开发自己的react-native组件并发布到npm

Quenice LV.3
2018年09月06日 09:26 · 阅读 5656

原文链接：www.jianshu.com

写在前面

在做 `react-native` 开发的时候，我们经常会找到一些第三方组件，并且通过 `npm install` 的方式很方便的安装使用。在使用的同时，你是否想过，我们自己应该如何开发并发布一个组件呢？不管是给自己的多个项目共用，或者开源给到别人用，这都是一件很酷的事情。

那么今天，我就以我自己开发的一个在 `ios` 和 `android` 通用的 `CardView` 组件为例来讲一下，如何开发一个自己的组件，并开源到github、发布到npm上去

一个说明

我的组件名为react-native-rn-cardview。教程里如果出现 `react-native-cardview` 组件名，视为同一意思。

1 创建并实现

1.1 创建自定义组件模版项目

1.1.1 安装react-native-create-library

```
$ npm install -g react-native-create-library
```

1.1.2 创建模板项目

我们用命令 `react-native-create-library` 创建项目，并指定平台为 `ios,android`，指定 `android` 中的 `package`，其他参数可以自行参考在react-native-create-library在 `github` 上的文档说明，这里就不赘述

```
$ react-native-create-library --package-identifier com.quenice.cardview --platforms android,ios cardview
```

我们重命名一下项目名

```
$ mv cardview react-native-cardview
```

有人可能会说，楼主为什么不直接生成 `react-native-cardview` 的项目，而要先生成 `cardview` 再重命名。其实这是一个小技巧，因为利用 `react-native-create-library` 生产的项目，一些跟组件相关的名称或者类会默认加上 `react-native` 或者 `RN` 前缀。
例如，如果你的初始项目名是 `react-native-card-view`，那么 `package.json` 中定义的组件名将是 `react-native-react-native-card-view`，android模块中定义的相关类会
是 `RNReactNativeCardviewModule.java`，这显然比较丑啊。

ok，我们继续。

现在的目录结构：

```
$ tree
```

```
└── react-native-cardview
    ├── README.md
    ├── android
    │   ├── build.gradle
    │   └── src
    │       └── main
    │           ├── AndroidManifest.xml
    │           └── java
    │               └── com
    │                   └── reactlibrary
    │                       ├── RNCardviewModule.java
    │                       └── RNCardviewPackage.java
    ├── index.js
```

```
    ├── ios
    │   ├── RNCardview.h
    │   ├── RNCardview.m
    │   ├── RNCardview.podspec
    │   ├── RNCardview.xcodeproj
    │   │   └── project.pbxproj
    │   └── RNCardview.xcworkspace
    │       └── contents.xcworkspacedata
    └── package.json
```

生成好组件项目后，就可以开始编写实现代码了

2 编写代码

编写代码分为三部分，一部分是android原生代码，一部分是iOS原生代码，一部分是react-native(或者javascript)代码。由于react-native-cardview只涉及到android原生模块，所以本篇文章暂不涉及到iOS原生模块开发，如果大家感兴趣，我可以另开一篇文章专门讲一下iOS原生模块相关内容

2.1 编写Android Native Module

编写android原生代码，一般以下三个类是必须的：

2.1.1 RNxxxModule

这个类主要作用是定义原生模块名，可以直接在javascript中通过 `React.NativeModules.xxx` 来访问，其中 `xxx` 是在 `RNxxxModule` 类中定义的 `getName` 方法返回值。以下为我组件react-native-cardview中的Module类

RNCardViewModule.java

scala 复制代码

```scala
package com.quenice.reactnative;

import com.facebook.react.bridge.ReactApplicationContext;
import com.facebook.react.bridge.ReactContextBaseJavaModule;
import com.facebook.react.bridge.ReactMethod;
import com.facebook.react.bridge.Callback;

public class RNCardViewModule extends ReactContextBaseJavaModule {

  private final ReactApplicationContext reactContext;

  public RNCardViewModule(ReactApplicationContext reactContext) {
    super(reactContext);
    this.reactContext = reactContext;
  }

  @Override
  public String getName() {
    return "RNCardView";
  }
}
```

