Displaying a Notification

Display your first cross-platform notification with Notifee.

← Basic Usage Events →

Notification Title & Body

Notifications in their basic form contain a title and a main body of text. Let's go ahead and display a basic notification inside of a React Native app when a button is pressed.

Import the library and display a basic view with a Button:

```
import React from 'react';
import { View, Button } from 'react-native';
import notifee from '@notifee/react-native';

function Screen() {
  return (
    <View>
      <Button title="Display Notification" onPress={() => {}} />
    </View>
  );
}
```

Now the button is in place, create a function to handle the button press and display a notification:

```
function Screen() {
  async function onDisplayNotification() {
    // Request permissions (required for iOS)
    await notifee.requestPermission()
```

```
      name: 'Default Channel',
    });

    // Display a notification
    await notifee.displayNotification({
      title: 'Notification Title',
      body: 'Main body content of the notification',
      android: {
        channelId,
        smallIcon: 'name-of-a-small-icon', // optional, defaults to 'ic_launcher'.
        // pressAction is needed if you want the notification to open the app when
        pressAction: {
          id: 'default',
        },
      },
    });
  }

  return (
    <View>
      <Button title="Display Notification" onPress={() => onDisplayNotification()}
    </View>
  );
}
```

When the button is pressed, we perform three tasks: requesting permission, creating a channel & displaying a notification.

Requesting permission is required for iOS as notifications are disabled by default. You need to ask the user to enable notifications by calling `requestPermission` before displaying any notifications.

> To learn more about requesting permissions, view the iOS Permissions documentation

updated each time a call to `createChannel` is performed, so it is safe to keep calling this method.

Once the channel has been created, the `displayNotification` method is called passing in a `title` and `body`. The required `channelId` is also passed inside of the `android` property object to assign the notification to the channel. On iOS platform, the call to `createChannel` resolves instantly & gracefully (iOS has no concept of a channel), then calls `displayNotification`.

In addition to channels, small icons are an Android-only concept and are required to display a notification. If no icon is specified, Notifee will fallback to the default launcher icon (`ic_launcher`).

> To learn more about channels and small icons, view the Android Channels and Small Icon documentation.

Go ahead and press the button! A notification icon will appear on your device and will be visible when pulling down the notification shade.

Updating a notification

When notifications are created, a random unique ID is assigned to them. These IDs can later be used to update any notification which is still present on the user's device. It is also possible to provide a custom notification ID by passing a string value to the `id` field inside of `displayNotification`:

```
async function onDisplayNotification() {
  const channelId = await notifee.createChannel({
    id: 'default',
    name: 'Default Channel',
  });

  // Required for iOS
  // See https://notifee.app/react-native/docs/ios/permissions
  await notifee.requestPermission();
```

```
    title: 'Notification Title',
    body: 'Main body content of the notification',
    android: {
      channelId,
    },
  });

  // Sometime later...
  await notifee.displayNotification({
    id: '123',
    title: 'Updated Notification Title',
    body: 'Updated main body content of the notification',
    android: {
      channelId,
    },
  });
}
```

When updating a notification, you must pass a brand new notification options payload to **`displayNotification`**, rather than the specific property you want to update.

Cancelling a notification

There may be situations whereby you wish to cancel and remove the notification from the device, for example the notification is no longer relevant or has expired.

To cancel a notification, the **`cancelNotification`** method can be called with the unique notification ID which was used when created:

```
async function cancel(notificationId) {
  await notifee.cancelNotification(notificationId);
}
```

It is also possible to cancel a notification with a tag:

Please note that on Android, a cancelled notification *may* briefly show up in future calls to `getDisplayedNotifications` . This appears to be a limitation of the platform in that cancel seems to be asynchronous but return immediately. See the related issue for more information.

← Basic Usage Events →