

# Native Modules Intro

## ! INFO

Native Module and Native Components are our stable technologies used by the legacy architecture. They will be deprecated in the future when the New Architecture will be stable. The New Architecture uses [Turbo Native Module](#) and [Fabric Native Components](#) to achieve similar results.

Sometimes a React Native app needs to access a native platform API that is not available by default in JavaScript, for example the native APIs to access Apple or Google Pay. Maybe you want to reuse some existing Objective-C, Swift, Java or C++ libraries without having to reimplement it in JavaScript, or write some high performance, multi-threaded code for things like image processing.

The NativeModule system exposes instances of Java/Objective-C/C++ (native) classes to JavaScript (JS) as JS objects, thereby allowing you to execute arbitrary native code from within JS. While we don't expect this feature to be part of the usual development process, it is essential that it exists. If React Native doesn't export a native API that your JS app needs you should be able to export it yourself!

## Native Module Setup

There are two ways to write a native module for your React Native application:

1. Directly within your React Native application's iOS/Android projects
2. As a NPM package that can be installed as a dependency by your/other React Native applications

This guide will first walk you through implementing a native module directly within a React Native application. However the native module you build in the following guide can be distributed as an NPM package. Check out the [Setting Up a Native Module as a NPM Package](#) guide if you are interested in doing so.


# Getting Started

In the following sections we will walk you through guides on how to build a native module directly within a React Native application. As a prerequisite, you will need a React Native application to work within. You can follow the steps [here](#) to setup a React Native application if you do not already have one.

Imagine that you want to access the iOS/Android native calendar APIs from JavaScript within a React Native application in order to create calendar events. React Native does not expose a JavaScript API to communicate with the native calendar libraries. However, through native modules, you can write native code that communicates with native calendar APIs. Then you can invoke that native code through JavaScript in your React Native application.

In the following sections you will create such a Calendar native module for both [Android](#) and [iOS](#).

Is this page useful?  

 Edit this page

Last updated on **Jun 21, 2023**