LayoutAnimation

Automatically animates views to their new positions when the next layout happens.

A common way to use this API is to call it before updating the state hook in functional components and calling `setState` in class components.

Note that in order to get this to work on **Android** you need to set the following flags via `UIManager`:

```
if (Platform.OS === 'android') {
  if (UIManager.setLayoutAnimationEnabledExperimental) {
    UIManager.setLayoutAnimationEnabledExperimental(true);
  }
}
```

Example

LayoutAnimation

Expo

```
import React, {useState} from 'react';
import {
  LayoutAnimation,
  Platform,
  StyleSheet,
  Text,
  TouchableOpacity,
  UIManager,
  View,
} from 'react-native';

if (
  Platform.OS === 'android' &&
  UIManager.setLayoutAnimationEnabledExperimental
) {
  UIManager.setLayoutAnimationEnabledExperimental(true);
}
const App = () => {
  const [expanded, setExpanded] = useState(false);

  return (
    <View style={style.container}>
      <TouchableOpacity
        onPress={() => {

LayoutAnimation.configureNext(LayoutAnimation.Presets.sprin
```

Download Expo Go and scan the QR code to get started.

Connected devices 0

Log in or set a Device ID to open this Snack from Expo Go on your device or simulator.

Preview My Device iOS Android

Reference

Methods

configureNext()

```
static configureNext(
  config: LayoutAnimationConfig,
  onAnimationDidEnd?: () => void,
  onAnimationDidFail?: () => void,
);
```

Schedules an animation to happen on the next layout.

Parameters:

NAME	TYPE	REQUIRED	DESCRIPTION
config	object	Yes	See config description below.
onAnimationDidEnd	function	No	Called when the animation finished.
onAnimationDidFail	function	No	Called when the animation failed.

The `config` parameter is an object with the keys below. `create` returns a valid object for `config`, and the `Presets` objects can also all be passed as the `config`.

- `duration` in milliseconds
- `create`, optional config for animating in new views
- `update`, optional config for animating views that have been updated
- `delete`, optional config for animating views as they are removed

The config that's passed to `create`, `update`, or `delete` has the following keys:

- `type`, the animation type to use
- `property`, the layout property to animate (optional, but recommended for `create` and `delete`)
- `springDamping` (number, optional and only for use with `type: Type.spring`)
- `initialVelocity` (number, optional)
- `delay` (number, optional)
- `duration` (number, optional)

`create()`

```
static create(duration, type, creationProp)
```

Helper that creates an object (with `create`, `update`, and `delete` fields) to pass into `configureNext`. The `type` parameter is an animation type, and the `creationProp` parameter is a layout property.

Example:

LayoutAnimation

Expo

```
import React, {useState} from 'react';
import {
  View,
  Platform,
  UIManager,
  LayoutAnimation,
  StyleSheet,
  Button,
} from 'react-native';

if (
  Platform.OS === 'android' &&
  UIManager.setLayoutAnimationEnabledExperimental
) {
  UIManager.setLayoutAnimationEnabledExperimental(true);
}

const App = () => {
  const [boxPosition, setBoxPosition] = useState('left');

  const toggleBox = () => {
    LayoutAnimation.configureNext({
      duration: 500,
      create: {type: 'linear', property: 'opacity'},
      update: {type: 'spring', springDamping: 0.4},
      delete: {type: 'linear', property: 'opacity'},
```

Preview My Device iOS Android

Properties

Types

An enumeration of animation types to be used in the `create` method, or in the `create` / `update` / `delete` configs for `configureNext`. (example usage: `LayoutAnimation.Types.easeIn`)

TYPES
spring
linear

TYPES
easeInEaseOut
easeIn
easeOut
keyboard

Properties

An enumeration of layout properties to be animated to be used in the `create` method, or in the `create` / `update` / `delete` configs for `configureNext`. (example usage: `LayoutAnimation.Properties.opacity`)

PROPERTIES
opacity
scaleX
scaleY
scaleXY

Presets

A set of predefined animation configs to pass into `configureNext`.

PRESETS	VALUE
easeInEaseOut	`create(300, 'easeInEaseOut', 'opacity')`
linear	`create(500, 'linear', 'opacity')`
spring	`{duration: 700, create: {type: 'linear', property: 'opacity'}, update: {type: 'spring', springDamping: 0.4}, delete: {type: 'linear', property: 'opacity'} }`

`easeInEaseOut`

Calls `configureNext()` with `Presets.easeInEaseOut`.

`linear`

Calls `configureNext()` with `Presets.linear`.

`spring`

Calls `configureNext()` with `Presets.spring`.

Example:

LayoutAnimation

Expo

```
import React, {useState} from 'react';
import {
  View,
  Platform,
  UIManager,
  LayoutAnimation,
  StyleSheet,
  Button,
} from 'react-native';

if (
  Platform.OS === 'android' &&
  UIManager.setLayoutAnimationEnabledExperimental
) {
  UIManager.setLayoutAnimationEnabledExperimental(true);
}

const App = () => {
  const [firstBoxPosition, setFirstBoxPosition] =
useState('left');
  const [secondBoxPosition, setSecondBoxPosition] =
useState('left');
  const [thirdBoxPosition, setThirdBoxPosition] =
useState('left');

  const toggleFirstBox = () => {
```

Download Expo Go and scan the QR code to get started.

Connected devices 0

Log in or set a Device ID to open this Snack from Expo Go on your device or simulator.

Preview My Device iOS Android

Is this page useful?

Edit this page

*Last updated on **Jun 21, 2023***