如何用 RN 从 0 到 1 开发直播应用

buhe

为什么提供 RN sdk?

降低门槛

- iOS 和 Android 开发需要了解太多概念和技能
- 熟悉一套完全不同的 IDE 和编程语言
- 分别开发两个「看起来」一样的产品

用 RN 是一种什么体验?

react-native init awesome

创建项目

<View><Text>Hey</Text></View>

写一个例子，挺简单的

npm i awesome-component

开源组件好多，好赞

react-native link ?

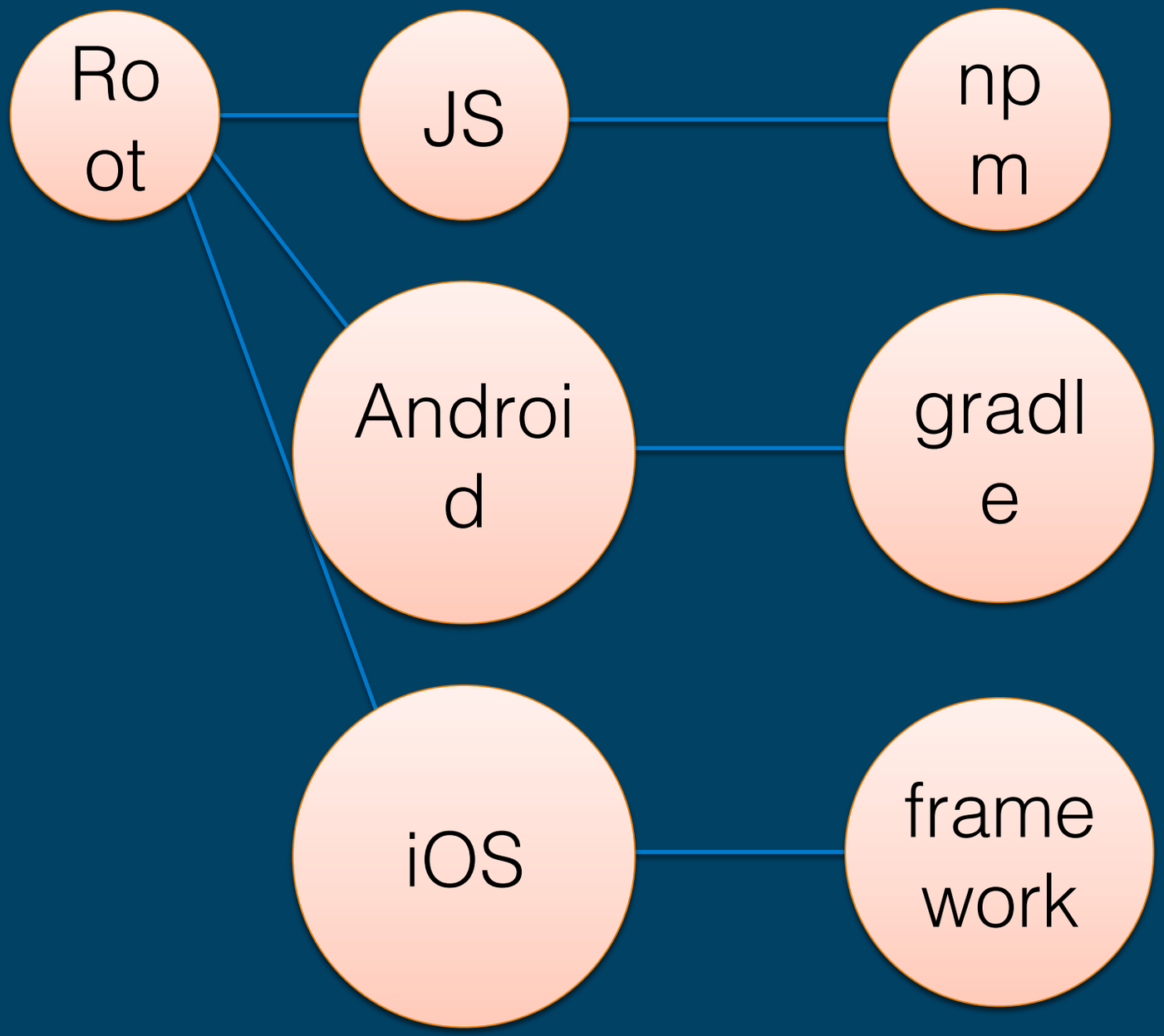
Native 组件什么鬼?

