Rakamin

React Native UI Interaction and Animation

Building Interactive and Animated User Interfaces

Outline Pembelajaran

Rakamin Academy

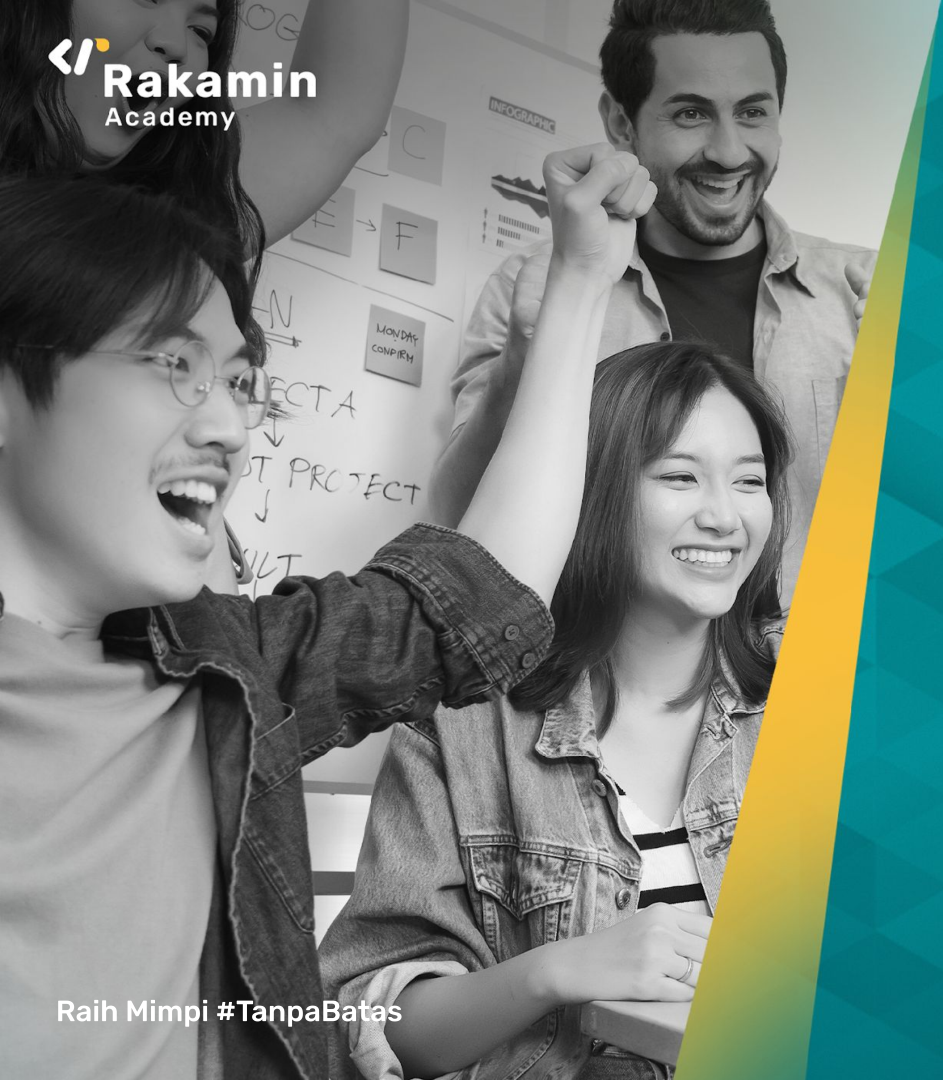
Objectives

- ☐ Overview of UI interaction in React Native.
- ☐ Importance of gestures and animations in modern mobile apps.
- ☐ Understand how to bring the concept of gestures and animations in React Native into practice.

