



RN

Using custom Native Modules

BALA KRISHNA RAGALA



Native Modules

Custom code from native platform

What are Native Modules

- Targeted platform API based code (modules)
- Some Examples
 - ImageProcessing
 - Camera access
 - Database
 - Playing videos and audio
 - OAuth provider for authenticating users

Why Native Modules

- Reusability
- Extensibility
- Multithreaded capabilities
- High performance code

Using Custom Native Modules

Be a consumer :-)

What we will build & Learn

- Build
 - Use react-native-vector-icons module
- Learn
 - Setup the solution to use native modules
 - Automatic linking (demo)
 - Manual linking (reference screens)

Linking Libraries - Facebook says

Linking Libraries

Not every app uses all the native capabilities, and including the code to support all those features would impact the binary size... But we still want to make it easy to add these features whenever you need them.

With that in mind we exposed many of these features as independent static libraries.

For most of the libs it will be as simple as dragging two files, sometimes a third step will be necessary, but no more than that.

Linking Libraries - Types

- Automatic (Easy, hassle free)
- Manual (Cumbersome, execution with discipline)

Automatic Linking

Keep smiling, all will be done with single command

Automatic Linking - Steps

1. `npm install <library-with-native-dependencies> --save`
2. `react-native link (rnpm, react native package manager)`

Here comes much simple combined command

```
react-native install <library-with-native-dependencies>
```

Automatic Linking - In action

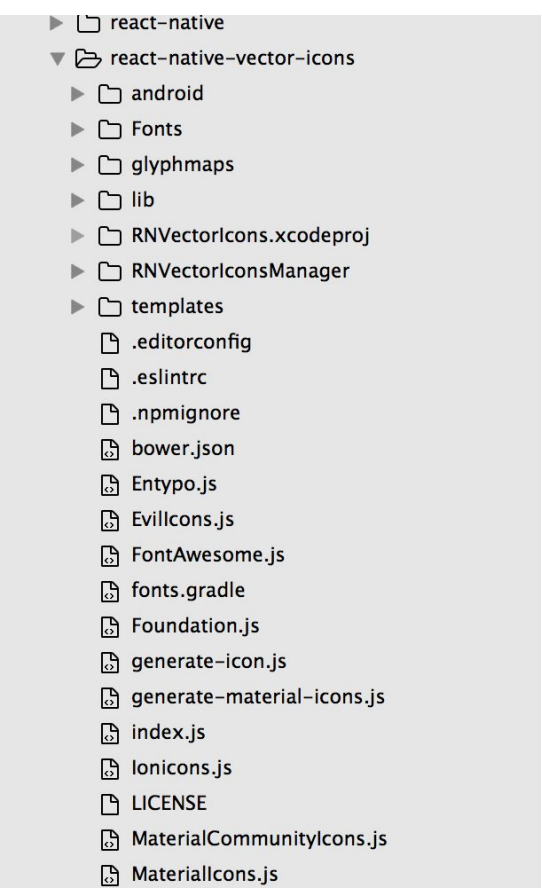
```
MOKSHAs-MacBook-Pro:UsingNativeModules moksha$ react-native install react-native-
Scanning 548 folders for symlinks in /Users/moksha/Zeolearn/react-native/UsingNa
UsingNativeModules@0.0.1 /Users/moksha/Zeolearn/react-native/UsingNativeModules
└─ react-native-vector-icons@4.0.0
```

```
Scanning 549 folders for symlinks in /Users/moksha/Zeolearn/react-native/UsingNa
rnpm-install info Linking react-native-vector-icons android dependency
rnpm-install info Android module react-native-vector-icons has been successfully
rnpm-install info Linking react-native-vector-icons ios dependency
rnpm-install info iOS module react-native-vector-icons has been successfully link
rnpm-install info Linking assets to ios project
rnpm-install WARN ERRGROUP Group 'Resources' does not exist in your XCode projec
for you.
rnpm-install info Linking assets to android project
rnpm-install info Assets have been successfully linked to your project
rnpm-install info Module react-native-vector-icons has been successfully install
```