Java ? obj-c ???

Native API 什么鬼?

项目结构

React Native 原始结构



优点

- 所有代码在一个 repo，修改比较方便
- repo 权限管理简单

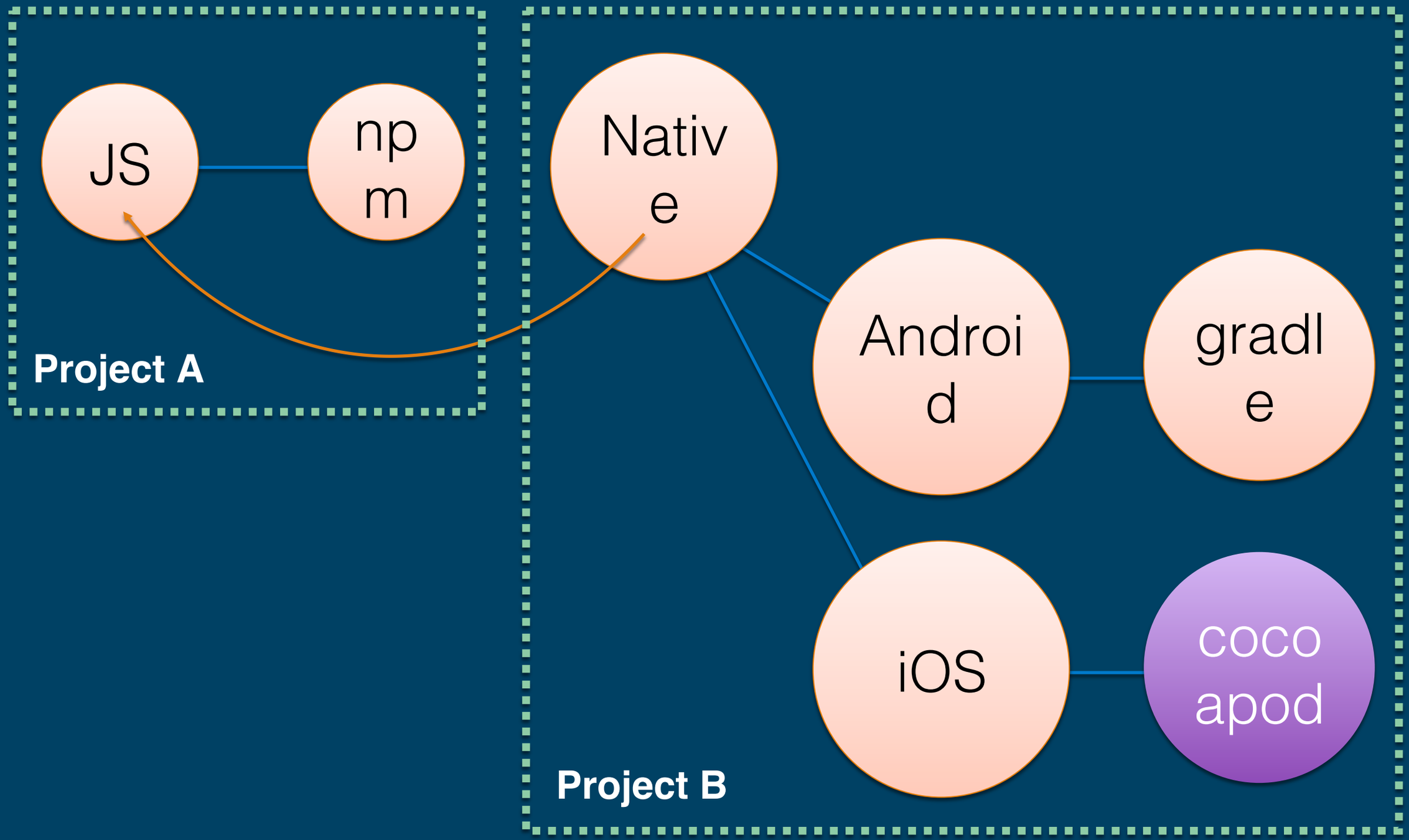
缺点

- 如果公司内部有多个项目，Native Library 部分需要集成多次，重复耗时的工作

- iOS 用了 framework 方式集成，比较难以自动化，集成比较复杂的 framework (如: pili) 非常困难

- 前端 JSer 理解 Native 部分很有难度，工作语言上下文需要频繁切换

新的项目结构


JS
npm
Project A
Native
Android
gradle
iOS
cocoapod
Project B

解决问题

- 内部多个项目共用一个 Native codebase，用工具自动化模块选择，降低 JSer 使用 RN 门槛

- Native 和 JS 项目分离，JSer 关注业务，鼓励用 pure JS 来实现功能，方便代码调试和多平台复用

- iOS 用 cocoapod 代替 framework 方式，更易于自动化构建和复杂类库集成

UI Kit

QINIU

UI Kit 选择

- NativeBase
- shoutem/ui
- React Native + Custom Styles

NativeBase 和 shoutem/ui

- 先设计统一的视觉 Guideline
- 修改 Theme 符合 Guideline
- 启动时间会比较长，但是随着项目的深入后续会有比较高的收益

React Native + Custom Styles

- 更灵活的布局和效果
- 比较适合没有统一视觉 Guideline， 像素还原设计师的原型
- 开始阶段比较容易做出原型

导航路由选择

导航路由

- https://github.com/exponentjs/ex-navigation
- https://github.com/wix/react-native-navigation
- https://github.com/aksonov/react-native-router-flux

react-native-navigation

- 优点
- 原生组件，体验流畅
- 支持 Navigator ,Tabbar,Drawer 等常见导航布局

react-native-navigation

- 缺点
  - 原生组件，需要使用原生代码
  - 需要修改原生项目的启动代码，对项目有侵入性

react-native-router-flux

- 优点
  - Flux 架构的原教旨主义，状态驱动导航
  - 支持 Modals，Tabbar，Custom navbar，Switch，Splitting Scenes，Drawer (side-menu)，Sub-Scenes
  - Pure JS 实现，Style 可定制性强

react-native-router-flux

- 缺点