React Native Gesture Handler

Raih Mimpi **#TanpaBatas**

Rakamin Academy

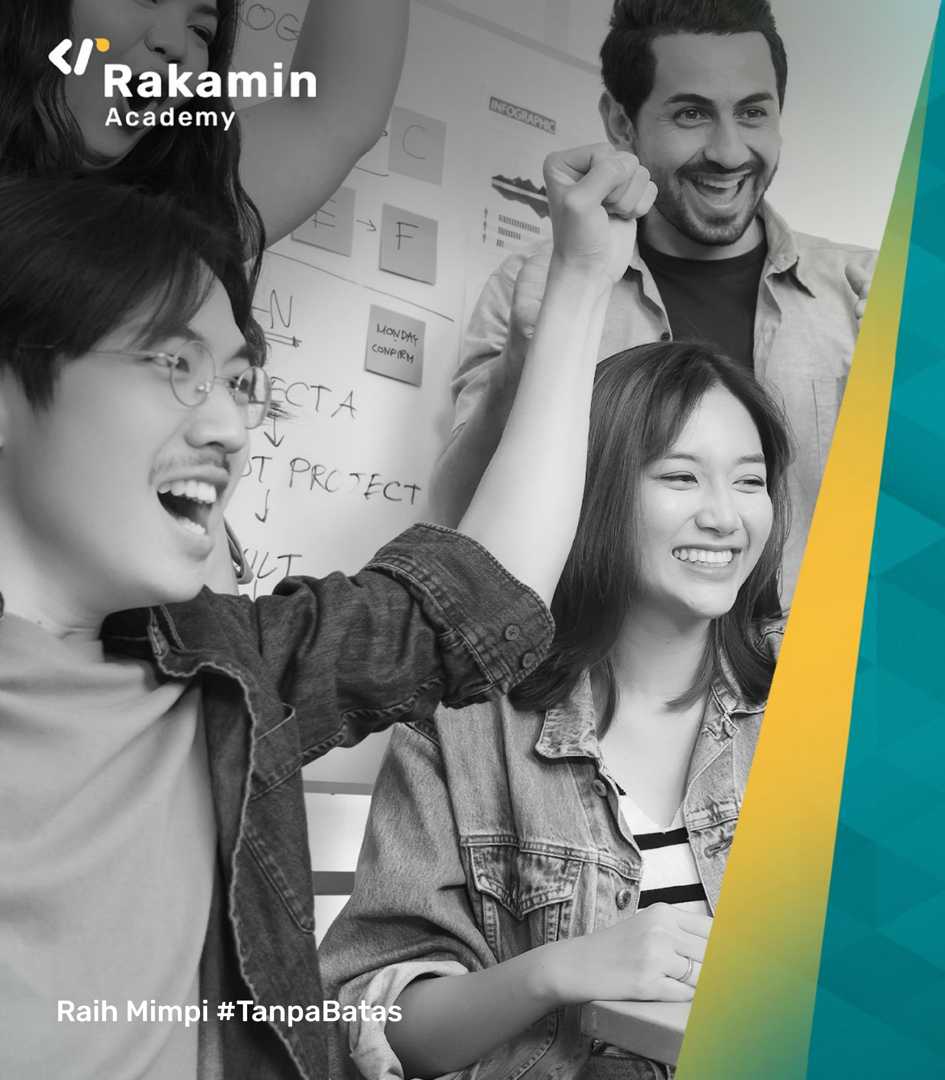


React Native Gesture Handler

React Native Gesture Handler is a library that brings gesture handling capabilities to React Native apps, enabling more native and responsive gesture interactions.

Raih Mimpi #TanpaBatas

Rakamin Academy



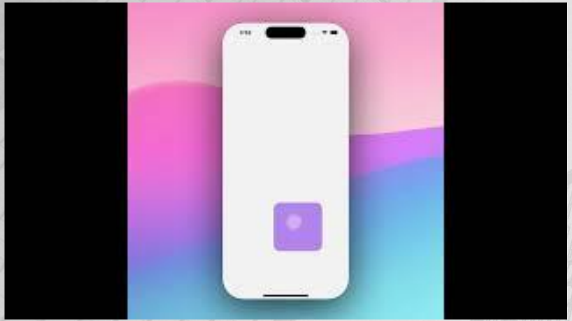
Why React Native Gesture Handler?

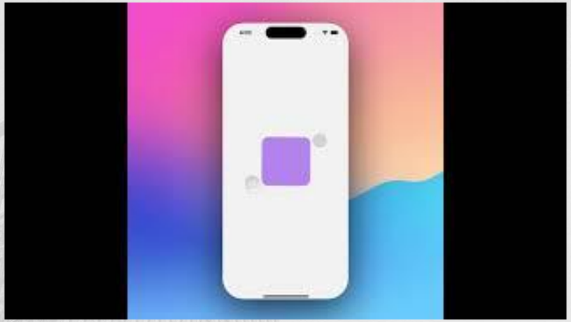
Native Feel: Gesture Handler allows you to handle gestures directly in the native layer, providing a more responsive and fluid user experience.

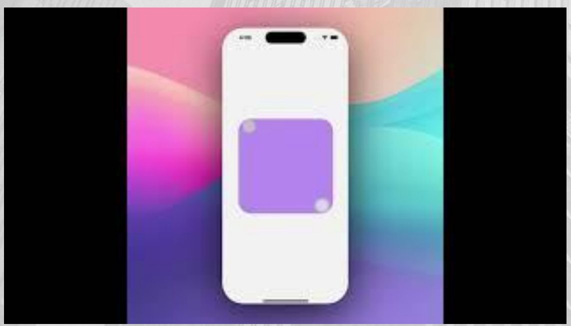
Compatibility: Works well with React Native's native components and third-party libraries.

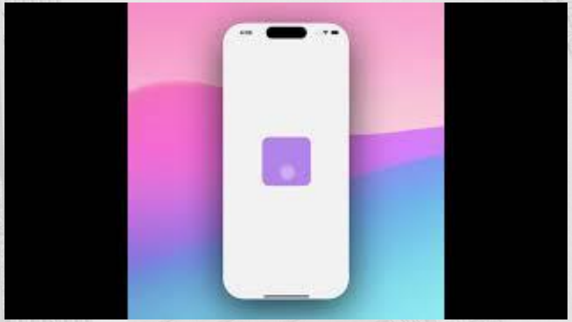
Complex Gestures: Supports complex gestures that are difficult to implement using the standard onPress or onLongPress handlers.

