

A M Y T A X I S T O R Y

# THE GEARS PROJECT





# Rubén Sospedra

Javascript hacker  
@sospedra\_r



# Senior software engineer at **mytaxi**

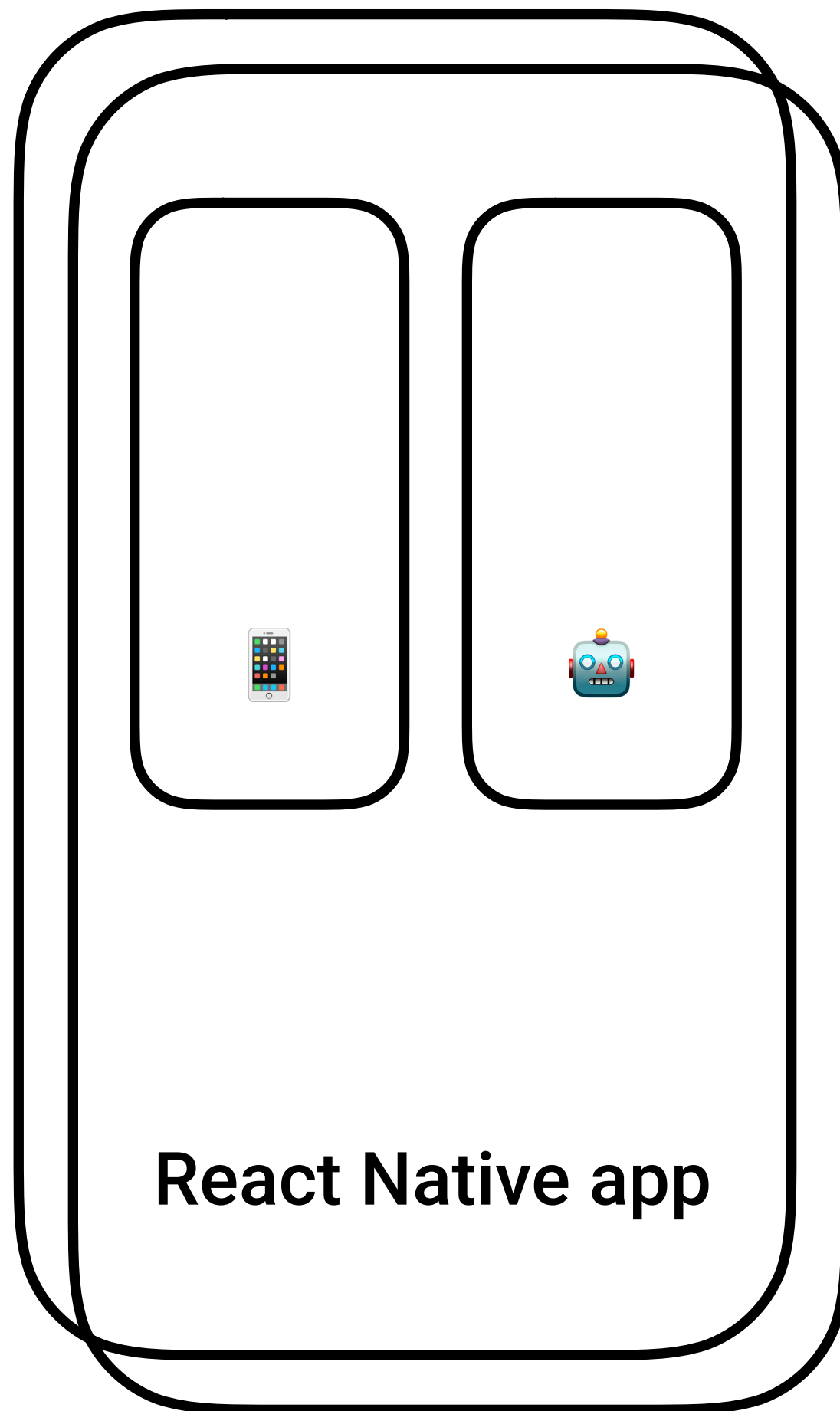


