Version: 3.x

useSharedValue

`useSharedValue` lets you define shared values in your components.

Reference

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

function App() {
  const sv = useSharedValue(100);

  // read a shared value
  console.log(sv.value);

  // and modify it
  sv.value += 50;
}
```

Type definitions

Arguments

`initialValue`

The value you want to be initially stored to a `.value` property. It can be any JavaScript value like `number`, `string` or `boolean` but also data structures such as `array` and `object`.

Returns

`useSharedValue` returns a shared value with a single `value` property initially set to the `initialValue`.

Values stored in shared values can be accessed and modified by their `.value` property.

Example

Preview Code

```
import React from 'react';
import { Button, StyleSheet, View } from 'react-native';
import Animated, { useSharedValue } from 'react-native-reanimated';

export default function App() {
  const width = useSharedValue(100);

  const handlePress = () => {
    width.value += 50;
  };

  return (
    <View style={styles.container}>
      <Animated.View style={{ ...styles.box, width }} />
      <Button onPress={handlePress} title="Click me" />
```

Remarks

- When you change the `sv.value` Reanimated will update the styles and keep the shared value in sync between the threads. However, this won't trigger a typical React re-render because a shared value is a plain JavaScript object.
- When you change the `sv.value` the update will happen synchronously on the UI thread. On the other hand, on the JavaScript thread the update is asynchronous. This means when you try to immediately log the `value` after the change it will log the previously stored value.

```
function App() {
  const sv = useSharedValue(100); // initially set 100

  sv.value += 50; // changing value stored in a shared value

  console.log(sv.value); // will still log 100
}
```

- Stay away from destructuring assignment when working with shared values. While this is a completely valid JavaScript code it will make Reanimated unable to keep the reactivity of a shared value.

```
function App() {
  let { value } = sv; // don't do this

  console.log(value); // you can read the value just fine

  value += 50; // but this won't update the styles
}
```

Platform compatibility

Android	iOS	Web
✅	✅	✅

Edit this page