集成到现有原生应用

如果你正准备从头开始制作一个新的应用，那么 React Native 会是个非常好的选择。但如果你只想给现有的原生应用中添加一两个视图或是业务流程，React Native 也同样不在话下。只需简单几步，你就可以给原有应用加上新的基于 React Native 的特性、画面和视图等。

具体的步骤根据你所开发的目标平台不同而不同。

> 译注：本文档可能更新不够及时，不能保证适用于最新版本，欢迎了解的朋友使用页面底部的编辑链接帮忙改进此文档。一个实用的建议是可以使用 `npx react-native init NewProject` 创建一个最新版本的纯 RN 项目，去参考其 Podfile 或是 gradle 等的配置，以它们为准。

Android (Java) iOS (Objective-C) iOS (Swift)

核心概念

把 React Native 组件集成到 Android 应用中有如下几个主要步骤：

1. 配置好 React Native 依赖和项目结构。
2. 创建 js 文件，编写 React Native 组件的 js 代码。
3. 在应用中添加一个 `ReactRootView` 。这个 `ReactRootView` 正是用来承载你的 React Native 组件的容器。
4. 启动 React Native 的 Metro 服务，运行应用。
5. 验证这部分组件是否正常工作。

开发环境准备

首先按照开发环境搭建教程来安装 React Native 在 Android 平台上所需的一切依赖软件。

1. 配置项目目录结构

首先创建一个空目录用于存放 React Native 项目，然后在其中创建一个 `/android` 子目录，把你现有的 Android 项目拷贝到 `/android` 子目录中。

2. 安装 JavaScript 依赖包

在项目根目录下创建一个名为 `package.json` 的空文本文件，然后填入以下内容：

```
{
  "name": "MyReactNativeApp",
  "version": "0.0.1",
  "private": true,
  "scripts": {
    "start": "yarn react-native start"
  }
}
```

> 示例中的 `version` 字段没有太大意义（除非你要把你的项目发布到 npm 仓库）。`scripts` 中是用于启动 Metro 服务的命令。

接下来我们使用 yarn 或 npm（两者都是 node 的包管理器）来安装 React 和 React Native 模块。请打开一个终端/命令提示行，进入到项目目录中（即包含有 package.json 文件的目录），然后运行下列命令来安装：

```
$ yarn add react-native
```

这样默认会安装最新版本的 React Native，同时会打印出类似下面的警告信息（你可能需要滚动屏幕才能注意到）：

> warning "react-native@0.52.2" has unmet peer dependency "react@16.2.0".

这是正常现象，意味着我们还需要安装指定版本的 React：

```
$ yarn add react@16.2.0
```

注意必须严格匹配警告信息中所列出的版本，高了或者低了都不可以。

> 如果你使用多个第三方依赖，可能这些第三方各自要求的 react 版本有所冲突，此时应优先满足 `react-native` 所需要的 `react` 版本。其他第三方能用则用，不能用则只能考虑选择其他库。

所有 JavaScript 依赖模块都会被安装到项目根目录下的 `node_modules/` 目录中（这个目录我们原则上不复制、不移动、不修改、不上传，随用随装）。

把 `node_modules/` 目录记录到 `.gitignore` 文件中（即不上传到版本控制系统，只保留在本地）。

把 React Native 添加到你的应用中

配置 maven

在你的 app 中 `build.gradle` 文件中添加 React Native 和 JSC 引擎依赖:

```
dependencies {
    implementation "com.android.support:appcompat-v7:27.1.1"
    ...
    implementation "com.facebook.react:react-native:+" // From node_modules
    implementation "org.webkit:android-jsc:+"
}
```

> 如果想要指定特定的 React Native 版本，可以用具体的版本号替换 `+`，当然前提是你从 npm 里下载的是这个版本。

在项目的 `build.gradle` 文件中为 React Native 和 JSC 引擎添加 maven 源的路径，必须写在 "allprojects" 代码块中

```
allprojects {
    repositories {
        maven {
            // All of React Native (JS, Android binaries) is installed from
npm
            url "$rootDir/../node_modules/react-native/android"
        }
        maven {
            // Android JSC is installed from npm
            url("$rootDir/../node_modules/jsc-android/dist")
        }
        ...
    }
    ...
}
```

> 确保依赖路径的正确！以免在 Android Studio 运行 Gradle 同步构建时抛出 “Failed to resolve: com.facebook.react:react-native:0.x.x" 异常。

启用原生模块的自动链接

To use the power of autolinking, we have to apply it a few places. First add the following entry to `settings.gradle`：

```
apply from: file("../node_modules/@react-native-community/cli-platform-android/native_modules.gradle"); applyNativeModulesSettingsGradle(settings)
```

