



Navigators

Material Bottom Tabs

Version: 6.x

# Material Bottom Tabs Navigator

The `material-bottom-tabs` navigator is moved to `react-native-paper`. Refer to `react-native-paper`'s documentation for API and usage. For any issues with the navigator, please open an issue in `react-native-paper`'s repository.

A material-design themed tab bar on the bottom of the screen that lets you switch between different routes with animation. Routes are lazily initialized - their screen components are not mounted until they are first focused.

This wraps the `BottomNavigation` component from `react-native-paper`. If you configure the Babel plugin, it won't include the whole `react-native-paper` library in your bundle.



## Installation

To use this navigator, ensure that you have `@react-navigation/native` and its dependencies (follow this guide), then install `@react-navigation/material-bottom-tabs`:

**npm**    **Yarn**

```
npm install @react-navigation/material-bottom-tabs react-native-paper react-native-vector-icons
```

This API also requires that you install `react-native-vector-icons`! If you are using Expo managed workflow, it will work without any extra steps. Otherwise, follow these installation instructions.

To use this tab navigator, import it from `@react-navigation/material-bottom-tabs`

## API Definition

💡 If you encounter any bugs while using `createMaterialBottomTabNavigator`, please open issues on `react-native-paper` rather than the `react-navigation` repository!

To use this tab navigator, import it from `@react-navigation/material-bottom-tabs`:

```
import { createMaterialBottomTabNavigator } from '@react-navigation/material-bottom-tabs';

const Tab = createMaterialBottomTabNavigator();

function MyTabs() {
  return (
    <Tab.Navigator>
      <Tab.Screen name="Home" component={HomeScreen} />
      <Tab.Screen name="Settings" component={SettingsScreen} />
    </Tab.Navigator>
  );
}
```

Try this example on Snack [↗](#)

For a complete usage guide please visit [Tab Navigation](#)

## RouteConfigs

The route configs object is a mapping from route name to a route config.

## Props

The `Tab.Navigator` component accepts following props:

**id**

Optional unique ID for the navigator. This can be used with `navigation.getParent` to refer to this navigator in a child navigator.

**initialRouteName**

The name of the route to render on first load of the navigator.

**screenOptions**

Default options to use for the screens in the navigator.

**backBehavior**

This controls what happens when `goBack` is called in the navigator. This includes pressing the device's back button or back gesture on Android.

It supports the following values:

- `firstRoute` - return to the first screen defined in the navigator (default)
- `initialRoute` - return to initial screen passed in `initialRouteName` prop, if not passed, defaults to the first screen
- `order` - return to screen defined before the focused screen
- `history` - return to last visited screen in the navigator; if the same screen is visited multiple times, the older entries are dropped from the history
- `none` - do not handle back button

**shifting**

Whether the shifting style is used, the active tab icon shifts up to show the label and the inactive tabs won't have a label.

By default, this is `true` when you have more than 3 tabs. Pass `shifting={false}` to explicitly disable this animation, or `shifting={true}` to always use this animation.

**labeled**

Whether to show labels in tabs. When `false`, only icons will be displayed.

**activeColor**

Custom color for icon and label in the active tab.

**inactiveColor**

Custom color for icon and label in the inactive tab.

**barStyle**

Style for the bottom navigation bar. You can pass custom background color here:

```
<Tab.Navigator
  initialRouteName="Home"
  activeColor="#f0edf6"
  inactiveColor="#3e2465"
  barStyle={{ backgroundColor: '#694fad' }}
>
  { /* ... */ }
</Tab.Navigator>
```

Try this example on Snack [↗](#)

If you have a translucent navigation bar on Android, you can also set a bottom padding here:

```
<Tab.Navigator
  initialRouteName="Home"
  activeColor="#f0edf6"
  inactiveColor="#3e2465"
  barStyle={{ paddingBottom: 48 }}
>
  { /* ... */ }
</Tab.Navigator>
```

## Options

The following [options](#) can be used to configure the screens in the navigator:

**title**

Generic title that can be used as a fallback for `headerTitle` and `tabBarLabel`.

