Version: 3.x

useAnimatedProps

`useAnimatedStyle` lets you create an animated props object which can be animated using shared values. This object is used to animate properties of third-party components.

For animating style use `useAnimatedStyle` instead.

Reference

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

function App() {
  const animatedProps = useAnimatedProps(() => {
    return {
      opacity: sv.value ? 1 : 0,
    };
  });

  return <Animated.View animatedProps={animatedProps} />;
}
```

Type definitions

Arguments

`updater`

A function returning an object with properties you want to animate.

`dependencies` Optional

An optional array of dependencies.

Only relevant when using Reanimated without the Babel plugin on the Web.

`adapters` Optional

An optional function or an array of functions.

Sometimes when working with third-party libraries properties might be named differently on the API surface from what they really represent on the native side. Adapters make it possible to handle these cases by defining a way to convert these props.

Reanimated comes with two built-in adapters:

- `SVGAdapter` for handling `transform` property in `react-native-svg`
- `TextInputAdapter`.

You can create your own adapters using `createAnimatedPropAdapter` function.

Here's an example of adapting `fill` and `stroke` properties from `react-native-svg` to be able to animate them with Reanimated.

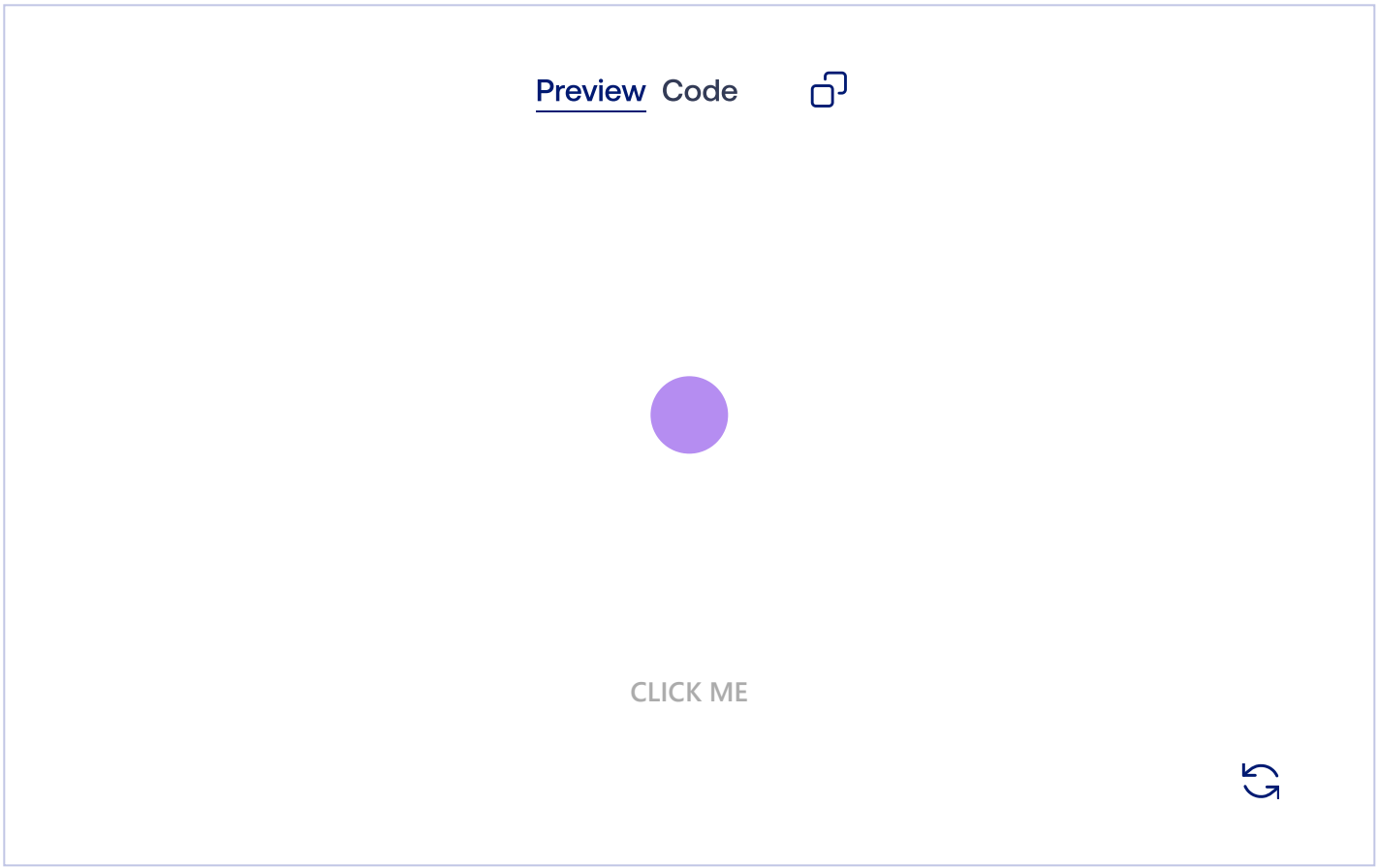
Expand the full code

```
const adapter = createAnimatedPropAdapter(
  (props) => {
    if (Object.keys(props).includes('fill')) {
      props.fill = { type: 0, payload: processColor(props.fill) };
    }
    if (Object.keys(props).includes('stroke')) {
      props.stroke = { type: 0, payload: processColor(props.stroke) };
    }
  },
  ['fill', 'stroke']
);
```

Returns

`useAnimatedProps` returns an animated props object which has to be passed to `animatedProps` property of an Animated component that you want to animate.

Example



Remarks

- You can share animated props between components to avoid code duplication.
- We recommend to create adapters outside of the component's body to avoid unnecessary recalculations.

Platform compatibility

Android	iOS	Web
✅	✅	✅

Edit this page