Next add the following entry at the very bottom of the `app/build.gradle`：

```
apply from: file("../../node_modules/@react-native-community/cli-platform-android/native_modules.gradle"); applyNativeModulesAppBuildGradle(project)
```

配置权限

接着，在 `AndroidManifest.xml` 清单文件中声明网络权限:

```
<uses-permission android:name="android.permission.INTERNET" />
```

如果需要访问 `DevSettingsActivity` 界面（即开发者菜单），则还需要在 `AndroidManifest.xml` 中声明:

```
<activity android:name="com.facebook.react.devsupport.DevSettingsActivity" />
```

开发者菜单一般仅用于在开发时从 Packager 服务器刷新 JavaScript 代码，所以在正式发布时你可以去掉这一权限。

允许明文传输（http 接口） (API level 28+)

> 从 Android 9 (API level 28)开始，默认情况下明文传输（http 接口）是禁用的，只能访问 https 接口。 this prevents your application from connecting to the [Metro bundler][metro]. The changes below allow cleartext traffic in debug builds.

1. 为 debug 版本启用 `usesCleartextTraffic` 选项

在 `src/debug/AndroidManifest.xml` 中添加 `usesCleartextTraffic` 选项：

```
<!-- ... -->
<application
  android:usesCleartextTraffic="true" tools:targetApi="28" >
  <!-- ... -->
</application>
<!-- ... -->
```

如果希望在正式打包后也能继续访问 http 接口，则需要在 `src/main/AndroidManifest.xml` 中也添加这一选项。

To learn more about Network Security Config and the cleartext traffic policy see this link.

代码集成

Now we will actually modify the native Android application to integrate React Native.

React Native 组件

我们首先要写的是"High Score"（得分排行榜）的 JavaScript 端的代码。

1. 创建一个 `index.js` 文件

首先在项目根目录中创建一个空的 `index.js` 文件。(注意一些老的教程可能提到，在 0.49 版本之前是 index.android.js 文件)

`index.js` 是 React Native 应用在 Android 上的入口文件。而且它是不可或缺的！它可以是个很简单的文件，简单到可以只包含一行 `require/import` 导入语句。本教程中为了简单示范，把全部的代码都写到了 `index.js` 里（当然实际开发中我们并不推荐这样做）。

2. 添加你自己的 React Native 代码

在 `index.js` 中添加你自己的组件。这里我们只是简单的添加一个 `<Text>` 组件，然后用一个带有样式的 `<View>` 组件把它包起来。

```
import React from 'react';
import {
  AppRegistry,
  StyleSheet,
```

```
  Text,
  View
} from 'react-native';

class HelloWorld extends React.Component {
  render() {
    return (
      <View style={styles.container}>
        <Text style={styles.hello}>Hello, World</Text>
      </View>
    );
  }
}
var styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center'
  },
  hello: {
    fontSize: 20,
    textAlign: 'center',
    margin: 10
  }
});

AppRegistry.registerComponent(
  'MyReactNativeApp',
  () => HelloWorld
);
```

3. 配置权限以便开发中的红屏错误能正确显示

如果你的应用会运行在 Android 6.0（API level 23）或更高版本，请确保你在开发版本中有打开 悬浮窗(overlay) 权限。你可以在代码中使用 `Settings.canDrawOverlays(this);` 来检查。之所以需要这一权限，是因为我们会把开发中的报错显示在悬浮窗中（仅在开发阶段需要）。在 Android 6.0（API level 23）中用户需要手动同意授权。具体请求授权的做法是在 `onCreate()` 中添加如下代码。其中 `OVERLAY_PERMISSION_REQ_CODE` 是用于回传授权结果的字段。

```
  private final int OVERLAY_PERMISSION_REQ_CODE = 1;  // 任写一个值

  ...

  if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
      if (!Settings.canDrawOverlays(this)) {
          Intent intent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION,
                                     Uri.parse("package:" + getPackageName()));
          startActivityForResult(intent, OVERLAY_PERMISSION_REQ_CODE);
```

```
    }
}
```

Finally, the `onActivityResult()` method (as shown in the code below) has to be overridden to handle the permission Accepted or Denied cases for consistent UX. Also, for integrating Native Modules which use `startActivityForResult`, we need to pass the result to the `onActivityResult` method of our `ReactInstanceManager` instance.