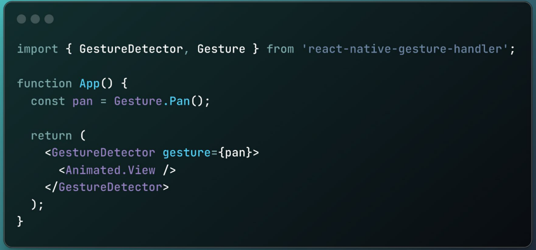
Supported Gestures


Pan Gesture


Rotation Gesture

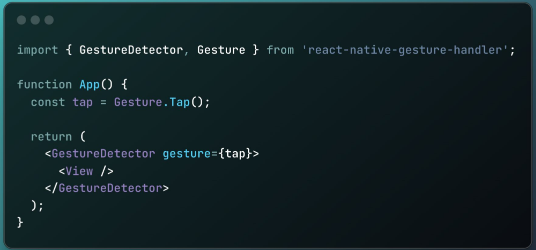

Pinch Gesture


Fling Gesture



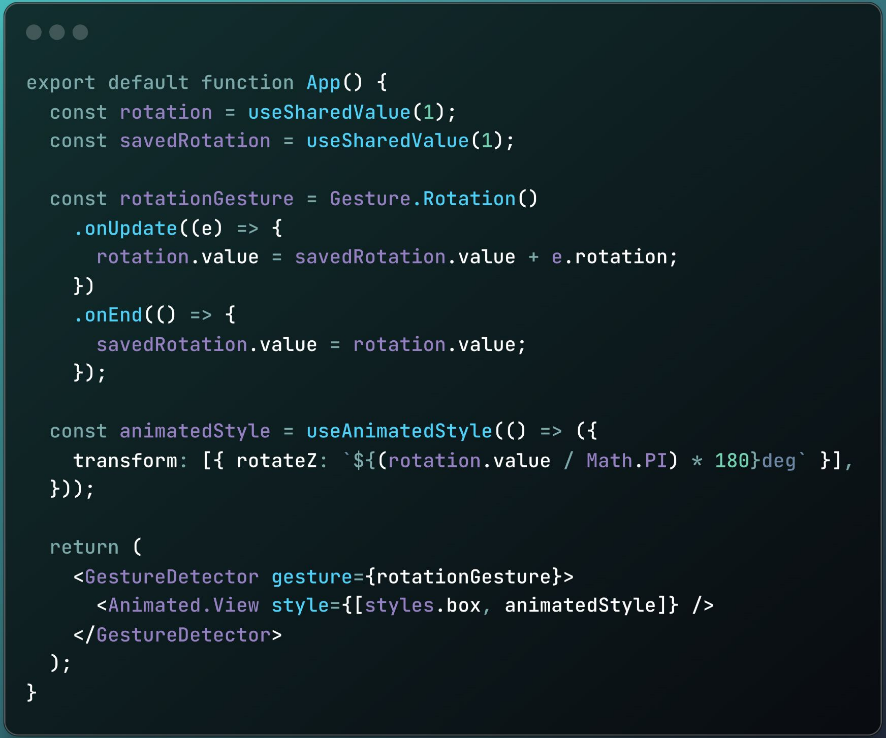
```
import { GestureDetector, Gesture } from 'react-native-gesture-handler';

function App() {
  const pan = Gesture.Pan();

  return (
    <GestureDetector gesture={pan}>
      <Animated.View />
    </GestureDetector>
  );
}
```

Pan



```
import { GestureDetector, Gesture } from 'react-native-gesture-handler';

function App() {
  const tap = Gesture.Tap();

  return (
    <GestureDetector gesture={tap}>
      <View />
    </GestureDetector>
  );
}
```

Tap

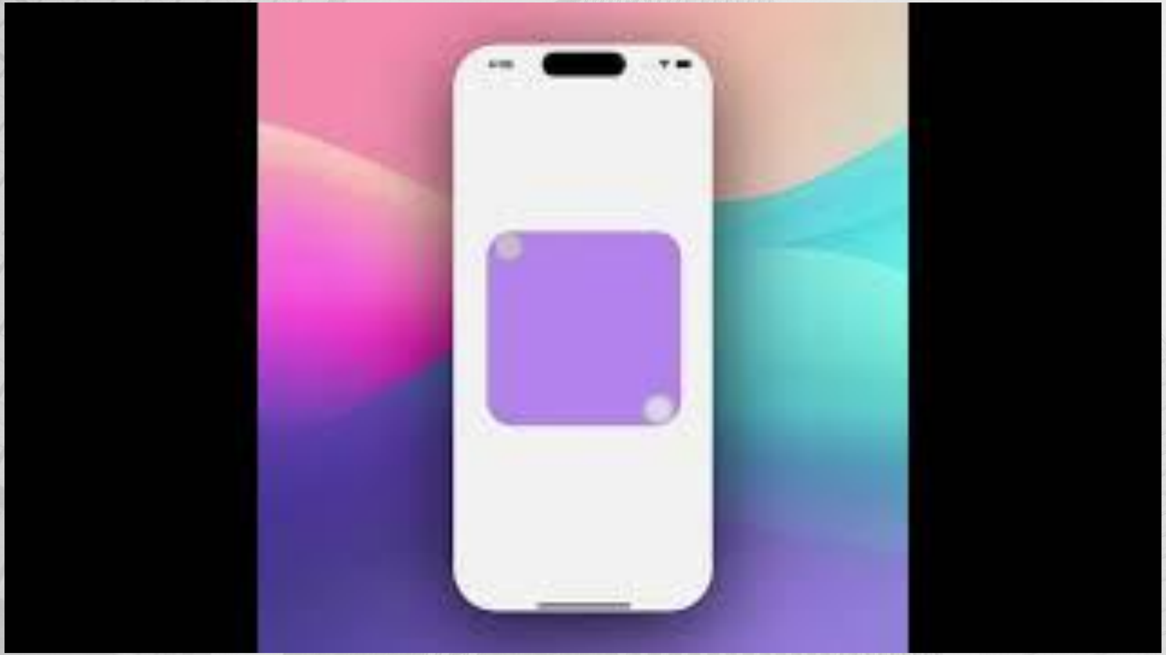


```
export default function App() {
  const rotation = useSharedValue(1);
  const savedRotation = useSharedValue(1);

  const rotationGesture = Gesture.Rotation()
    .onUpdate((e) => {
      rotation.value = savedRotation.value + e.rotation;
    })
    .onEnd(() => {
      savedRotation.value = rotation.value;
    });

  const animatedStyle = useAnimatedStyle(() => ({
    transform: [{ rotateZ: `${(rotation.value / Math.PI) * 180}deg` }],
  }));

  return (
    <GestureDetector gesture={rotationGesture}>
      <Animated.View style={[styles.box, animatedStyle]} />
    </GestureDetector>
  );
}
```