# What's gears?

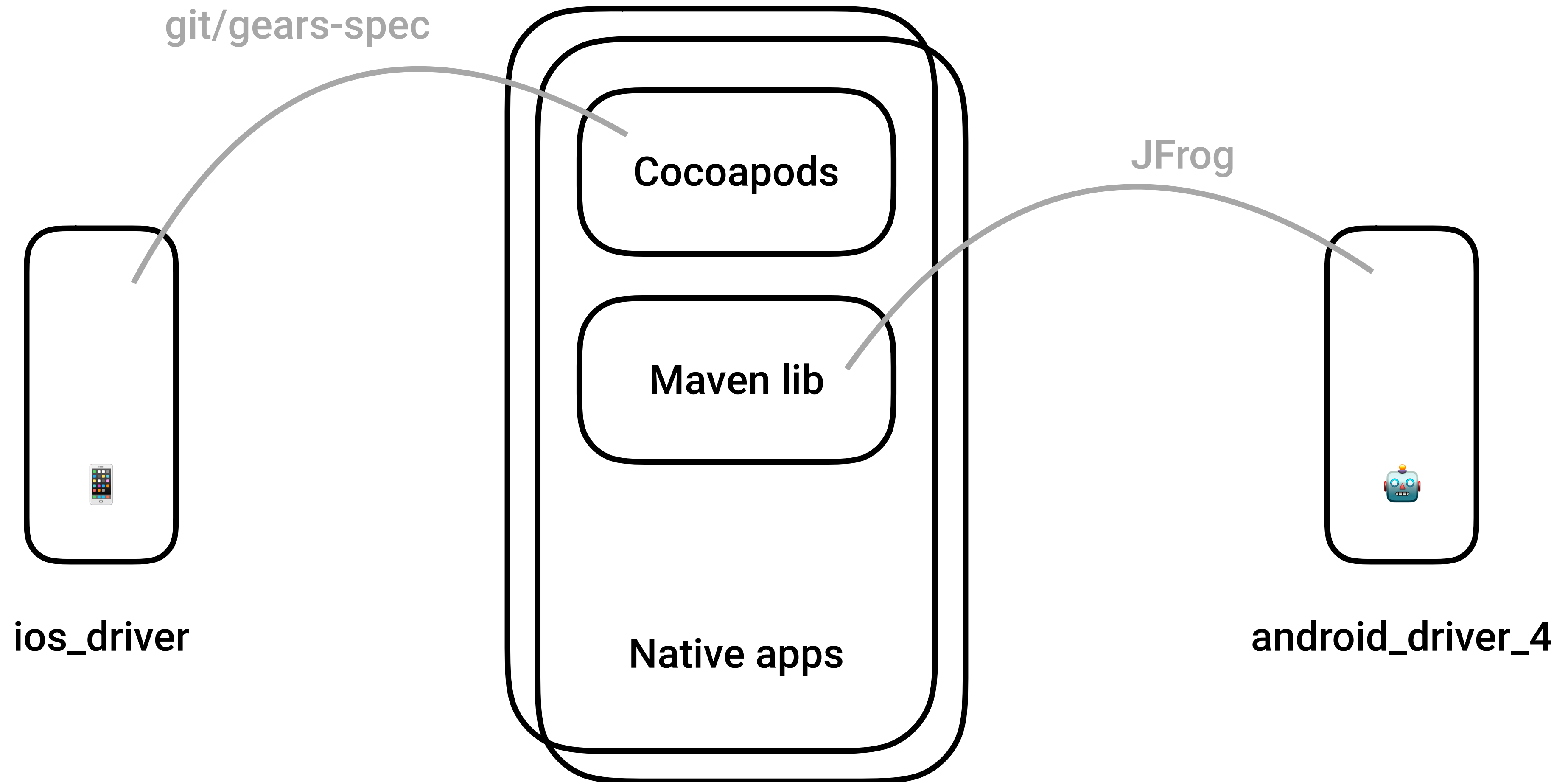
- ① It's a way to organise and ship a **react-native** project
- ② Where the outputs are **libraries** and not applications
- ③ And the focus is on the **native communication**

