Introduction

Version: 2.6.0 – 2.12.0

Introduction

Gesture Handler aims to replace React Native's built in touch system called Gesture Responder System.

The motivation for building this library was to address the performance limitations of React Native's Gesture Responder System and to provide more control over the built-in native components that can handle gestures. We recommend this talk by Krzysztof Magiera in which he explains issues with the responder system.

In a nutshell, the library provides:

- A way to use a platform's native touch handling system for recognizing pinch, rotation and pan (besides a few other gestures).
- The ability to define relations between gesture handlers, e.g. when you have a pan handler in `ScrollView` you can make that `ScrollView` wait until it knows pan won't recognize.
- Mechanisms to use touchables that run in native thread and follow platform default behavior; e.g. in the event they are in a scrollable component, turning into pressed state is slightly delayed to prevent it from highlighting when you fling.

INFO

It is recommended to use Reanimated 2 for animations when using React Native Gesture Handler as its more advanced features rely heavily on the worklets provided by Reanimated.

RNGH 2.0

RNGH2 introduces a new way of creating gestures. Instead of creating a gesture handler component for every gesture you want to create, you just need to create a `GestureDetector` component and assign to it all the gestures you want it to recognize. It is also designed to work seamlessly with `Reanimated 2` and it will automatically detect if it is installed, and start using it if it is. You can create gestures using the `Gesture` object and methods it provides, and configure them in the builder-like pattern. If you want to specify behavior between the gestures instead of using `waitFor` and `simultaneousGestures` you can use the new system of gesture composition. Along the new API, version 2.0 brings one of the most requested features: touch events and manual gestures. Thanks to great work done by the Reanimated team, we were able to provide synchronous communication between gestures and their native implementation using worklets. This allows to manage gesture state from the JS without risking race-conditions.

Interoperability with gesture handlers

The new API with gestures is somewhat compatible with the old gesture handlers. Unfortunately you cannot use the new gesture composing with gesture handlers, however you can still mark relations using refs. If you want to make a gesture handler wait for (or simultaneous with) a gesture, simply use withRef method on the gesture to set the ref object and pass it to the appropriate property on the gesture handler.

Similarly, if you want to make a gesture simultaneous with (or wait for failure of) a gesture handler, set the ref prop of the gesture handler and pass the same ref to the simultaneousWithExternalGesture or requireExternalGestureToFail method on the gesture object.

This should allow you to migrate your codebase from the gesture handlers to gestures smoothly and at your own pace. Just keep in mind that the gesture handlers cannot have the GestureDetector as their direct child, as it's a functional component.

Automatic workletization of gesture callbacks

Reanimated's Babel plugin is setup in a way that automatically marks callbacks passed to gestures in the configuration chain as worklets. This means that as long as all your callbacks are defined in a single chain, you don't need to add a `'worklet';` directive at the beginning of the functions. Here is an example that will be automatically workletized:

```
const gesture = Gesture.Tap().onBegin(() => {
  console.log(_WORKLET);
});
```

And here are some examples that won't:

```
const gesture = Gesture.Tap();
gesture.onBegin(() => {
  console.log(_WORKLET);
});
```

```
const callback = () => {
  console.log(_WORKLET);
};
const gesture = Gesture.Tap().onBegin(callback);
```

```
const callback = () => {
  console.log(_WORKLET);
};
const gesture = Gesture.Tap();
gesture.onBegin(callback);
```

Learning resources

Apps

Gesture Handler Example App – official gesture handler "showcase" app.

Gesture Handler Example on Expo – the official app you can install and play with using Expo.

Talks and workshops

Declarative future of gestures and animations in React Native by Krzysztof Magiera - talk that explains motivation behind creating gesture handler library. It also presents react-native-reanimated and how and when it can be used with gesture handler.

React Native workshop with Expo team @ReactEurope 2018 by Brent Vetne – great workshop explaining gesture handler in details and presenting a few exercises that will help get you started.

Living in an async world of React Native by Krzysztof Magiera – talk which highlights some issue with the React Native's touch system Gesture Handler aims to address. Also the motivation for building this library is explained.

React Native Touch & Gesture by Krzysztof Magiera - talk explaining JS responder system limitations and points out some of the core features of Gesture Handler.

Contributing

If you are interested in the project and want to contribute or support it in other ways don't hesitate to contact me on Twitter!

All PRs are welcome, but talk to us before you start working on something big.

The easiest way to get started with contributing code is by:

- Reviewing the list of open issues and trying to solve the one that seem approachable to you.
- Updating the documentation whenever you see some information is unclear, missing or out of date.

Code is only one way how you can contribute. You may want to consider replying on issues if you know how to help.

Community

We are very proud of the community that has been build around this package. We really appreciate all your help regardless of it is a pull request, issue report, helping others by commenting on existing issues or posting some demo or video tutorial on social media. If you've build something with this library you'd like to share, please contact us as we'd love to help share it with others.

Gesture Handler Team 🚀

Jakub Piasecki

Jakub Gonet

Krzysztof Magiera
@kzzzf

Sponsors

We really appreciate our sponsors! Thanks to them we can develop our library and make the react-native world a better place. Special thanks for:

[Shopify](#) [Expo](#)