  - Flux 架构的原教旨主义
  - 性能一般，动画不够流畅
  - 路由的 DSL 用来表达 Modal , Tabber 等语义不清晰
  - 所有界面的路由信息在一个地方被声明，侵入了页面的内部的逻辑，破坏了单个页面的内聚性。

react-native-router-flux

```
<Router createReducer={reducerCreate} getSceneStyle={getSceneStyle}>
  <Scene key="modal" component={Modal} >
    <Scene key="root" hideNavBar hideTabBar>
      <Scene key="echo" clone component={EchoView} getTitle={(navState) => navState.key} />
      <Scene
        key="switcher"
        component={Switch}
        selector={() => { return 'text1'; }}
      >
        <Scene
          key="text1"
          text="text1"
          component={(props) => <SwitcherPage
            {...props}
            text={currentSwitchPage}
          />}
        />
        <Scene
          key="text2"
          text="text2"
          component={(props) => <SwitcherPage
            {...props}
            text={currentSwitchPage}
          />}
        />
      </Scene>
      <Scene key="register" component={Register} title="Register" />
      <Scene key="register2" component={Register} title="Register2" duration={1} />
      <Scene key="home" component={Home} title="Replace" type={ActionConst.REPLACE} />
      <Scene key="launch" component={Launch} title="Launch" initial />
      <Scene key="login" direction="vertical" >
        <Scene key="loginModal" direction="vertical" component={Login} title="Login" />
```