All started with a  
problem

tribe responsibility vs product ownership



So we tried  
something







myt-frontend-components / gears-specs

## Source

master ▾



[gears-specs](#) / **version.txt**

Source view

Diff to previous

History ▾

Contributors ▾

```
1 # Required sources
2 source 'ssh://git@stash.intapps.it:7999/fec/gears-specs.git'
3 source 'https://github.com/CocoaPods/Specs.git'
4
5 # Fix yoga import issue #711
6 plugin 'cocoapods-fix-react-native'
7
8 # Last version of Gears pods
9 pod 'Gears', '~> 1.1.6'
10 pod 'yoga', :podspec => '~/cocoapods/repos/intapps-gears-specs/yoga/0.54.4/yoga.podspec.json'
```



Bitbucket

Projects

Repositories ▾

People ▾



myt-frontend-components / gears



## Source

master ▾



[gears](#) / [ios](#) / **gears-Bridging-Header.h**

Source view

Diff to previous

History ▾

Contributors ▾

```
1 //  
2 // Use this file to import your target's public headers that you would like to  
3 //  
4  
5 #import <Gears/NativeToReactViewController.h>  
6 #import <Gears/RenderController.h>
```



myt-frontend-components / gears

## Source



master ▾



[gears](#) / [android](#) / **version.txt**

Source view

Diff to previous

History ▾

Contributors ▾

```
1 # Required sources
2 compile "com.mytaxi.android:gears:1.1.6"
3 compile "com.facebook.react:react-native:0.54.4"
4
```





myt-frontend-components / gears

## Source

master ▾



[gears](#) / [android](#) / [app](#) / [src](#) / [main](#) / [java](#) / [com](#) / [gearsdemo](#) / **MainActivity.java**

