Write a Native Module

Like Facebook React Native, You.i Engine One lets you extend the functionality of your React Native application with native code.

Let's try that out with a simple native module that updates text to reflect the platform it's running on.

Create a Native Module

1. Create a new file in `MyApp/youi/src` called `PlatformNameModule.h` with the following contents.

> **Note:** Native module class definitions use the macro `YI_RN_MODULE`. Class methods declared in the header also use various `YI_RN_EXPORT*` macros. Together, these macros instruct You.i Engine One to make the class and related methods accessible from JavaScript.

```
#ifndef _PLATFORM_NAME_MODULE_H_
#define _PLATFORM_NAME_MODULE_H_
#include <youireact/NativeModule.h>

class YI_RN_MODULE(PlatformNameModule)
 {
 public:
     YI_RN_EXPORT_NAME(PlatformNameModule);

     YI_RN_EXPORT_CONSTANT(name);
 };

#endif // _PLATFORM_NAME_MODULE_H_
```

2. Next, create `MyApp/youi/src/PlatformNameModule.cpp` as follows.

> **Note:** The class implementation also uses some `YI_RN*` macros. The first (`YI_RN_INSTANTIATE_MODULE`) is mandatory and must always be present in any native module you write.

```
#include "PlatformNameModule.h"
#include <platform/YiDeviceBridgeLocator.h>
#include <platform/YiDeviceInformationBridge.h>
#include <youireact/NativeModuleRegistry.h>

YI_RN_INSTANTIATE_MODULE(PlatformNameModule);
YI_RN_REGISTER_MODULE(PlatformNameModule);

YI_RN_DEFINE_EXPORT_CONSTANT(PlatformNameModule, name)
{
   auto bridge = CYIDeviceBridgeLocator::GetDeviceInformationBridge();
   return ToDynamic(bridge ? bridge->GetDeviceOSName() :  CYIString::EmptyString());
}
```

3. Have a quick read through the code you just copied and pasted to see whether you can follow what it's doing. Our new class, `PlatformNameModule`, has a single exposed constant, `name`, whose value is the DeviceOSName string returned by the You.i Engine One C++ device bridge.

4. To include the native module in your application, you need to tell the compiler about your new class files. Edit `youi/SourceList.cmake` to add the `.cpp` and `.h` files to the `YI_PROJECT_SOURCE` and `YI_PROJECT_HEADERS`

sections, respectively.

```
# © You i Labs Inc. 2000-2019. All rights reserved.

set (YI_PROJECT_SOURCE
    src/App.cpp
    src/AppFactory.cpp
    src/PlatformNameModule.cpp
)

set (YI_PROJECT_HEADERS
    src/App.h
    src/PlatformNameModule.h
)
```

5. Call `youi-tv generate` to take in the new source file dependencies. Then rebuild your app.

Mac Linux Windows

```
youi-tv generate -p osx
youi-tv build -p osx
```

If you run your application now, you'll see that there's no change yet. You need to call the native module via JavaScript.

> **Pro Tip:** The `youi-tv build` command calls `generate` for you the first time you run it. But if you make C++ changes to your application, such as adding a native module, call `youi-tv generate` explicitly to make sure the changes are picked up.

Add the Native Module to JavaScript

1. Add the last import line below to `index.youi.js` to bring all native modules (including the one we just created) into your JavaScript application.

```
import React, { Component } from "react";
import { AppRegistry, Image, StyleSheet, Text, View } from "react-native";
import { FormFactor } from "@youi/react-native-youi";
import { Button } from './button.js';
import { lightStyles, darkStyles } from './styles.js';
import { NativeModules } from 'react-native';
...
```

2. In `render()`, load your native module from `NativeModules`.

```
...
render() {

  const PlatformName = NativeModules.PlatformNameModule;
...
```

3. Update your app's text to make use of the platform name returned by your native module.

```
<Text style={styles.body}>
</Text>
  <Text style={[styles.headlineText, themeStyles.headlineText]}
    accessibilityLabel="Welcome to your first {PlatformName.name} You I React Native app"
  >
    Welcome to your first {PlatformName.name} You.i React Native app!
  </Text>
...
```

Run your app again to see a string that matches the OS of your development platform.

Native modules can do more than expose a constant to JavaScript! When you write a native module for your app, there's

a good chance it will be more complex than the example above. Your native module can expose methods, take a callback, and even trigger an event. And of course, you can optionally register .cpp and .h files based on the target platform, giving you the power to expose platform-specific features to your app. For more information, see the full Native Module documentation.

Congratulations! You now have the basics of writing a You.i RN application.

So far, we've seen your You.i RN app work on just one platform - your development platform. The best part about You.i Engine One is that you can code it once and deploy to multiple platforms.

Follow along with the next step in this tutorial as we try your app on an Android or Roku device.

Updated: February 18, 2020