ex-navigation

- 优点
  - Pure JS 实现，Style 可定制性强
  - 支持 Navigator ,Tabbar,Drawer 等常见导航布局
  - 可以和 redux 集成，但不强制要求
  - 路由 DSL 不表达整个 App 的页面结构

ex-navigation

- 缺点
  - 非原生控件，页面逻辑复杂会导致切换动画丢帧
  - Tabber 的切换响应速度不如原生体验

用到的 Native Library

react-native-pili

- 七牛直播云 React Native SDK
  - 视频推流、切换摄像头、丰富的参数调整
  - 音频推流
  - RTMP、FLV、HLS 流媒体播放
  - https://github.com/buhe/react-native-pili

react-native-pili

```
<Streaming
    rtmpURL={"rtmp://pili-publish.pilitest.qiniucdn.com/pilitest/demo_test?key=6eeee8a82246636e"}
    style={{
        height:400,
        width:400,
    }}
    zoom={1} //zoom
    muted={true} //muted
    focus={false} //focus
    profile={{  //video and audio profile
       video:{
         fps:30,
         bps:1000 * 1024,
         maxFrameInterval:48
       },
       audio:{
         rate:44100,
         bitrate:96 * 1024
       },
    started={false} //streaming status
    onReady={()=>{}} //onReady event
    onConnecting={()=>{}} //onConnecting event
    onStreaming={()=>{}} //onStreaming event
    onShutdown={()=>{}} //onShutdown event
    onIOError={()=>{}} //onIOError event
    onDisconnected={()=>{}} //onDisconnected event
    />
```

react-native-pili

```
<Player
  source={{
    uri:"rtmp://pili-live-rtmp.pilitest.qiniucdn.com/pilitest/xxx",
    timeout: 10 * 1000, //live streaming timeout (ms) Android only
    live:true, //live streaming ? Android only
    hardCodec:false, //hard codec [recommended false]  Android only
    }}
    started={true} //iOS only
    muted={false} //iOS only
    style={{
      height:200,
      width:200,
    }}
    onLoading={()=>{}} //loading from remote or local
    onPaused={()=>{}} //pause event
    onShutdown={()=>{}} //stopped event
    onError={()=>{}} //error event
    onPlaying={()=>{}} //play event
    />
```

react-native-config

- 配置分离，独立于 iOS 和 Android 工程文件 — .env
- 不同环境不同配置: prod ,dev

react-native-config

Declare config variables in `.env`:

```
API_URL=https://myapi.com
GOOGLE_MAPS_API_KEY=abcdefgh
```

Then access from your app:

```
import Config from 'react-native-config'

Config.API_URL  // 'https://myapi.com'
Config.GOOGLE_MAPS_API_KEY  // 'abcdefgh'
```

```
public HttpURLConnection getApiClient() {
    URL url = new URL(BuildConfig.API_URL);
    // ...
}
```

```
<meta-data
  android:name="com.google.android.geo.API_KEY"
  android:value="@string/GOOGLE_MAPS_API_KEY" />
```

```
// import header
#import "ReactNativeConfig.h"

// then read individual keys like:
NSString *apiUrl = [ReactNativeConfig envFor:@"API_URL"];

// or just fetch the whole config
NSDictionary *config = [ReactNativeConfig env];
```