Source view

Diff to previous

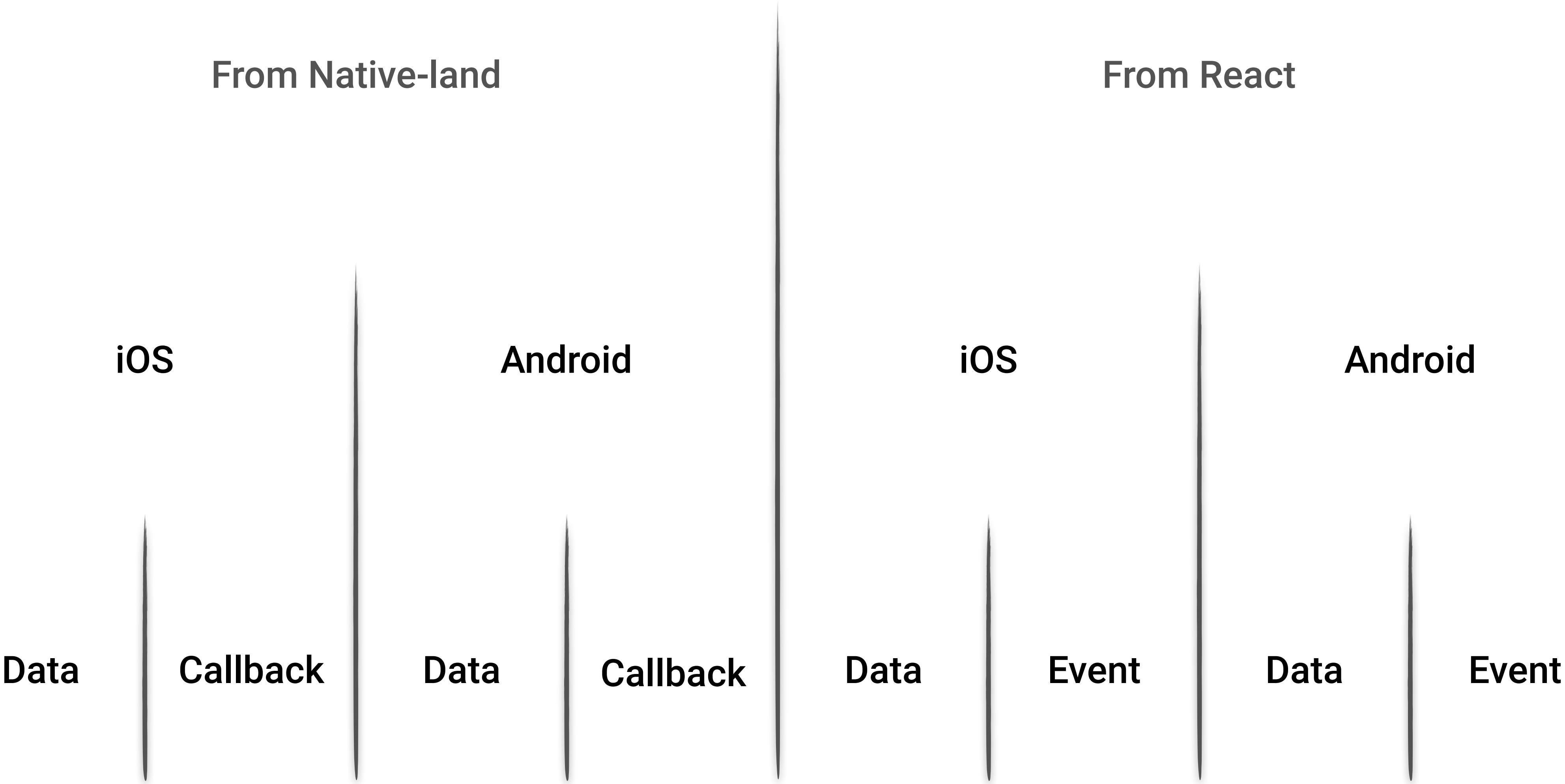
History ▾

Contributors ▾

```
1 package com.gearsdemo;
2
3 import android.app.Activity;
4 import android.content.Intent;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Button;
8
9 import com.mytaxi.gears.cases.ReactToNative;
10 import com.mytaxi.gears.cases.NativeToReact;
11 import com.gearsdemo.GRSPlayground;
12
13 public class MainActivity extends Activity {
```