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data)
{
    if (requestCode == OVERLAY_PERMISSION_REQ_CODE) {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
            if (!Settings.canDrawOverlays(this)) {
                // SYSTEM_ALERT_WINDOW permission not granted
            }
        }
    }
    mReactInstanceManager.onActivityResult( this, requestCode, resultCode,
data );
}
```

核心组件： `ReactRootView`

我们还需要添加一些原生代码来启动 React Native 的运行时环境并让它开始渲染。首先需要在一个 `Activity` 中创建一个 `ReactRootView` 对象，然后在这个对象之中启动 React Native 应用，并将它设为界面的主视图。

> 如果你要在安卓 5.0 以下的系统上运行，请用 `com.android.support:appcompat` 包中的 `AppCompatActivity` 代替 `Activity` 。

```
public class MyReactActivity extends Activity implements
DefaultHardwareBackBtnHandler {
    private ReactRootView mReactRootView;
    private ReactInstanceManager mReactInstanceManager;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        SoLoader.init(this, false);

        mReactRootView = new ReactRootView(this);
```

```
        List<ReactPackage> packages = new
PackageList(getApplication()).getPackages();
        // 有一些第三方可能不能自动链接，对于这些包我们可以用下面的方式手动添加进来：
        // packages.add(new MyReactNativePackage());
        // 同时需要手动把他们添加到`settings.gradle`和 `app/build.gradle`配置文件
中。

        mReactInstanceManager = ReactInstanceManager.builder()
                .setApplication(getApplication())
                .setCurrentActivity(this)
                .setBundleAssetName("index.android.bundle")
                .setJSMainModulePath("index")
                .addPackages(packages)
                .setUseDeveloperSupport(BuildConfig.DEBUG)
                .setInitialLifecycleState(LifecycleState.RESUMED)
                .build();
         // 注意这里的MyReactNativeApp 必须对应"index.js"中的
        // "AppRegistry.registerComponent()"的第一个参数
        mReactRootView.startReactApplication(mReactInstanceManager,
"MyReactNativeApp", null);

        setContentView(mReactRootView);
    }

    @Override
    public void invokeDefaultOnBackPressed() {
        super.onBackPressed();
    }
}
```

Perform a “Sync Project files with Gradle” operation.

如果你使用的是 Android Studio , 可以使用 `Alt + Enter` 快捷键来自动为 MyReactActivity 类补上缺失的 import 语句。注意 `BuildConfig` 应该是在你自己的包中自动生成，无需额外引入。千万不要从 `com.facebook...` 的包中引入！

我们需要把 `MyReactActivity` 的主题设定为 `Theme.AppCompat.Light.NoActionBar` ，因为里面有许多组件都使用了这一主题。

```
<activity
  android:name=".MyReactActivity"
  android:label="@string/app_name"
  android:theme="@style/Theme.AppCompat.Light.NoActionBar">
</activity>
```

> 一个 `ReactInstanceManager` 可以在多个 activities 或 fragments 间共享。 You will want to make your own `ReactFragment` or `ReactActivity` and have a singleton *holder* that holds a `ReactInstanceManager`. When you need the `ReactInstanceManager` (e.g., to hook up the `ReactInstanceManager` to the lifecycle of those Activities or Fragments) use the one provided by the singleton.

下一步我们需要把一些 activity 的生命周期回调传递给 `ReactInstanceManager`：

```
@Override
protected void onPause() {
    super.onPause();

    if (mReactInstanceManager != null) {
        mReactInstanceManager.onHostPause(this);
    }
}

@Override
protected void onResume() {
    super.onResume();

    if (mReactInstanceManager != null) {
        mReactInstanceManager.onHostResume(this, this);
    }
}

@Override
protected void onDestroy() {
    super.onDestroy();

    if (mReactInstanceManager != null) {
        mReactInstanceManager.onHostDestroy(this);
    }
    if (mReactRootView != null) {
        mReactRootView.unmountReactApplication();
    }
}
```

我们还需要把后退按钮事件传递给 React Native：

```
@Override
 public void onBackPressed() {
    if (mReactInstanceManager != null) {
        mReactInstanceManager.onBackPressed();
    } else {
```

```
            super.onBackPressed();
        }
    }
```

This allows JavaScript to control what happens when the user presses the hardware back button (e.g. to implement navigation). When JavaScript doesn't handle the back button press, your `invokeDefaultOnBackPressed` method will be called. By default this finishes your `Activity`.

Finally, we need to hook up the dev menu. By default, this is activated by (rage) shaking the device, but this is not very useful in emulators. So we make it show when you press the hardware menu button (use `Ctrl + M` if you're using Android Studio emulator):