2.1.2 RNxxxManager

Manager类主要是组件的原生实现，并且把react-native组件的属性映射到原生属性

RNCardViewManager.java

java 复制代码

```java
package com.quenice.reactnative;

import android.graphics.Color;
import android.support.v7.widget.CardView;
import android.view.View;

import com.facebook.react.uimanager.PixelUtil;
import com.facebook.react.uimanager.ThemedReactContext;
import com.facebook.react.uimanager.ViewGroupManager;
import com.facebook.react.uimanager.annotations.ReactProp;
import com.facebook.react.views.view.ReactViewGroup;

public class RNCardViewManager extends ViewGroupManager<CardView> {
    @Override
    public String getName() {
        return "RNCardView";
    }

    @Override
    protected CardView createViewInstance(ThemedReactContext reactContext) {
```

```
        CardView cardView = new CardView(reactContext);
        cardView.setUseCompatPadding(true);
        cardView.setContentPadding(0, 0, 0, 0);
        ReactViewGroup reactViewGroup = new ReactViewGroup(reactContext);
        cardView.addView(reactViewGroup);
        return cardView;
    }

    @ReactProp(name = "cardElevation", defaultFloat = 0f)
    public void setCardElevation(CardView view, float cardElevation) {
        view.setCardElevation(PixelUtil.toPixelFromDIP(cardElevation));
    }
...
}
```

2.1.3 RNxxxPackage

Package类主要用于注册原生模块、原生组件实现，也就是注册上面的Module和Manager类

RNCardViewPackage.java

java 复制代码

```java
package com.quenice.reactnative;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import com.facebook.react.ReactPackage;
import com.facebook.react.bridge.NativeModule;
import com.facebook.react.bridge.ReactApplicationContext;
import com.facebook.react.uimanager.ViewManager;
import com.facebook.react.bridge.JavaScriptModule;
public class RNCardViewPackage implements ReactPackage {
    @Override
    public List<NativeModule> createNativeModules(ReactApplicationContext reactContext) {
      return Arrays.<NativeModule>asList(new RNCardViewModule(reactContext));
    }

    // Deprecated from RN 0.47
    public List<Class<? extends JavaScriptModule>> createJSModules() {
      return Collections.emptyList();
    }

    @Override
    public List<ViewManager> createViewManagers(ReactApplicationContext reactContext) {
      return Arrays.<ViewManager>asList(new RNCardViewManager());
    }
}
```

2.2 编写iOS原生代码

react-native-cardview

`iOS` 的实现方式直接利用 `react-native` 中 `shadow` 相关属性就可以实现，所以本文暂不涉及

2.3 编写ReactNative代码

编写好了 `android/iOS` 原生模块后，需要编写 `javascript` 代码来桥接 `react-native` 与原生模块。

RNCardView.android.js

javascript 复制代码

```javascript
import PropTypes from 'prop-types';
import {requireNativeComponent, View} from 'react-native';

const iface = {
    name: 'CardView',
    propTypes: {
        cardElevation: PropTypes.number,
        maxCardElevation: PropTypes.number,
        backgroundColor: PropTypes.string,
        radius: PropTypes.number,
        ...View.propTypes, // include the default view properties
    },
};

module.exports = requireNativeComponent('RNCardView', iface);
```

3 代码上传与组件发布

3.1 代码上传到github

编写完代码后，我们需要把它上传到github上，之后在组件发布到npm的时候也需要用到代码的github地址。如果你没有做github相关的配置，可以参考我另一篇文章：安装Git并配置连接GitHub

执行以下命令把代码同步到你github对应的repository中：

```ruby
$ git add .
$ git commit -a -m "initial commit"
$ git push -u origin master
```

同步之后可以到github中看下是否push成功：

```arduino
https://github.com/YourGithubAccount/YourRepository
```

3.2 组件发布

开发好组件之后，想在其他的项目(或者提供给其他人安装使用)中通过 `npm install` 的方式安装你的组件，那么你的组件必须发布到npm registry中。

3.2.1 npm registry

npm registry 是什么

简单来说，npm registry就相当于一个包注册管理中心。它管理着全世界的开发者们发布上来的各种插件，同时开发者们可以通过 `npm install` 的方式安装所需要的插件。

npm官方registry为：registry.npmjs.org/

国内速度较快的为：registry.npm.taobao.org/

查看

你可以查看当前使用的registry:

```arduino
$ npm config get registry
```

切换

当然也可以通过命令切换当前使用的npm registry

```shell
# 全局切换
$ npm config set registry http://registry.npmjs.org/
```

