**Version: 3.x**

# runOnUI

`runOnUI` lets you asynchronously run workletized functions on the UI thread.

Most commonly used either with an `useEffect` to start an animation on component mount/unmount or with `measure` and `scrollTo` functions which have implementations only on the UI thread.

## Reference

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

function App() {
  // E.g. in event handler or in an effect
  runOnUI((greeting) => {
    console.log(`${greeting} from the UI thread`);
  })('Howdy');

  // ...
}
```

Type definitions

### Arguments

#### fn

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

### Returns

`runOnUI` returns a function that accepts arguments for the function passed as the first argument.

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

## Example

## Remarks

- When implementing your animations you should first reach for more general solutions such as `useDerivedValue`, `useAnimatedReaction` or running code in gesture callbacks and only use `runOnUI` after you've tried other methods.
- It's a common mistake to execute function inside of runOnUI like this: ~~`runOnUI(myWorklet(10))()`~~. Here, the correct usage would be `runOnUI(myWorklet)(10)`.

- The callback passed as the argument is automatically **workletized** and ready to be run on the **UI thread**.
- Make sure not to execute `runOnUI` on the UI thread as this will result in an error.

## Platform compatibility

| Android | iOS | Web |
|---|---|---|
| ✓ | ✓ | ✓ |

Edit this page