Version: 6.x

State persistence

You might want to save the user's location in the app, so that they are immediately returned to the same location after the app is restarted.

This is especially valuable during development because it allows the developer to stay on the same screen when they refresh the app.

Usage

To be able to persist the navigation state, we can use the `onStateChange` and `initialState` props of the container.

- `onStateChange` - This prop notifies us of any state changes. We can persist the state in this callback.
- `initialState` - This prop allows us to pass an initial state to use for navigation state. We can pass the restored state in this prop.

```
import * as React from 'react';
import { Linking, Platform } from 'react-native';
import AsyncStorage from '@react-native-async-storage/async-storage';
import { NavigationContainer } from '@react-navigation/native';

const PERSISTENCE_KEY = 'NAVIGATION_STATE_V1';

export default function App() {
  const [isReady, setIsReady] = React.useState(false);
  const [initialState, setInitialState] = React.useState();

  React.useEffect(() => {
    const restoreState = async () => {
      try {
        const initialUrl = await Linking.getInitialURL();

        if (Platform.OS !== 'web' && initialUrl == null) {
```

```
          // Only restore state if there's no deep link and we're not on web
          const savedStateString = await AsyncStorage.getItem(PERSISTENCE_KEY);
          const state = savedStateString ? JSON.parse(savedStateString) :
undefined;

          if (state !== undefined) {
            setInitialState(state);
          }
        }
      } finally {
        setIsReady(true);
      }
    };

    if (!isReady) {
      restoreState();
    }
  }, [isReady]);

  if (!isReady) {
    return null;
  }

  return (
    <NavigationContainer
      initialState={initialState}
      onStateChange={(state) =>
        AsyncStorage.setItem(PERSISTENCE_KEY, JSON.stringify(state))
      }
    >
      {/* ... */}
    </NavigationContainer>
  );
}
```

Try this example on Snack

Development Mode

This feature is particularly useful in development mode. You can enable it selectively using the following approach:

```
const [isReady, setIsReady] = React.useState(__DEV__ ? false : true);
```

While it can be used for production as well, use it with caution as it can make the app unusable if the app is crashing on a particular screen - as the user will still be on the same screen after restarting.

Loading View

Because the state is restored asynchronously, the app must render an empty/loading view for a moment before we have the initial state. To handle this, we can return a loading view when `isReady` is `false`:

```
if (!isReady) {
  return <ActivityIndicator />;
}
```

Warning: Serializable State

Each param, route, and navigation state must be fully serializable for this feature to work. Typically, you would serialize the state as a JSON string. This means that your routes and params must contain no functions, class instances, or recursive data structures. React Navigation already warns you during development if it encounters non-serializable data, so watch out for the warning if you plan to persist navigation state.

You can modify the initial state object before passing it to container, but note that if your `initialState` isn't a valid navigation state, React Navigation may not be able to handle the situation gracefully.

Edit this page