Rotation

Sample Full Code

Pinch Gesture Example.



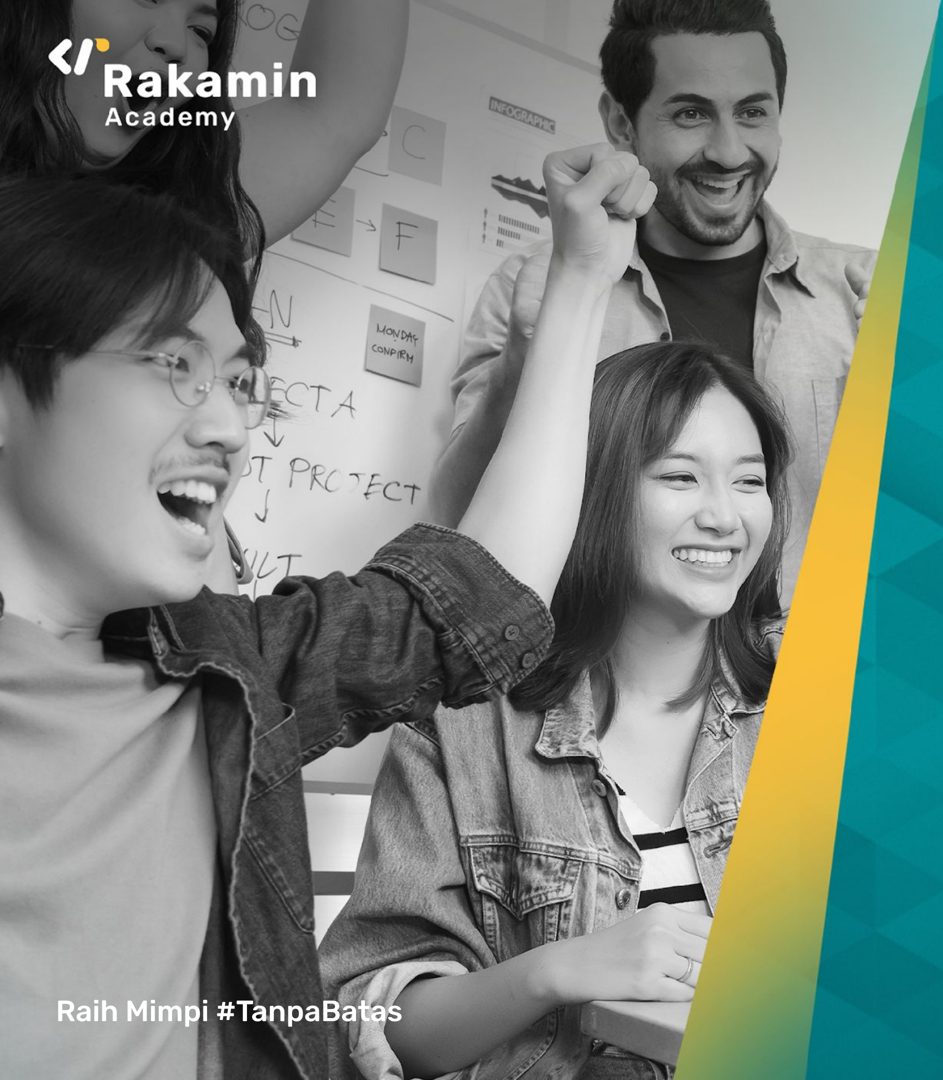
```
import React from 'react';
import { Dimensions, StyleSheet } from 'react-native';
import {
  Gesture,
  GestureDetector,
  GestureHandlerRootView,
} from 'react-native-gesture-handler';
import Animated, {
  useSharedValue,
  useAnimatedStyle,
} from 'react-native-reanimated';

const { width, height } = Dimensions.get('screen');

function clamp(val, min, max) {
  return Math.min(Math.max(val, min), max);
}

export default function App() {
  const scale = useSharedValue(1);
  const startScale = useSharedValue(0);

  const pinch = Gesture.Pinch()
    .onStart(() => {
      startScale.value = scale.value;
    })
    .onUpdate((event) => {
      scale.value = clamp(
        startScale.value * event.scale,
        0.5,
        Math.min(width / 100, height / 100)
      );
    })
    .runOnJS(true);

  const boxAnimatedStyles = useAnimatedStyle(() => ({
    transform: [{ scale: scale.value }],
  }));

  return (
    <GestureHandlerRootView style={styles.container}>
      <GestureDetector gesture={pinch}>
        <Animated.View style={[styles.box, boxAnimatedStyles]}></Animated.View>
      </GestureDetector>
    </GestureHandlerRootView>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  box: {
    width: 100,
    height: 100,
    borderRadius: 20,
    backgroundColor: '#b58df1',
  },
  dot: {
    width: 24,
    height: 24,
    borderRadius: 12,
    backgroundColor: '#ccc',
    position: 'absolute',
    left: '50%',
    top: '50%',
    pointerEvents: 'none',
  },
});
```