Automatic Linking - Under the hood - 1

- Installs npm module with JS and native code

```
},  
"dependencies": {  
  "react": "15.4.2",  
  "react-native": "0.42.0",  
  "react-native-vector-icons": "^4.0.0"  
},  
"devDependencies": {
```



```
▶ react-native  
▼ react-native-vector-icons  
  ▶ android  
  ▶ Fonts  
  ▶ glyphmaps  
  ▶ lib  
  ▶ RNVectorIcons.xcodeproj  
  ▶ RNVectorIconsManager  
  ▶ templates  
  .editorconfig  
  .eslintrc  
  .npmignore  
  bower.json  
  Entypo.js  
  EvilIcons.js  
  FontAwesome.js  
  fonts.gradle  
  Foundation.js  
  generate-icon.js  
  generate-material-icons.js  
  index.js  
  Ionicons.js  
  LICENSE  
  MaterialCommunityIcons.js  
  MaterialIcons.js
```

Automatic Linking - Under the hood - 2 - android

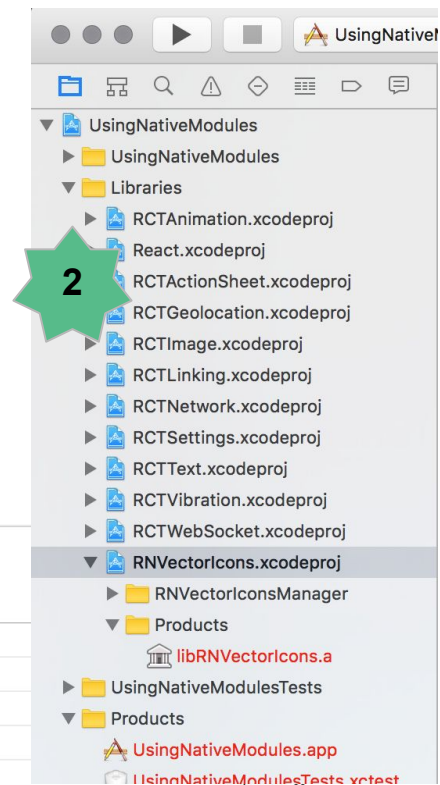
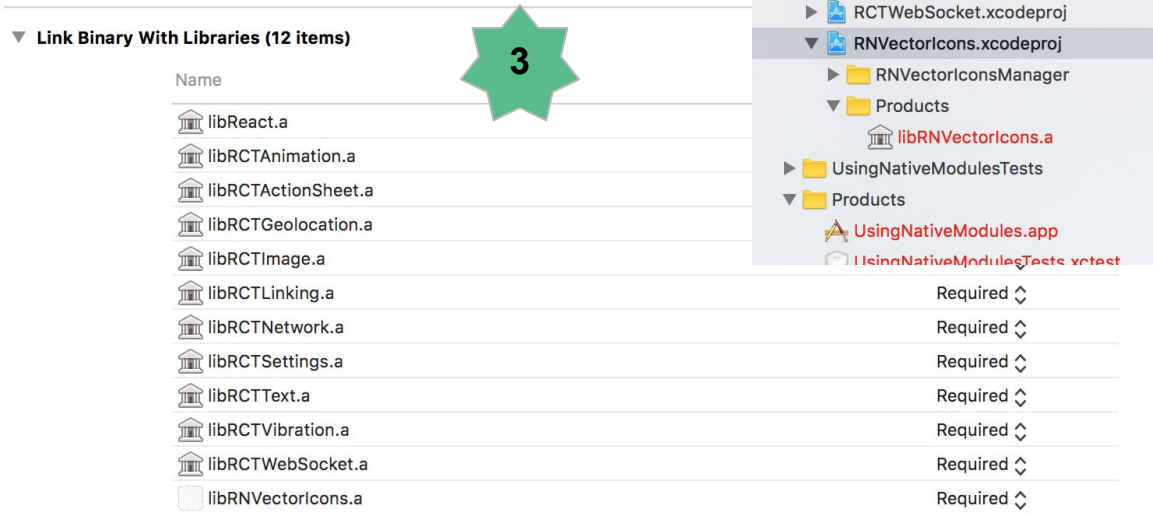
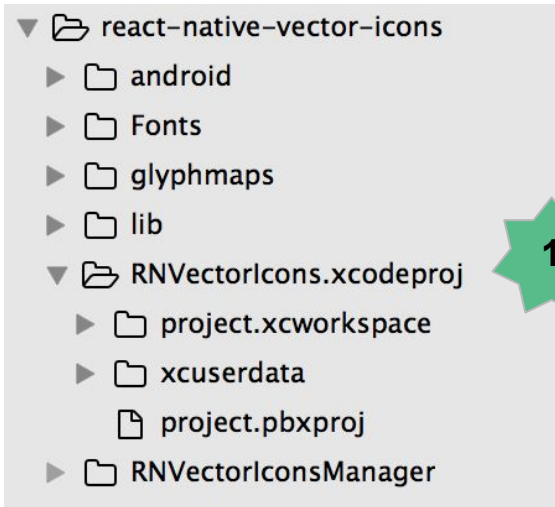
- Updates the local android project settings.gradle with dependency details



```
1 rootProject.name = 'UsingNativeModules'
2 include ':react-native-vector-icons'
3 project(':react-native-vector-icons').projectDir =
4     new File(rootProject.projectDir, '../node_modules/react-native-vector-icons/android')
5
6 include ':app'
7
```

