Version: 3.x

runOnJS

`runOnJS` lets you asynchronously run non-workletized functions that couldn't otherwise run on the UI thread. This applies to most external libraries as they don't have their functions marked with `"worklet";` directive.

`runOnJS` is usually used to update React state either on animation finish or conditionally within a gesture.

Reference

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

function App() {
  // While on the UI thread
  runOnJS(navigation.goBack)();
}
```

Type definitions

Arguments

fn

A reference to a function you want to execute on the JavaScript thread from the UI thread. Arguments to your function have to be passed to the function returned from `runOnJS` i.e. `runOnJS(setValue)(10);`.

Returns

`runOnJS` returns a function that accepts arguments for the function passed as the first argument. This function can be safely executed on the UI thread.

> **INFO**
>
> Don't forget to call the function returned from `runOnJS`.

Example

Preview Code

CLICK ME

Remarks

- Functions passed to `runOnJS` must be defined in the JavaScript thread scope, i.e. in the component body or the global scope. This code won't work because `myFunction` is defined in the `withTiming` callback, which is only executed in the UI thread:

```
withTiming(0, {}, () => {
  // myFunction is defined on the UI thread
  const myFunction = () => {
    // ...
  };
```

```
  runOnJS(myFunction)(); // 💥
});
```

- It's a common mistake to execute function inside of runOnJS like this: ~~`runOnJS(setValue(10))()`~~. Here, the correct usage would be `runOnJS(setValue)(10)`.
- It's safe to run functions via `runOnJS` on the **JavaScript thread**, as this has no effect.

Platform compatibility

Android	iOS	Web
✅	✅	✅

✎ Edit this page