Rakamin Academy

Sample Use Cases

- **Swipe to Delete:** Swipe a list item to delete it.
- **Drag and Drop:** Drag items from one location to another.
- **Interactive Buttons:** Buttons that respond to long presses with a visual change.

React Native Reanimated

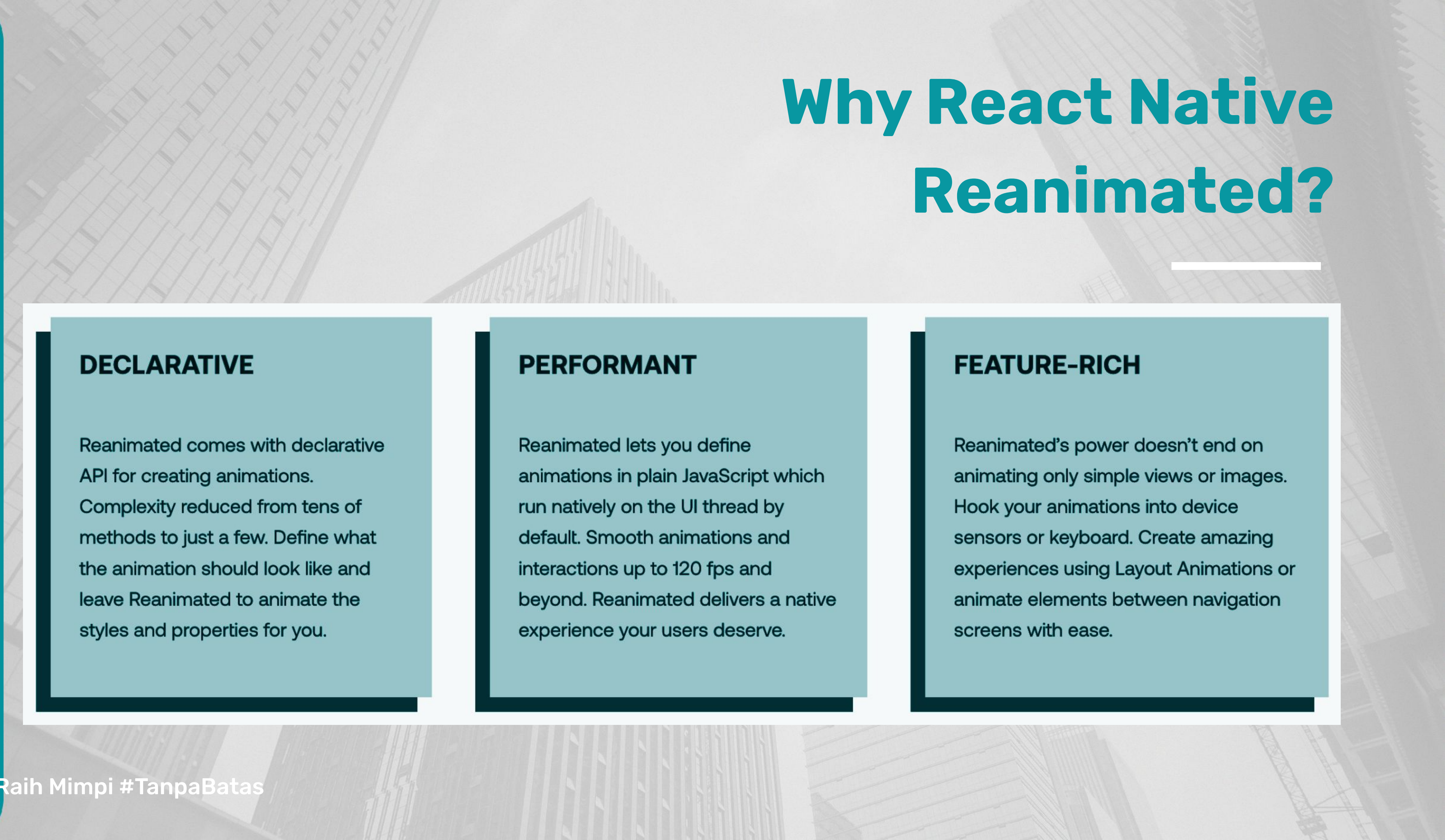


Rakamin Academy

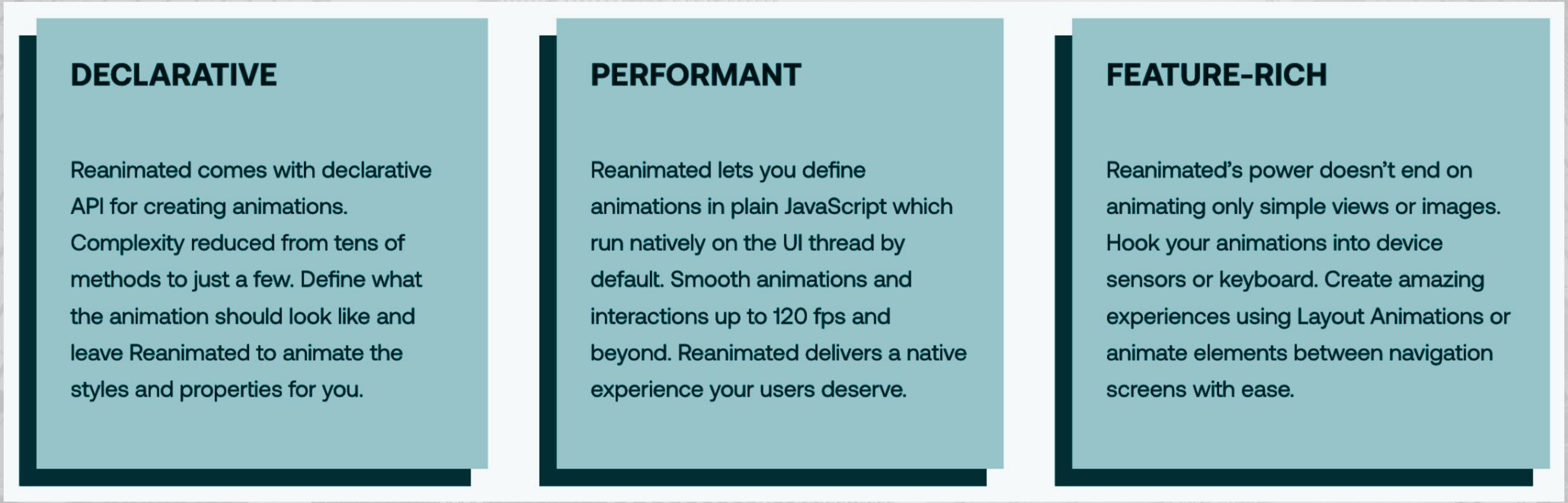
React Native Reanimated

React Native Reanimated is a powerful library for creating smooth, high-performance animations in React Native apps, allowing animations to run directly on the native thread for better fluidity and responsiveness.

Raih Mimpi #TanpaBatas



Why React Native Reanimated?



DECLARATIVE