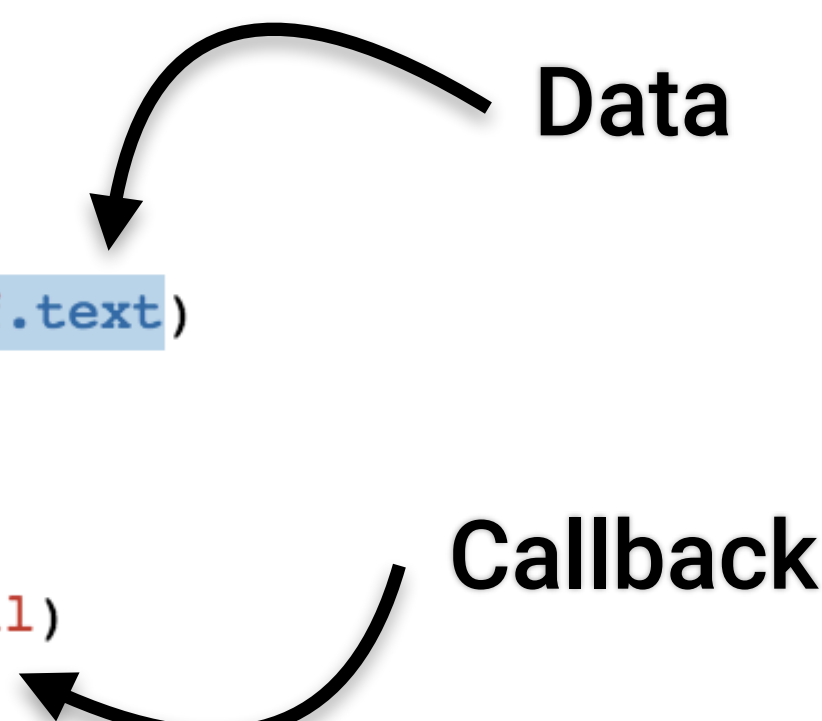
And we found a  
solution





# From Native in iOS

```
8 import Foundation
9 import UIKit
10
11 class SwiftNativeToReact: UIViewController, UITextFieldDelegate, NativeToReactViewControllerDelegate {
12     var text: String = ""
13
14     @IBAction func reactNativeButtonPressed(_ sender: Any) {
15         // Mount
16         let nativeToReactVC = NativeToReactViewController(word: self.text)
17         // Add delegate
18         nativeToReactVC.delegate = self
19         // Render
20         self.present(nativeToReactVC, animated: true, completion: nil)
21     }
22
23     func textField(_ textField: UITextField, shouldChangeCharactersIn range: NSRange, replacementString string: String)
24         // missing the last char
25         self.text = textField.text ?? ""
26         return true
27     }
28
29     func native2ReactDismiss(_ nativeToReactViewController: NativeToReactViewController) {
30         nativeToReactViewController.dismiss(animated: true, completion: nil)
31     }
32 }
```



The diagram illustrates the data flow and callback mechanism in the provided Swift code. A curved arrow labeled "Data" points from the `self.text` property access on line 16 to the `word` parameter of the `NativeToReactViewController` constructor. Another curved arrow labeled "Callback" points from the `self.present` call on line 20 to the `native2ReactDismiss` method on line 29, which is the delegate method for dismissing the presented view controller.

# From Native in iOS **data implementation**

```
30 - (instancetype)initWithWord:(NSString *)word {
31     if ((self = [super initWithNibName:nil bundle:nil])) {
32         self.rootView = nil;
33         self.reactInitialProperties = @{
34             // Add the ReactNative props you need here
35             @"word": word
36         };
37     }
38     return self;
39 }
```

```
57 // Render the React-Native component using the shared RCTBridge
58 self.rootView = [[RCTRootView alloc] initWithBridge: [[GearsBridge sharedInstance] bridge]
59     moduleName: NativeToReactModuleName
60     initialProperties: self.reactInitialProperties];
61
```

# From Native in iOS **callback** implementation

```
--  
21 // Expose this macro as a ReactNative.NativeModule.NativeToReact.dismiss in js  
22 // Add as many macros as you need in javascript-land  
23 RCT_EXPORT_METHOD(dismiss) {  
24     // Connect the macro with the ViewController instance through events (notifications)  
25     dispatch_async(dispatch_get_main_queue(), ^{  
26         [[NSNotificationCenter defaultCenter] postNotificationName:NativeToReactEventDismiss object:nil];  
27     });  
28 }  
29  
  
41 - (void)registerHooks {  
42     // Add the hooks to native-land here with the corresponding callback below  
43     [[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(dismissCallback) name:NativeToReactEventDismiss objec  
44 }  
45  
46 - (void)dismissCallback {  
47     if ([self.delegate respondsToSelector:@selector(native2ReactDismiss:)]) {  
48         // Make this "nativeToReactDismiss" method required  
49         [self.delegate native2ReactDismiss:self];  
50     }  
51 }  
52
```



# From Native in Android

```
90
91     @Override
92     protected Bundle getInitialProps() {
93         Bundle initialProps = new Bundle();
94         initialProps.putString("word", "this is a word");
95         return initialProps;
96     }
97
98     @Override
99     protected List<ReactPackage> getPackages() {
100         List<ReactPackage> packages = super.getPackages();
101         packages.add(new ReactPackageWithGearsNativeModule());
102         return packages;
103     }

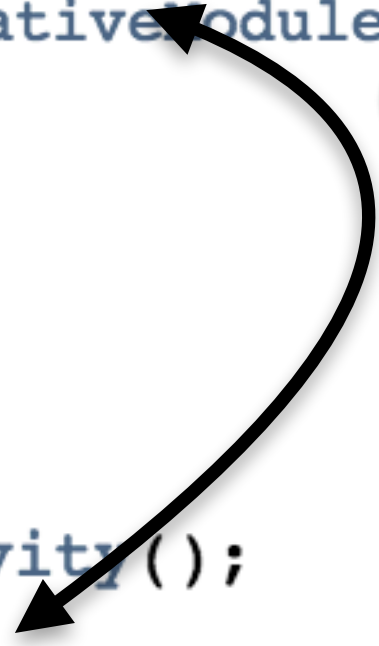
```

