RootTag

`RootTag` is an opaque identifier assigned to the native root view of your React Native surface — i.e. the `ReactRootView` or `RCTRootView` instance for Android or iOS respectively. In short, it is a surface identifier.

When to use a RootTag?

For most React Native developers, you likely won't need to deal with `RootTag`s.

`RootTag`s are useful for when an app renders **multiple React Native root views** and you need to handle native API calls differently depending on the surface. An example of this is when an app is using native navigation and each screen is a separate React Native root view.

In native navigation, every React Native root view is rendered in a platform's navigation view (e.g., `Activity` for Android, `UINavigationViewController` for iOS). By this, you are able to leverage the navigation paradigms of the platform such as native look and feel and navigation transitions. The functionality to interact with the native navigation APIs can be exposed to React Native via a native module.

For example, to update the title bar of a screen, you would call the navigation module's API `setTitle("Updated Title")`, but it would need to know which screen in the stack to update. A `RootTag` is necessary here to identify the root view and its hosting container.

Another use case for `RootTag` is when your app needs to attribute a certain JavaScript call to native based on its originating root view. A `RootTag` is necessary to differentiate the source of the call from different surfaces.

How to access the RootTag... if you need it

In versions 0.65 and below, RootTag is accessed via a legacy context. To prepare React Native for Concurrent features coming in React 18 and beyond, we are migrating to the

latest Context API via `RootTagContext` in 0.66. Version 0.65 supports both the legacy context and the recommended `RootTagContext` to allow developers time to migrate their call-sites. See the breaking changes summary.

How to access `RootTag` via the `RootTagContext`.

```
import {RootTagContext} from 'react-native';
import NativeAnalytics from 'native-analytics';
import NativeNavigation from 'native-navigation';

function ScreenA() {
  const rootTag = useContext(RootTagContext);

  const updateTitle = title => {
    NativeNavigation.setTitle(rootTag, title);
  };

  const handleOneEvent = () => {
    NativeAnalytics.logEvent(rootTag, 'one_event');
  };

  // ...
}

class ScreenB extends React.Component {
  static contextType: typeof RootTagContext = RootTagContext;

  updateTitle(title) {
    NativeNavigation.setTitle(this.context, title);
  }

  handleOneEvent() {
    NativeAnalytics.logEvent(this.context, 'one_event');
  }

  // ...
}
```

Learn more about the Context API for classes and hooks from the React docs.

Breaking Change in 0.65

`RootTagContext` was formerly named `unstable_RootTagContext` and changed to `RootTagContext` in 0.65. Please update any usages of `unstable_RootTagContext` in your codebase.

Breaking Change in 0.66

The legacy context access to `RootTag` will be removed and replaced by `RootTagContext`. Beginning in 0.65, we encourage developers to proactively migrate `RootTag` accesses to `RootTagContext`.

Future Plans

With the new React Native architecture progressing, there will be future iterations to `RootTag`, with the intention to keep the `RootTag` type opaque and prevent thrash in React Native codebases. Please do not rely on the fact that RootTag currently aliases to a number! If your app relies on RootTags, keep an eye on our version change logs, which you can find here.

Is this page useful? 

Edit this page

*Last updated on **Jun 21, 2023***