Reanimated comes with declarative API for creating animations. Complexity reduced from tens of methods to just a few. Define what the animation should look like and leave Reanimated to animate the styles and properties for you.

PERFORMANT

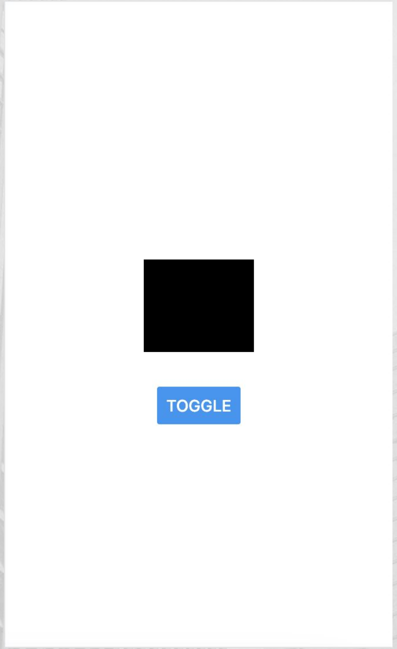
Reanimated lets you define animations in plain JavaScript which run natively on the UI thread by default. Smooth animations and interactions up to 120 fps and beyond. Reanimated delivers a native experience your users deserve.

FEATURE-RICH

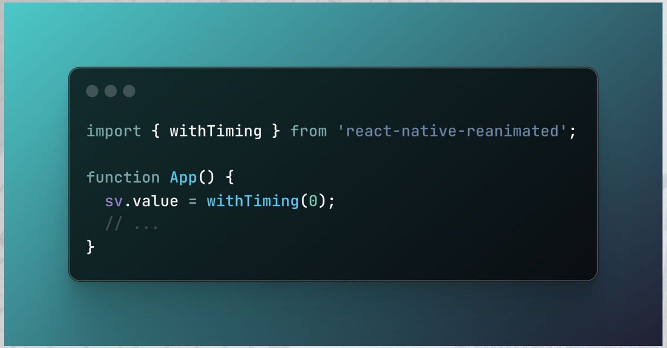
Reanimated's power doesn't end on animating only simple views or images. Hook your animations into device sensors or keyboard. Create amazing experiences using Layout Animations or animate elements between navigation screens with ease.

Sample Code

Simple random box width animation.

