5.6 Linking(链接)

`Linking` 提供了一个通用的接口来与传进传出的 app 链接交互.

基本用法

处理链接

如果你的应用被注册到你的 app 里面的外部链接调起，你可以在任何组件中这样处理：

```
  componentDidMount() {
    var url = Linking.getInitialURL().then((url) => {
       if (url) {
        console.log('Initial url is: ' + url);
      }
    }).catch(err => console.error('An error occurred', err));
  },
```

注意：在 android 上添加链接的详细说明请参考

Enabling Deep Links for App Content - Add Intent Filters for Your Deep Links

如果你想在已经存在的 MaiinActivity 实例中接受(intent)意图，你可以在 `AndroidManifest.xml` 中给

MainActivity 的启动模式`(launchMode)`添加 `singleTask`，更多信息请看`<activity>`文档：

```
<activity
  android:name=".MainActivity"
  android:launchMode="singleTask">
```

注意：在 ios 你需要按照以下的步骤在你的应用里添加 `RCTLinking`，如要在 app 启动后也监听传入的

app 链接，需要在 `AppDelegate.m` 添加以下代码

```
#import "RCTLinkingManager.h"
```

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url
  sourceApplication:(NSString *)sourceApplication annotation:(id)annotation
{
  return [RCTLinkingManager application:application openURL:url
            sourceApplication:sourceApplication
annotation:annotation];
}
```

```
// Only if your app is using [Universal Links]
(https://developer.apple.com/library/prerelease/ios/documentation/General/Conceptu
al/AppSearch/UniversalLinks.html)
```

```
- (BOOL)application:(UIApplication *)application continueUserActivity:
(NSUserActivity *)userActivity
   restorationHandler:(void (^)(NSArray * _Nullable))restorationHandler
{
  return  [RCTLinkingManager application:application
            continueUserActivity:userActivity
              restorationHandler:restorationHandler];
}
```

然后你的 React 组件就可以监听 `Linking` 相关事件了

```
componentDidMount() {
  Linking.addEventListener('url', this._handleOpenURL);
},
componentWillUnmount() {
  Linking.removeEventListener('url', this._handleOpenURL);
},
_handleOpenURL(event) {
  console.log(event.url);
}
```

打开外部链接

要启动一个连接相对应的应用(打开浏览器，邮箱，通讯录等，)，需要调用：

```
Linking.openURL(url).catch(err => console.error('An error occurred', err));
```

如果想在打开链接前先检查是否安装了对应的应用，则调用以下方法:

```
Linking.canOpenURL(url).then(supported => {
  if (!supported) {
    console.log('Can\'t handle url: ' + url);
  } else {
    return Linking.openURL(url);
  }
}).catch(err => console.error('An error occurred', err));
```

方法

constructor(0)

addEventListener(type, handler)

添加一个监听 Linking 变化的事件. type 参数应填 `url`， 并提供一个处理函数。

removeEventListener(type, handler)

删除一个监听函数, type 参数为 `url`.

openURL(url)

尝试使用设备上已经安装的应用打开指定的 `url`.

你可以用其他的 URLs，如地理位置（37.484847，- 122.148386”），或者通讯录和其他任何的网

址，直接打开即可。

注意：如果系统不知道如何处理给定的 url，则此方法会调用失败．如果你传入的 url 不是一个 http(s)

链接，则最好先通过{@code canOpenURL}方法检查一下.

注意：对于 Web 链接，协议头必须要有 ("http://", "https://") .

canOpenURL(url)

判断设备上是否有已经安装的应用可以处理指定的 `url`.

注意：对于 Web 链接，协议头必须要有 ("http://", "https://") .

注意：注：对于 iOS 9 以上版本，你还需要在 Info.plist 中添加 LSApplicationQueriesSchemes 字段.

@param URL 打开的网址

getInitialURL(0)

如果应用是被一个链接调起的，则会返回相应的链接地址．否则它会返回 `null`.

注意：了解更多 Android 上的深度链接

http://developer.android.com/training/app-indexing/deep-linking.html#handling-intents