Version: 2.6.0 – 2.12.0

Testing

> INFO
>
> If you want to use `fireGestureHandler` and `getByGestureTestId`, you need to import them from `react-native-gesture-handler/jest-utils` package.

fireGestureHandler(gestureOrHandler, eventList)

Simulates one event stream (i.e. event sequence starting with `BEGIN` state and ending with one of `END`/`FAIL`/`CANCEL` states), calling appropriate callbacks associated with given gesture handler.

Arguments

`gestureOrHandler`

Represents either:

1. Gesture handler component found by Jest queries (e.g. `getByTestId`)
2. Gesture found by `getByGestureTestId()`

`eventList`

Event data passed to appropriate callback. RNGH fills event list if required data is missing using these rules:

1. `oldState` is filled using state of the previous event. `BEGIN` events use `UNDETERMINED` value as previous event.
2. Events after first `ACTIVE` state can omit `state` field.
3. Handler specific data is filled (e.g. `numberOfTouches`, `x` fields) with defaults.
4. Missing `BEGIN` and `END` events are added with data copied from first and last passed event, respectively.
5. If first event don't have `state` field, the `ACTIVE` state is assumed.

Some examples:

```
const oldStateFilled = [
  { state: State.BEGAN },
  { state: State.ACTIVE },
  { state: State.END },
]; // three events with specified state are fired.

const implicitActiveState = [
  { state: State.BEGAN },
  { state: State.ACTIVE },
  { x: 5 },
  { state: State.END },
]; // 4 events, including two ACTIVE events (second one has overriden additional
data).

const implicitBegin = [
  { x: 1, y: 11 },
  { x: 2, y: 12, state: State.FAILED },
]; // 3 events, including implicit BEGAN, one ACTIVE, and one FAILED event with
additional data.

const implicitBeginAndEnd = [
  { x: 5, y: 15 },
  { x: 6, y: 16 },
  { x: 7, y: 17 },
]; // 5 events, including 3 ACTIVE events and implicit BEGAN and END events. BEGAN
uses first event's additional data, END uses last event's additional data.

const allImplicits = []; // 3 events, one BEGIN, one ACTIVE, one END with defaults.
```

Example

Extracted from RNGH tests, check `Events.test.tsx` for full implementation.

```
it('sends events with additional data to handlers', () => {
  const panHandlers = mockedEventHandlers();
  render(<SingleHandler handlers={panHandlers} treatStartAsUpdate />);
  fireGestureHandler<PanGesture>(getByGestureTestId('pan'), [
    { state: State.BEGAN, translationX: 0 },
    { state: State.ACTIVE, translationX: 10 },
    { translationX: 20 },
    { translationX: 20 },
    { state: State.END, translationX: 30 },
  ]);

  expect(panHandlers.active).toBeCalledTimes(3);
  expect(panHandlers.active).toHaveBeenLastCalledWith(
```

```
        expect.objectContaining({ translationX: 20 })
      );
    });
```

getByGestureTestId(testID)

Returns opaque data type associated with gesture. Gesture is found via `testID` attribute in rendered components (see `withTestID` method).

Arguments

`testID`

String identifying gesture.

Notes

`testID` must be unique among components rendered in test.

Example

See above example for `fireGestureHandler`.