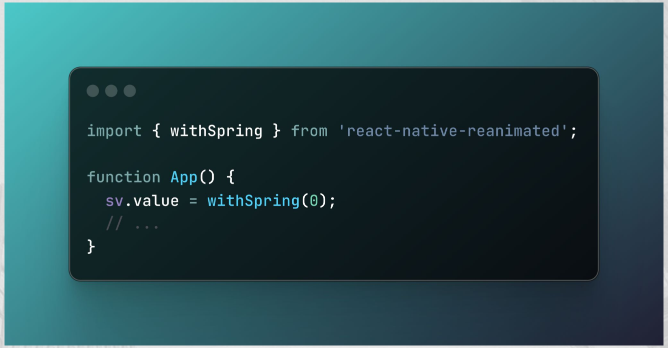


```
import Animated, {
  useSharedValue,
  withTiming,
  useAnimatedStyle,
  Easing,
} from 'react-native-reanimated';
import { View, Button, StyleSheet } from 'react-native';

export default function AnimatedStyleUpdateExample() {
  const randomWidth = useSharedValue(10);

  const config = {
    duration: 500,
    easing: Easing.bezier(0.5, 0.01, 0, 1),
  };

  const style = useAnimatedStyle(() => {
    return {
      width: withTiming(randomWidth.value, config),
    };
  });

  return (
    <View style={styles.container}>
      <Animated.View style={[styles.box, style]} />
      <Button
        title="toggle"
        onPress={() => {
          randomWidth.value = Math.random() * 350;
        }}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    alignItems: 'center',
    justifyContent: 'center',
  },
  box: {
    width: 100,
    height: 80,
    backgroundColor: 'black',
    margin: 30,
  },
});
```



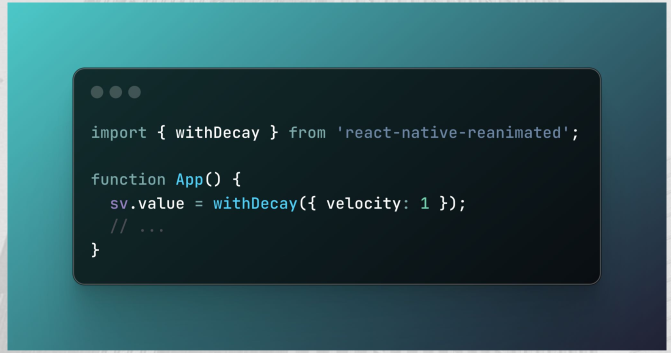
```
import { withTiming } from 'react-native-reanimated';

function App() {
  sv.value = withTiming(0);
  // ...
}
```

withTiming



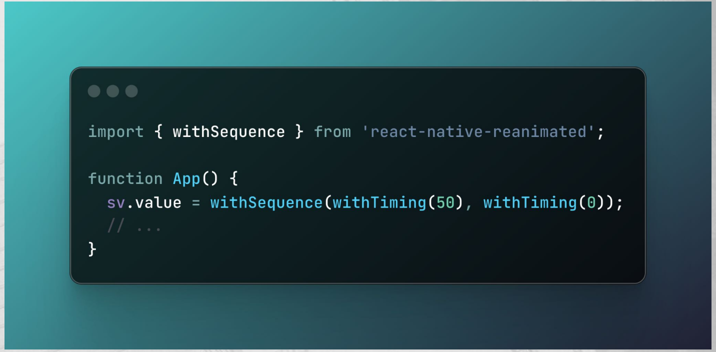
```
import { withSpring } from 'react-native-reanimated';

function App() {
  sv.value = withSpring(0);
  // ...
}
```

withSpring



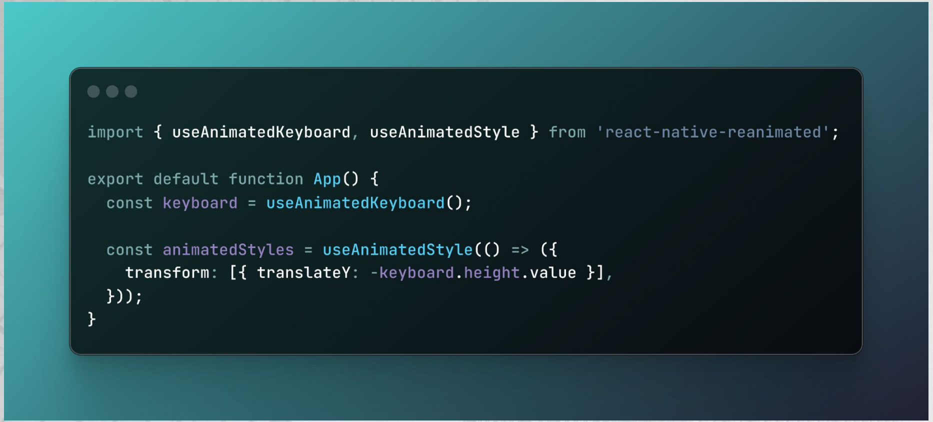
```
import { withDecay } from 'react-native-reanimated';

function App() {
  sv.value = withDecay({ velocity: 1 });
  // ...
}
```