其他

- air-umeng
- air-rongcloud

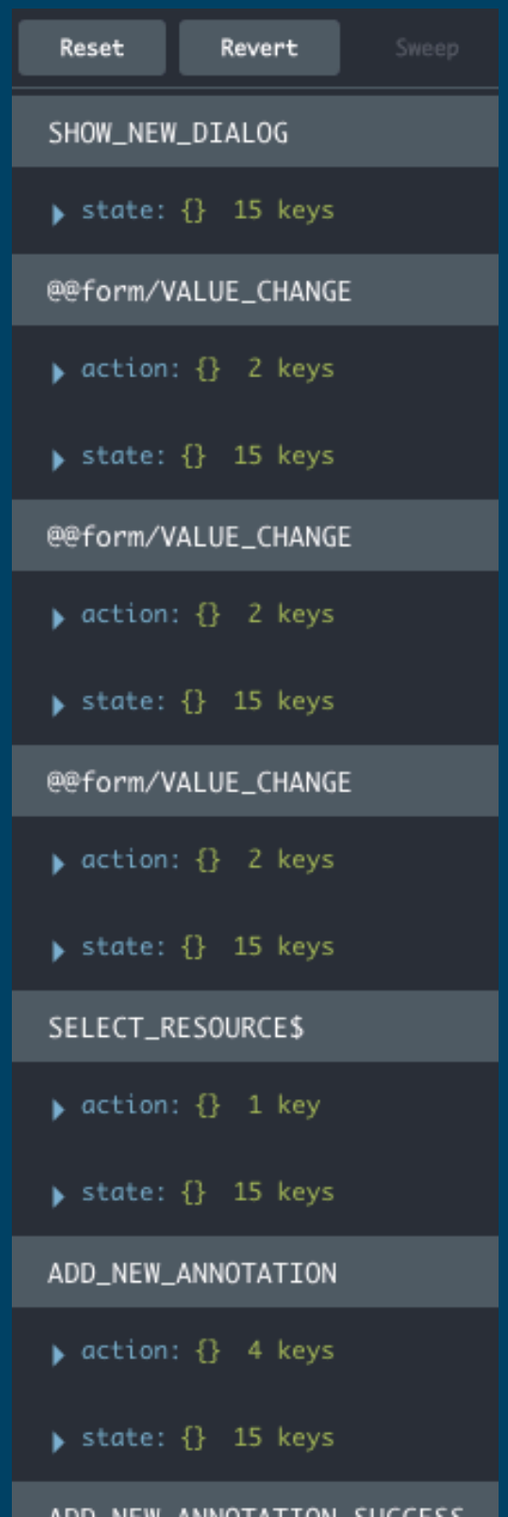
flux vs MV* ?