有时候你可能只想在执行某些npm命令时临时切换，这个时候，可以使用 `--registry` 来指定临时切换的registry，比如在npm发布

```ruby
$ npm publish --registry http://registry.npmjs.org/
```

就可以临时指定，当然，在命令执行结束之后，registry仍然会恢复到原来的registry

3.2.2 创建/登陆npm registry账户

要发布组件到npm registry，你必须要是npm registry的注册用户，通过：

```ruby
$ npm adduser
```

来新增一个用户，或者你已经在官网注册了一个用户，可以通过：

```ruby
$ npm login
```

来登陆npm registry账户。

利用以下两种方式来确认你是否创建/登陆成功npm registry

1. 命令`$ npm whoami`确认本地是否成功登陆认证成功
2. 在线打开 npmjs.com/~username 查看是否创建账户成功

3.2.3 发布前准备

3.2.3.1 .gitignore 和 .npmignore

1. 在**.gitignore**中定义哪些文件不上传到github中
2. 在**.npmignore**中定义哪些文件发布时不打包
3. 如果有**.gitignore**但是没有**.npmignore**文件，那么**.gitignore**可以充当**.npmignore**的作用
4. 具体规则可以参照：npm-developers, .gitignore or .npmignore pattern rules

3.2.3.2 package.json

package.json文件定义了发布的所有信息，包括：组件名、版本、作者、描述、依赖等等关键信息。具体可以参照 Working with package.json

下面是react-native-cardview的package.json文件内容：

```perl 复制代码
{
  "name": "react-native-rn-cardview",
  "version": "1.0.0",
  "description": "A ReactNative CardView Component for android and iOS",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [
    "react-native",
    "react-component",
    "react-native-component",
    "react",
    "mobile",
    "ios",
    "android",
    "cardview"
  ],
  "author": {
    "name": "quenice",
    "email": "qiubing.it@gmail.com"
  },
  "license": "MIT",
  "repository": {
    "type": "git",
    "url": "git@github.com:quenice/react-native-cardview.git"
  },
  "devDependencies": {
    "react": "^16.2.0",
    "react-native": "^0.53.0"
  },
  "peerDependencies": {
    "react": "^16.2.0"
  },
  "dependencies": {
    "prop-types": "^15.6.0"
  }
}
```

3.2.3.3 编写readme.md

可以在readme.md文件中详细说明组件的使用方法、注意事项等。一般使用Markdown语法来编写

3.2.4 发布

做好以上准备之后，就可以发布了。这里需要注意，首次发布跟后面更新发布是不一样的。

首次发布

第一次发布的话，直接执行命令：

```ruby 复制代码
$ npm publish
```

就搞定了，可以在线查看确认是否发布成功。访问链接(<package>是你发布的npm package名)：
www.npmjs.com/package/<pa...
看看是否已经有内容了，有内容说明发布成功了。

更新发布

如果不是首次发布，需要执行两个命令

```ruby 复制代码
$ npm version <update_type>
$ npm publish
```

`$ npm version` 命令是用来自动更新版本号，`update_type` 取值有 `patch ``minor ``major` 。那么在什么场景应该选择什么 `update_type` 呢？看下表

update_type	场景	版本号规则	举例
-	首次发布	版本号1.0.0	1.0.0
path	修复bug、微小改动时	从版本号第3位开始增量变动	1.0.0 -> 1.0.1
minor	上线新功能，并且对当前版本已有功能模块不影响时	从版本号第2位开始增量变动	1.0.3 -> 1.1.3
major	上线多个新功能模块，并且对当前版本已有功能会有影响时	从版本号第1位开始增量变动	1.0.3 -> 2.0.0

注意

如果首次发布版本号不是1.0.0的话，那么用 `$ npm version <update_type>` 来更新会报错，因为你没有按照它约定的版本规则来，这个时候，你可以手动修改 `package.json` 中的 `version` 字段为符合约定规则的版本号，然后直接执行 `$ npm publish` 就可以，然后下次再增量更新的时候，就可以直接使用 `$ npm version <update_type>` 的方式来自动更新版本号了