withDecay



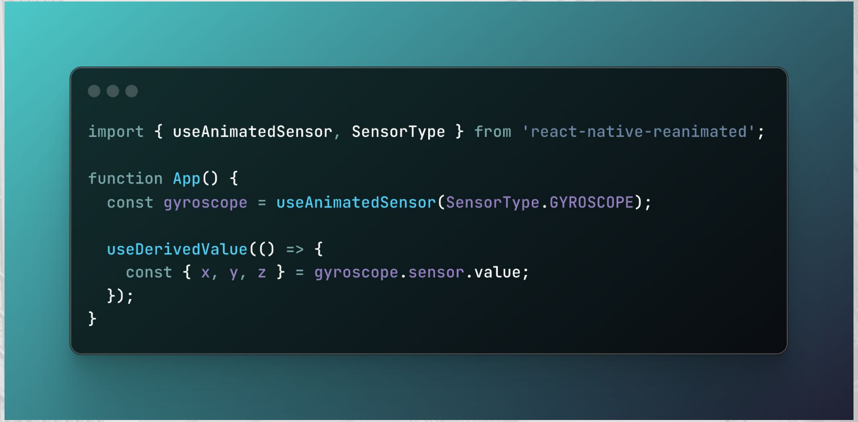
```
import { withSequence } from 'react-native-reanimated';

function App() {
  sv.value = withSequence(withTiming(50), withTiming(0));
  // ...
}
```

withSequence



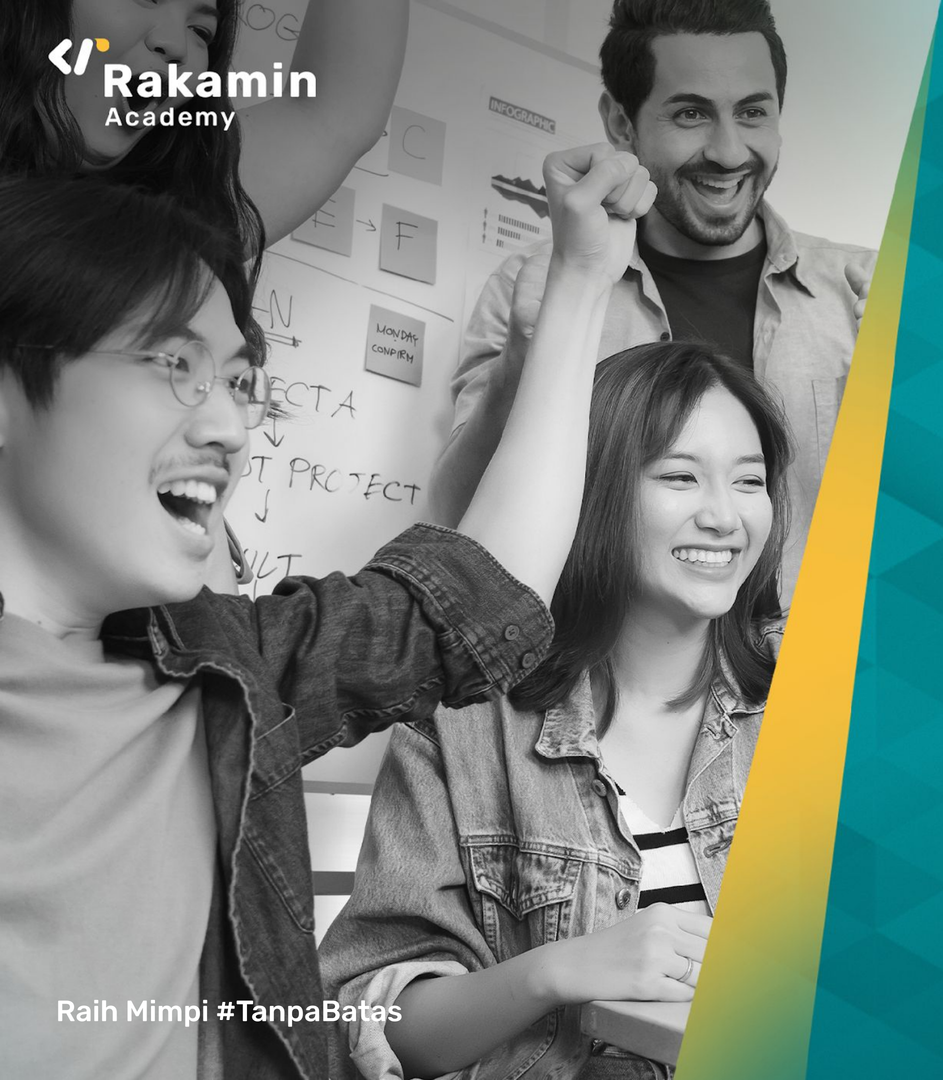
```
import { useAnimatedKeyboard, useAnimatedStyle } from 'react-native-reanimated';

export default function App() {
  const keyboard = useAnimatedKeyboard();

  const animatedStyles = useAnimatedStyle(() => ({
    transform: [{ translateY: -keyboard.height.value }],
  }));
}
```

useAnimatedKeyboard



```
import { useAnimatedSensor, SensorType } from 'react-native-reanimated';

function App() {
  const gyroscope = useAnimatedSensor(SensorType.GYROSCOPE);

  useDerivedValue(() => {
    const { x, y, z } = gyroscope.sensor.value;
  });
}
```

useAnimatedSensor



References

https://docs.swmansion.com/react-native-gesture-handler/

https://docs.swmansion.com/react-native-reanimated/

Rakamin Academy

Thank you

Raih Mimpi **#TanpaBatas**