `transition-timing-function`.

A useful tool to visualize cubic bezier curves can be found at <http://cubic-bezier.com/>

in()

```
static in(easing: number);
```

Runs an easing function forwards.

out()

```
static out(easing: number);
```

Runs an easing function backwards.

inOut()

```
static inOut(easing: number);
```

Makes any easing function symmetrical. The easing function will run forwards for half of the duration, then backwards for the rest of the duration.

Is this page useful?  



Edit this page

Last updated on **Jun 21, 2023**