Version: 3.x

interpolate

> 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 As we're rewriting the documentation some of the pages might be a little outdated.

Sometimes you need to map a value from one range to another. This is where you should use the `interpolate` function which approximates values between points in the output range and lets you map a value inside the input range to a corresponding approximation in the output range. It also supports a few types of Extrapolation to enable mapping outside the range.

Arguments

`value` **[Float]**

Value from within the input range that should be mapped to a value from the output range.

`input range` **[Float[]]**

An array of Floats that contains points that indicate the range of the input value. Values in the input range should be increasing.

`output range` **[Float[]]**

An array of Floats that contains points that indicate the range of the output value. It should have at least the same number of points as the input range.

`extrapolation type` **[Object | String]**

Can be either an object or a string. If an object is passed it should specify extrapolation explicitly for the right and left sides. If extrapolation for a side is not provided, it defaults to `Extrapolation.EXTEND`. Example extrapolation type object:

```
const extrapolation = {
  extrapolateLeft: Extrapolation.CLAMP,
  extrapolateRight: Extrapolation.IDENTITY,
};
```

If a string is provided, the provided extrapolation type is applied to both sides.

> ⓘ INFO
>
> Available extrapolation types:
>
> - `Extrapolation.CLAMP` - clamps the value to the edge of the output range
> - `Extrapolation.IDENTITY` - returns the value that is being interpolated
> - `Extrapolation.EXTEND` - approximates the value even outside of the range
>
> Available extrapolation string values:
>
> - `clamp`
> - `identity`
> - `extend`

Returns

`interpolate` returns the value after interpolation from within the output range.

Example

```
import React from 'react';
import { View, StyleSheet, Dimensions } from 'react-native';
import Animated, {
  useSharedValue,
  useAnimatedScrollHandler,
  useAnimatedStyle,
  interpolate,
} from 'react-native-reanimated';
```

```
export const HEADER_IMAGE_HEIGHT = Dimensions.get('window').width / 3;

export default function Test() {
  const scrollY = useSharedValue(0);
  const scrollHandler = useAnimatedScrollHandler({
    onScroll: (e) => {
      scrollY.value = e.contentOffset.y;
    },
  });
  const animatedStyles = useAnimatedStyle(() => {
    const scale = interpolate(scrollY.value, [-100, 0], [2, 1], {
      extrapolateRight: Extrapolation.CLAMP,
    });

    return {
      transform: [{ scale: scale }],
    };
  });

  return (
    <View style={{ flex: 1, alignItems: 'center' }}>
      <Animated.View
        style={[
          {
            position: 'absolute',
            top: 20,
            left: 0,
            width: 20,
            height: 20,
            backgroundColor: 'blue',
          },
          animatedStyles,
        ]}
      />

      <Animated.ScrollView
        style={StyleSheet.absoluteFill}
        onScroll={scrollHandler}></Animated.ScrollView>
    </View>
  );
}
```

Edit this page