**Data**

**Callback**

```
42     @ReactMethod
43     public void dismiss() {
44         Activity currentActivity = getCurrentActivity();
45
46         if (currentActivity != null) {
47             Thread dismissThread = getDismissThread(currentActivity);
48             dismissThread.start();
49         }
50     }

```





# From Native in Android **data implementation**

```
104  @Override
105  protected void onCreate(Bundle savedInstanceState) {
106      super.onCreate(savedInstanceState);
107
108      mReactRootView = new ReactRootView(ReactActivity.this);
109      mReactInstanceManager = ReactInstanceManager.builder()
110          .setApplication(getApplication())
111          .setBundleAssetName("index.bundle")
112          .setJSMainModulePath("index")
113          .addPackages(getPackages())
114          .setUseDeveloperSupport(checkDevMode())
115          .setInitialLifecycleState(LifecycleState.RESUMED)
116          .build();
117
118      mReactRootView.startReactApplication(mReactInstanceManager, getModuleName(), getInitialProps());
119      setContentView(mReactRootView);
120  }
```

# From Native in Android **callback** implementation

```
21
22 class GearsNativeModule extends ReactContextBaseJavaModule {
23     private static final String MODULE_NAME = "NativeToReact";
24     private static final String SHARED_CONSTANT = "SHARED_CONSTANT";
25
26     public GearsNativeModule(ReactApplicationContext reactContext) {
27         super(reactContext);
28     }
29
30     @Override
31     public String getName() {
32         return MODULE_NAME;
33     }
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69 class ReactPackageWithGearsNativeModule implements ReactPackage {
70     @Override
71     public List<ViewManager> createViewManagers(ReactApplicationContext reactContext) {
72         return Collections.emptyList();
73     }
74
75     @Override
76     public List<NativeModule> createNativeModules(ReactApplicationContext reactContext) {
77         List<NativeModule> modules = new ArrayList<>();
78
79         modules.add(new GearsNativeModule(reactContext));
80
81         return modules;
82     }
83 }
```

# From React in **iOS** and **Android**

```
11
12 export const nativeToReactName = 'NativeToReact'
13 const NativeToReactModule = getNativeModule(nativeToReactName)
14
15 /**
16  * Expects that native-land exposes a `NativeToReact` module with a `blackPre
17  */
18 const NativeToReact = (props) => (
19   <View style={styles.container}>
20     <Button
21       onPress={() => NativeToReactModule('dismiss')()}
22       color='#74b9ff'
23       title='Go back to native land'
24     />
25     <Text style={[styles.colorized, styles.welcome]}>
26       This is React Native module: NativeToReact
27     </Text>
28     <Text style={styles.colorized}>
29       Word props from native-land is {props.word}
30     </Text>
31   </View>
32 )
--
```

# With many benefits

- ① Teams are **standalone** and deploy independently
- ② We introduce **web technologies** to the native world
- ③ Product is **not restricted** by technology anymore
- ④ The **learning curve** for new devs is easy

# And some drawbacks

- ① At very few times you still need some **native devs** expertise
- ① **Bundle size** increases at least in 10Mb
- ① It's not a **silver bullet**. No intensive animations. No videogames
- ① ~~The native devs will hate you :P~~



# We're hiring!

## Cities:

Hamburg (HQ)

Berlin

Barcelona

## Jobs:

iOS & Android Developer

Backend & Frontend Developer

QA Engineer

BI & Data Science





@ s o s p e d r a \_ r

# Thanks for coming!