4 测试

组件从开发到最终发布的过程中，需要不断进行测试，确保功能正常，那如何进行测试呢？

4.1 创建一个react-native项目

首先我们创建一个叫做example的react-native项目

```csharp 复制代码
$ react-native init example
```

`example` 项目目录可以和组件项目目录 `react-native-cardview` 同级，当然你也可以放在任何你想放的位置，这里为了操作方便，我们就把两个目录放在同级目录。也就是说，现在的目录是这样

```java 复制代码
$ tree
.
├── example
└── react-native-cardview
```

然后我们要做的就是把本地或者已发布的组件安装到 `example` 项目中进行测试

4.2 本地代码测试

在组件未发布之前，我们可以直接安装本地代码到 `example` 项目中进行测试，有以下几种方式都可以做到

4.2.1 yarn link

```shell 复制代码
$ cd react-native-cardview
$ yarn link
$ cd ../example
```

```
$ yarn link react-native-cardview
$ react-native link react-native-cardview
```

说明几点：

1. `yarn link` 是把当前目录中的本地代码用yarn注册为 `react-native-cardview` 的一个本地组件，组件名字 `react-native-cardview` 其实是根据 `package.json` 中的 `name` 字段的值来的，跟目录名无关，只不过这里正好等于目录名
2. `yarn link react-native-cardview` 命令是把这个本地组件 `react-native-cardview` 安装到了 `example` 的项目中，你可以在 `example/node_modules` 中找到这个组件
3. `react-native link react-native-cardview` 这个大家应该知道，就是做了 `android/iOS` 的原生模块link
4. 其实 `yarn link` 这种方式简单来说，就是做了一个 `symbol link`，也就是说，`example/node_modules/react-native-cardview/` 目录中的内容是指向 `react-native-cardview/` 目录内容，你改动 `react-native-cardview/` 目录下的代码，相当于直接改动 `example/node_modules/react-native-cardview/` 这个目录中的代码，这样就能够达到边修改组件代码边看效果的目的了

4.2.2 package.json中配置本地路径

直接在 `example` 的 `package.json` 中增加 `dependencies`

example/package.json

```json 复制代码
{
  "name":"example",
  ...
  "dependencies": {
    "react-native": "^0.55.4",
    "react-native-cardview":"file:../react-native-cardview",
    ...
  }
  ...
}
```

然后执行

```java 复制代码
$ react-native link react-native-cardview
```

跟 `yarn link` 一样，也相当于做了 `symbol link`，直接修改 `react-native-cardview/` 目录下的代码，相当于直接改动 `example/node_modules/react-native-cardview/` 这个目录中的代码

4.2.3 直接copy本地代码

这种方式就比较简单粗暴了，直接copy `react-native-cardview/` 目录中内容到 `example/node_modules/react-native-cardview/` 这个目录中

```shell 复制代码
$ cp -rf react-native-cardview/ example/node_modules/
```

然后执行

```java 复制代码
$ react-native link react-native-cardview
```

这种方式缺点就是每次在 `react-native-cardview/` 改完代码后，需要手工copy到 `example/node_modules/react-native-cardview/`

4.3 已上传/发布代码测试

已上传到github或者发布到npm registry的组件，测试方式就跟普通我们安装一个第三方组件一样了。

4.3.1 通过github

加入你的代码通过git上传到了github仓库上，那么，你可以直接通过 `npm install` 来安装你的组件

```arduino 复制代码
npm install --save https://github.com/quenice/react-native-cardview.git
```

或者

```scss
npm install --save git@github.com:quenice/react-native-cardview.git
```

注意:根据你自己ghthub上的URL替换以上的 `HTTPS` 或者 `SSH`

然后执行

```java
$ react-native link react-native-rn-cardview
```

4.3.2 通过npm

这种方式就跟按照第三方组件没有区别了

```css
$ npm install --save react-native-rn-cardview
```

然后执行

```java
$ react-native link react-native-rn-cardview
```

结语

至此，一个 `react-native` 组件完整的**开发-测试-发布**的生命周期就讲完了。

由于是结合我自己开发的组件react-native-rn-cardview的实际开发过程，所以难免有遗漏，肯定也有许多不足的地方。如果大家有什么问题，或者发现哪里有错误，欢迎大家在评论区给我留言，我们一起探讨、一起解决。

另外如果在 `react-native` 中有需要用到 `CardView` 的，欢迎使用react-native-rn-cardview