Version: 3.x

Accessibility

In this section, we will explore how Reanimated provides support for enhanced accessibility in animations, particularly through its reduced motion functionality. This feature ensures a smoother experience for users who may have motion sensitivities or prefer less movement.

The reduced motion configuration can be used to define how animations should respond to the system's reduced motion setting. For a given animation, the value can be set to:

- `ReduceMotion.System` - This value adjusts the animation behavior based on whether the reduced motion accessibility setting is activated on the device. When enabled, the animation is disabled; otherwise, it remains active.
- `ReduceMotion.Always` - With this setting, the animation is consistently disabled, regardless of the device's accessibility configuration.
- `ReduceMotion.Never` - This option ensures that the animation remains enabled at all times.

By default all animations are configured with `ReduceMotion.System`.

Reduced motion in animations

```
import { withDelay, withTiming } from 'react-native-reanimated';

function App() {
  sv1.value = withTiming(0, { reduceMotion: ReduceMotion.System });
  sv2.value = withDelay(
    1000,
    withTiming(toValue, { duration }),
    ReduceMotion.System
  );
  // ...
}
```

When reduced motion is enabled:

- `withSpring` and `withTiming` return the `toValue` immediately

- `withDecay` returns the current value immediately, taking into account the clamp parameter
- `withDelay` initiates the next animation immediately
- `withRepeat`
  - when the `numberOfReps` is infinite or even and the animation is reversed, then the repeated animation does not start
  - otherwise, the repeated animation runs once
- `withSequence` exclusively starts animations that have reduced motion disabled

Higher-order animations pass the configuration to their children, only if the children have not been configured by the user.

For example, this animation will instantaneously reach the `toValue`:

```
import { withDelay, withTiming } from 'react-native-reanimated';

function App() {
  sv.value = withDelay(
    1000,
    withTiming(toValue, { duration }),
    ReduceMotion.Always
  );
  // ...
}
```

This animation will execute as usual even if reduced motion is enabled on the device:

```
import { withDelay, withTiming } from 'react-native-reanimated';

function App() {
  sv.value = withDelay(
    1000,
    withTiming(toValue, { duration }),
    ReduceMotion.Never
  );
  // ...
}
```

And here `withTiming` will be executed as usual and with no delay:

```
import { withDelay, withTiming } from 'react-native-reanimated';

function App() {
  sv.value = withDelay(
    1000,
    withTiming(toValue, { duration, reduceMotion: ReduceMotion.Never }),
    ReduceMotion.Always
  );
  // ...
}
```

Reduced motion in Layout Animations

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

function App() {
  const entering = BounceIn.reduceMotion(ReduceMotion.System);
  // ...
}
```

When reduced motion is enabled:

- entering, keyframe, and layout animations instantaneously reach their endpoints.
- exiting animations and shared transitions are omitted.

`useReducedMotion`

This hook returns a boolean indicating whether the reduced motion setting was enabled when the app started. It can be used in conjunction with other libraries or to conditionally display animations that are less intrusive.

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

function App() {
  const reduceMotion = useReducedMotion();
  const entering = reduceMotion
    ? FadeIn.reduceMotion(ReduceMotion.Never)
```

```
    : BounceIn;
  // ...
}
```

Edit this page