Automatic Linking - Under the hood - 3 - ios

- Copies the native module xcode project from node_modules under Libraries and links binaries




Manual Linking

Be focused and have a checklist and expertise

Manual Linking - Steps

1. `npm install <library-with-native-dependencies> --save`
2. Update the `settings.gradle` of local android project



```
settings.gradle
1  rootProject.name = 'UsingNativeModules'
2  include ':react-native-vector-icons'
3  project(':react-native-vector-icons').projectDir =
4      new File(rootProject.projectDir, '../node_modules/react-native-vector-icons/android')
5
6  include ':app'
7
```


Manual Linking - Steps - android

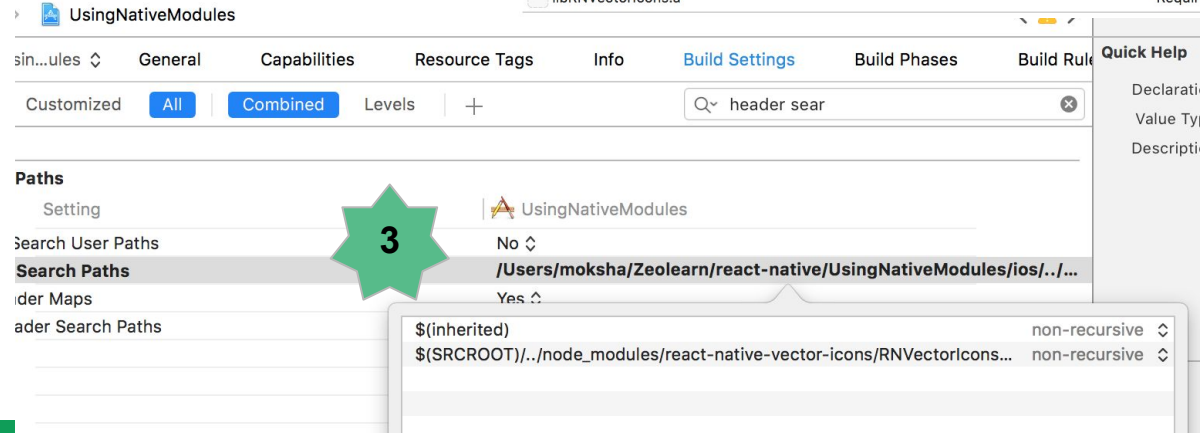
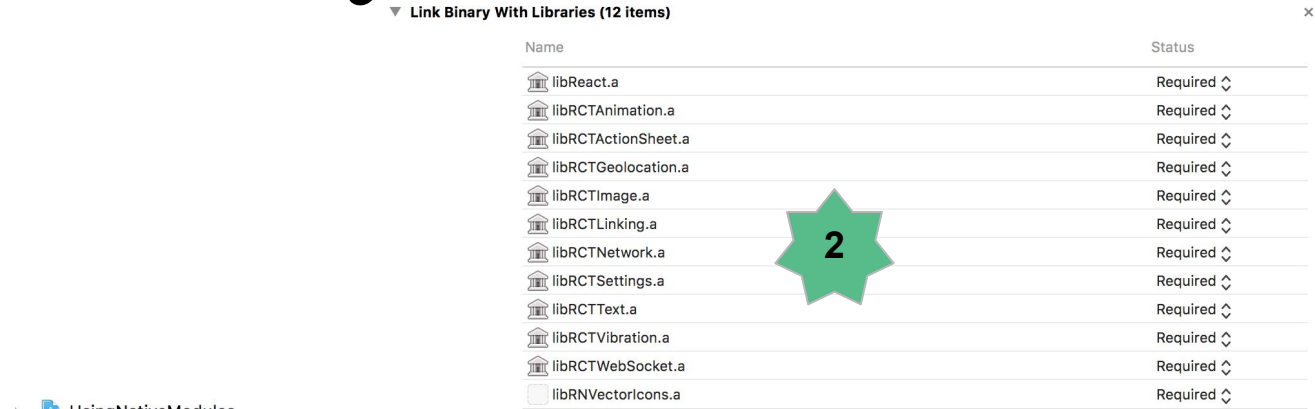
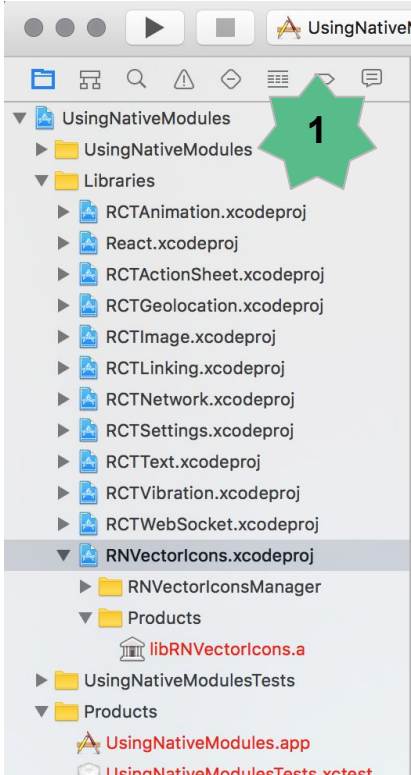
1. Update the settings.gradle of local android project with below configuration

```
settings.gradle
1  rootProject.name = 'UsingNativeModules'
2  include ':react-native-vector-icons'
3  project(':react-native-vector-icons').projectDir =
4      new File(rootProject.projectDir, '../node_modules/react-native-vector-icons/android')
5
6  include ':app'
7
```

Manual Linking - Steps - ios

1. Copy xcodeproject from node_modules to Libraries folder
2. Click on your main project file (the one that represents the .xcodeproj) select Build Phases and drag the static library from the Products folder inside the Library you are importing to Link Binary With Libraries
3. select Build Settings and search for Header Search Paths. There you should include the path to your library

Manual Linking - Steps - ios



Let's code

React Component using Evilicons

```
import React, { Component } from 'react';
import {
  AppRegistry,
  StyleSheet,
  Text,
  View
} from 'react-native';
import Icon from 'react-native-vector-icons/EvilIcons';
import * as globalStyles from './style.global'

export default class UsingNativeModules extends Component {
  render() {
    return (
      <View style={[globalStyles.COMMON_STYLES.pageContainer, styles.container]}>
        <Icon
          name="user"
          style={styles.avatarIcon}
        />
        <Text style={styles.text}>baluragla</Text>
        <Text style={styles.text}>BalaKrishna Ragala</Text>
      </View>
    );
  }
}
```