flux 架构

redux

- 优点
  - 状态描述业务流程
  - 状态描述视图
  - devtools， 免去大部分 debug 的烦恼

redux



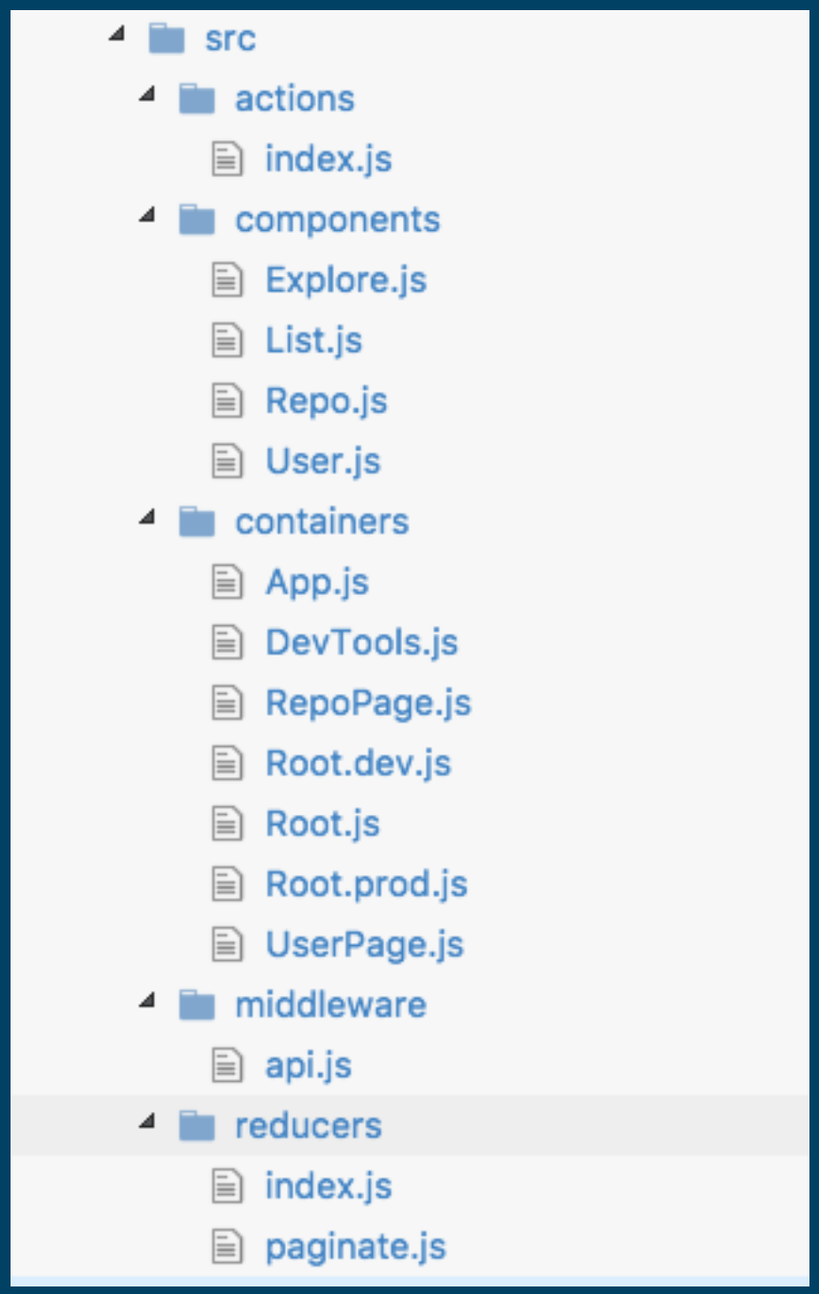
打开一个对话框 =>
输入一些表单信息 =>
提交 =>
提交成功 =>
关闭对话框 =>
重新加载列表 =>
列表加载成功

QINIU

redux

- 缺点
- 代码量大
- 写一个业务需要切换不同源文件
- 对原有编程思维的冲击

redux


src
actions
index.js
components
Explore.js
List.js
Repo.js
User.js
containers
App.js
DevTools.js
RepoPage.js
Root.dev.js
Root.js
Root.prod.js
UserPage.js
middleware
api.js
reducers
index.js
paginate.js

redux-thunk

- redux 生态中解决程序副作用最简单的实现
- redux-promise
- redux-saga

redux-thunk

```
function makeSandwichesForEverybody() {
  return function (dispatch, getState) {
    if (!getState().sandwiches.isShopOpen) {

      // You don't have to return Promises, but it's a handy convention
      // so the caller can always call .then() on async dispatch result.

      return Promise.resolve();
    }

    // We can dispatch both plain object actions and other thunks,
    // which lets us compose the asynchronous actions in a single flow.

    return dispatch(
      makeASandwichWithSecretSauce('My Grandma')
    ).then(() =>
      Promise.all([
        dispatch(makeASandwichWithSecretSauce('Me')),
        dispatch(makeASandwichWithSecretSauce('My wife'))
      ])
    ).then(() =>
      dispatch(makeASandwichWithSecretSauce('Our kids'))
    ).then(() =>
      dispatch(getState().myMoney > 42 ?
        withdrawMoney(42) :
        apologize('Me', 'The Sandwich Shop')
      )
    );
  };
}
```

延伸阅读

- 看漫画学：redux https://github.com/jasonslyvia/a-cartoon-intro-to-redux-cn

- flux vs mvc：https://zhuanlan.zhihu.com/p/21324696?refer=front-end

Author

buhe

github.com/buhe

wechat: 81128054

email: buhe@qiniu.com

focus: React Native

QINIU