### `tabBarIcon`

Function that given `{ focused: boolean, color: string }` returns a `React.Node`, to display in the tab bar.

### `tabBarColor`

Color for the tab bar when the tab corresponding to the screen is active. Used for the ripple effect. This is only supported when `shifting` is `true`.

### `tabBarLabel`

Title string of a tab displayed in the tab bar. When undefined, scene `title` is used. To hide, see `labeled` option in the previous section.

### `tabBarBadge`

Badge to show on the tab icon, can be `true` to show a dot, `string` or `number` to show text.

### `tabBarAccessibilityLabel`

Accessibility label for the tab button. This is read by the screen reader when the user taps the tab. It's recommended to set this if you don't have a label for the tab.

### `tabBarTestID`

ID to locate this tab button in tests.

## Events

The navigator can [emit events](#) on certain actions. Supported events are:

### `tabPress`

This event is fired when the user presses the tab button for the current screen in the tab bar. By default a tab press does several things:

- If the tab is not focused, tab press will focus that tab

- If the tab is already focused:
  - If the screen for the tab renders a scroll view, you can use `useScrollToTop` to scroll it to top
  - If the screen for the tab renders a stack navigator, a `popToTop` action is performed on the stack

To prevent the default behavior, you can call `event.preventDefault`:

```
React.useEffect(() => {  
  const unsubscribe = navigation.addListener('tabPress', (e) => {  
    // Prevent default behavior  
  
    e.preventDefault();  
    // Do something manually  
    // ...  
  });  
  
  return unsubscribe;  
}, [navigation]);
```

Try this example on Snack [↗](#)

## Helpers

The tab navigator adds the following methods to the navigation prop:

### `jumpTo`

Navigates to an existing screen in the tab navigator. The method accepts following arguments:

- `name` - *string* - Name of the route to jump to.
- `params` - *object* - Screen params to pass to the destination route.

```
navigation.jumpTo('Profile', { name: 'Michaś' });
```

Try this example on Snack [↗](#)

## Example

```
import { createMaterialBottomTabNavigator } from '@react-navigation/material-  
bottom-tabs';  
import MaterialCommunityIcons from 'react-native-vector-  
icons/MaterialCommunityIcons';  
  
const Tab = createMaterialBottomTabNavigator();  
  
function MyTabs() {  
  return (  
    <Tab.Navigator  
      initialRouteName="Feed"  
      activeColor="#e91e63"  
      barStyle={{ backgroundColor: 'tomato' }}  
    >  
      <Tab.Screen  
        name="Feed"  
        component={Feed}  
        options={{  
          tabBarLabel: 'Home',  
          tabBarIcon: ({ color }) => (  
            <MaterialCommunityIcons name="home" color={color} size={26} />  
          ),  
        }}  
      />  
      <Tab.Screen  
        name="Notifications"  
        component={Notifications}  
        options={{  
          tabBarLabel: 'Updates',  
          tabBarIcon: ({ color }) => (  
            <MaterialCommunityIcons name="bell" color={color} size={26} />  
          ),  
        }}  
      />  
      <Tab.Screen  
        name="Profile"  
        component={Profile}  
        options={{  
          tabBarLabel: 'Profile',  
          tabBarIcon: ({ color }) => (  
            <MaterialCommunityIcons name="account" color={color} size={26} />  
          ),  
        }}  
    >  
  )  
}
```

```
    />  
  </Tab.Navigator>  
);  
}
```

Try this example on Snack [↗](#)

## Using with `react-native-paper` (optional)

You can use the theming support in `react-native-paper` to customize the material bottom navigation by wrapping your app in `Provider` from `react-native-paper`. A common use case for this can be to customize the background color for the screens when your app has a dark theme. Follow the instructions on `react-native-paper`'s documentation to learn how to customize the theme.

[✎](#) Edit this page