Swift-Compatible Native Module Registration in React Native (iOS)

# 📍 Background

React Native’s New Architecture enables TurboModules and Fabric for improved performance. When building native modules for iOS:  
- Objective-C has traditionally been the standard.  
- Now, Swift is increasingly popular due to modern syntax and better integration with Apple’s ecosystem.  
  
Swift modules can now be registered as TurboModules, provided they are correctly exposed to React Native.

# 🔧 Setting Up a Swift Native Module

✅ Step 1: Create a Swift File  
In your iOS project (Xcode), right-click on your project > New File > Swift File.  
- Name it `MyModule.swift`  
- Xcode will prompt you to create a Bridging Header — accept it.  
  
✅ Step 2: Enable Swift Support in CocoaPods  
In your Podfile, make sure the platform and Swift version are compatible:  
  
```ruby  
platform :ios, '13.0'  
  
post\_install do |installer|  
 installer.pods\_project.targets.each do |target|  
 target.build\_configurations.each do |config|  
 config.build\_settings['SWIFT\_VERSION'] = '5.0'  
 end  
 end  
end  
```  
  
Then run:  
```bash  
cd ios  
pod install  
```

# 🚀 Example: Swift TurboModule (New Architecture)

MyModule.swift:  
  
```swift  
import Foundation  
  
@objc(MyModule)  
class MyModule: NSObject {  
 @objc  
 func greet(\_ name: String, resolve: @escaping RCTPromiseResolveBlock, reject: @escaping RCTPromiseRejectBlock) {  
 resolve("Hello, \(name) 👋")  
 }  
}  
```

# 🧾 Register with React Native Codegen

For TurboModule support, create a MyModule.h and MyModuleSpec interface.  
  
MyModule.h:  
  
```objc  
#import <React/RCTBridgeModule.h>  
  
@interface RCT\_EXTERN\_MODULE(MyModule, NSObject)  
  
RCT\_EXTERN\_METHOD(greet:(NSString \*)name  
 resolve:(RCTPromiseResolveBlock)resolve  
 reject:(RCTPromiseRejectBlock)reject)  
  
@end  
```

MyModule.ts (JS/TS interface for codegen):  
  
```ts  
import type { TurboModule } from 'react-native';  
import { TurboModuleRegistry } from 'react-native';  
  
export interface Spec extends TurboModule {  
 greet(name: string): Promise<string>;  
}  
  
export default TurboModuleRegistry.get<Spec>('MyModule');  
```

# 🧩 Integrating with JS

App.tsx:  
  
```tsx  
import MyModule from './MyModule';  
  
useEffect(() => {  
 MyModule.greet("React Native Dev").then((msg) => {  
 console.log(msg); // Hello, React Native Dev 👋  
 });  
}, []);  
```

# 🧪 Build & Run

1. Run `npx react-native run-ios`  
2. Check that the module loads successfully.  
3. You should see the greeting message in the console.

# ✅ Best Practices

| Practice | Description |  
|----------------------|----------------------------------------|  
| `@objc` | Required to expose Swift classes/methods |  
| `RCT\_EXTERN\_MODULE` | Used to register Swift methods with RN |  
| `RCTPromise\*` types | Promise-based methods |  
| Bridging Header | Auto-generated when adding first Swift file |  
| TurboModule Codegen | Enables native modules without JS bridge |

# 📚 Resources

- React Native Docs – Native Modules (iOS): https://reactnative.dev/docs/native-modules-intro  
- Expo + Native Modules: https://docs.expo.dev/modules/intro/  
- Apple Swift Documentation: https://developer.apple.com/swift/