```
@Override
public boolean onKeyUp(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_MENU && mReactInstanceManager != null) {
        mReactInstanceManager.showDevOptionsDialog();
        return true;
    }
    return super.onKeyUp(keyCode, event);
}
```

现在 activity 已就绪，可以运行一些 JavaScript 代码了。

测试集成结果

You have now done all the basic steps to integrate React Native with your current application. Now we will start the [Metro bundler][metro] to build the `index.bundle` package and the server running on localhost to serve it.

1. 运行 Metro 服务

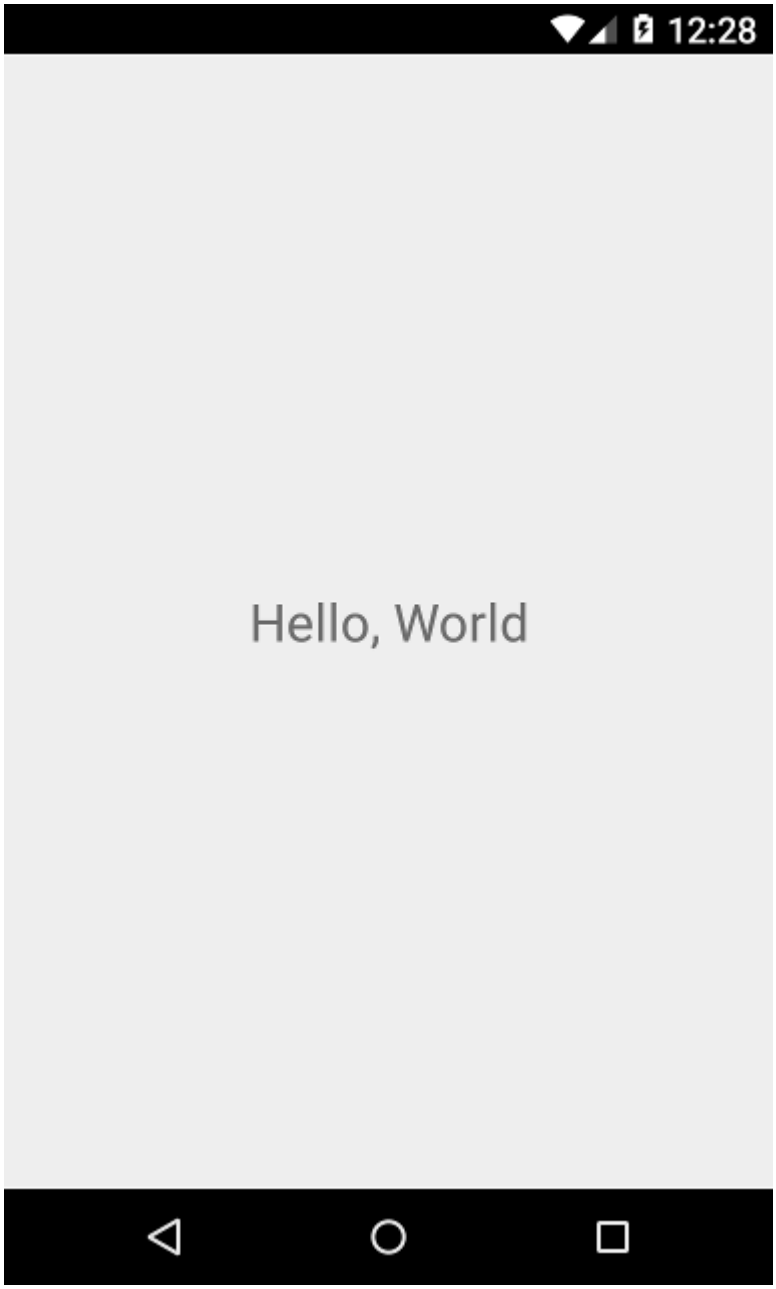
运行应用首先需要启动开发服务器（Metro）。你只需在项目根目录中执行以下命令即可：

```
$ yarn start
```

2. 运行你的应用

保持 Metro 的窗口运行不要关闭，然后像往常一样编译运行你的 Android 应用(在命令行中执行 `./gradlew installDebug` 或是在 Android Studio 中编译运行)。

编译执行一切顺利进行之后，在进入到 MyReactActivity 时应该就能立刻从 Metro 中读取 JavaScript 代码并执行和显示：



在 Android Studio 中打包

你也可以使用 Android Studio 来打 release 包！其步骤基本和原生应用一样，只是在每次编译打包之前需要先执行 js 文件的打包(即生成离线的 jsbundle 文件)。具体的 js 打包命令如下：

```
$ npx react-native bundle --platform android --dev false --entry-file
index.js --bundle-output android/com/your-company-name/app-package-
name/src/main/assets/index.android.bundle --assets-dest android/com/your-
company-name/app-package-name/src/main/res/
```

> 注意把上述命令中的路径替换为你实际项目的路径。如果 assets 目录不存在，则需要提前自己创建一个。

然后在 Android Studio 中正常生成 release 版本即可！

然后呢?

然后就可以开发啦~可是我完全不会 React Native 怎么办?

我们建议你先通读本站的所有文档，看看博客，看看论坛。如果觉得知识太零散，不够系统，那么你也可以考虑下购买我们的付费咨询服务。

改进文档

最近更新 ***2021/7/13***