Version: 6.x

Call a function when focused screen changes

In this guide we will call a function or render something on screen focusing. This is useful for making additional API calls when a user revisits a particular screen in a Tab Navigator, or to track user events as they tap around our app.

There are multiple approaches available to us:

1. Listening to the `'focus'` event with an event listener.
2. Using the `useFocusEffect` hook provided by react-navigation.
3. Using the `useIsFocused` hook provided by react-navigation.

Triggering an action with a `'focus'` event listener

We can also listen to the `'focus'` event with an event listener. After setting up an event listener, we must also stop listening to the event when the screen is unmounted.

With this approach, we will only be able to call an action when the screen focuses. This is useful for performing an action such as logging the screen view for analytics.

Example:

```
import * as React from 'react';
import { View } from 'react-native';

function ProfileScreen({ navigation }) {
  React.useEffect(() => {
    const unsubscribe = navigation.addListener('focus', () => {
      // The screen is focused
      // Call any action
    });

    // Return the function to unsubscribe from the event so it gets removed on
unmount
    return unsubscribe;
```

```
  }, [navigation]);

  return <View />;
}
```

Try this example on Snack

See the navigation events guide for more details on the event listener API.

In most cases, it's recommended to use the `useFocusEffect` hook instead of adding the listener manually. See below for details.

Triggering an action with the `useFocusEffect` hook

React Navigation provides a hook that runs an effect when the screen comes into focus and cleans it up when it goes out of focus. This is useful for cases such as adding event listeners, for fetching data with an API call when a screen becomes focused, or any other action that needs to happen once the screen comes into view.

This is particularly handy when we are trying to stop something when the page is unfocused, like stopping a video or audio file from playing, or stopping the tracking of a user's location.

```
import { useFocusEffect } from '@react-navigation/native';

function Profile({ userId }) {
  const [user, setUser] = React.useState(null);

  useFocusEffect(
    React.useCallback(() => {
      const unsubscribe = API.subscribe(userId, user => setUser(data));

      return () => unsubscribe();
    }, [userId])
  );

  return <ProfileContent user={user} />;
}
```

Try this example on Snack

See the `useFocusEffect` documentation for more details.

Re-rendering screen with the `useIsFocused` hook

React Navigation provides a hook that returns a boolean indicating whether the screen is focused or not.

The hook will return `true` when the screen is focused and `false` when our component is no longer focused. This enables us to render something conditionally based on whether the user is on the screen or not.

The `useIsFocused` hook will cause our component to re-render when we focus and unfocus a screen. Using this hook component may introduce unnecessary component re-renders as a screen comes in and out of focus. This could cause issues depending on the type of action we're calling on focusing. Hence we recommend to use this hook only if you need to trigger a re-render. For side-effects such as subscribing to events or fetching data, use the methods described above.

```
import * as React from 'react';
import { Text } from 'react-native';
import { useIsFocused } from '@react-navigation/native';

function Profile() {
  // This hook returns `true` if the screen is focused, `false` otherwise
  const isFocused = useIsFocused();

  return <Text>{isFocused ? 'focused' : 'unfocused'}</Text>;
}
```

Try this example on Snack

This example is also documented in the `useIsFocused` API documentation.

Edit this page