React Native 的定制与优化

殷文昭

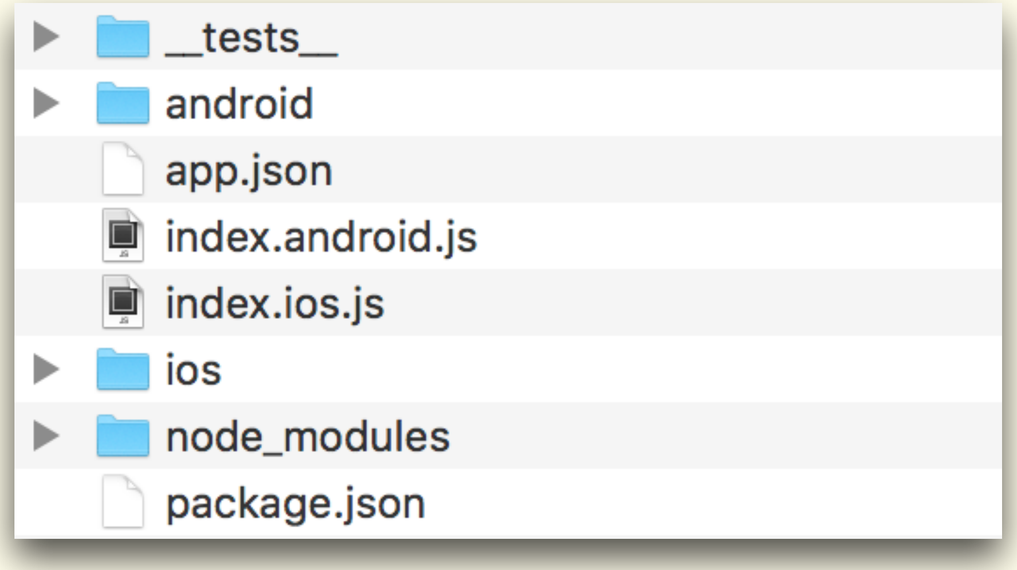
概要

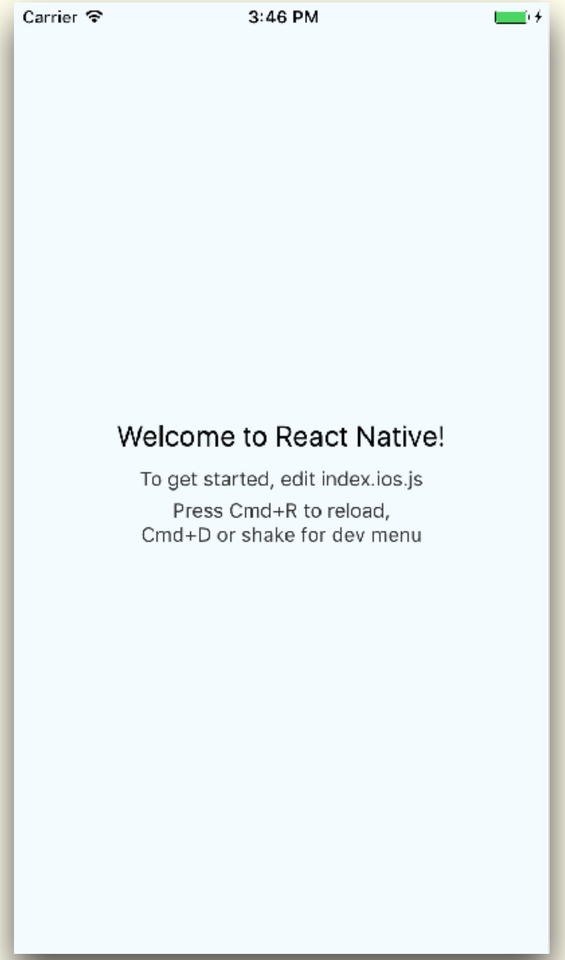
1. React Native 项目的快速搭建
2. React Native 的组件详解与分析
3. 状态管理框架 Redux 的应用
4. React Native 跨平台的定制与优化
5. 现有 App 快速引入 React Native 实践

React Native 项目的快速搭建

react-native-cli 工具快速搭建

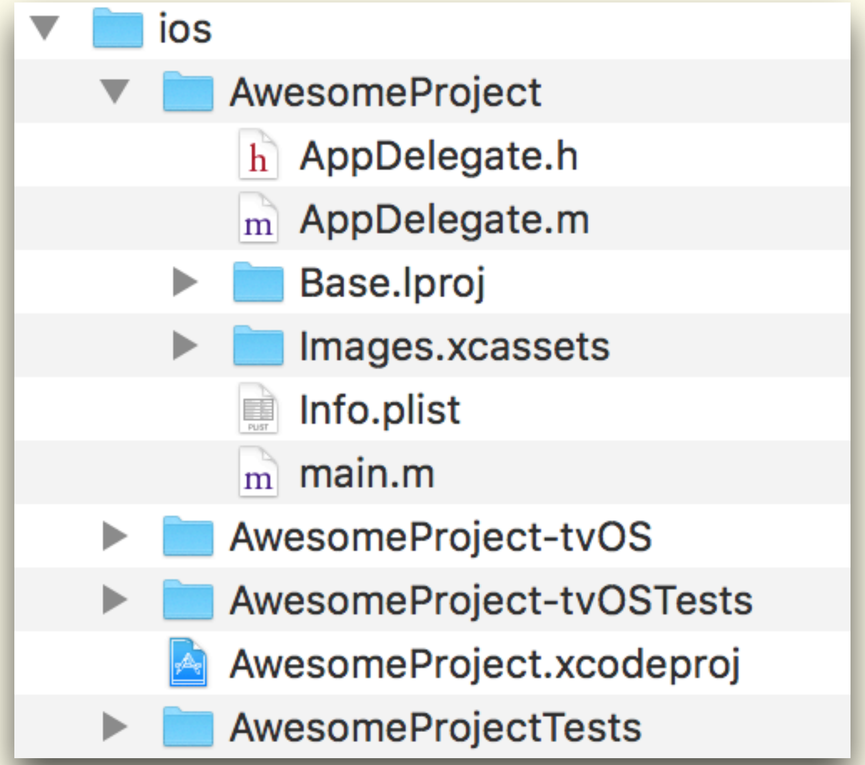
react-native init AwesomeProject



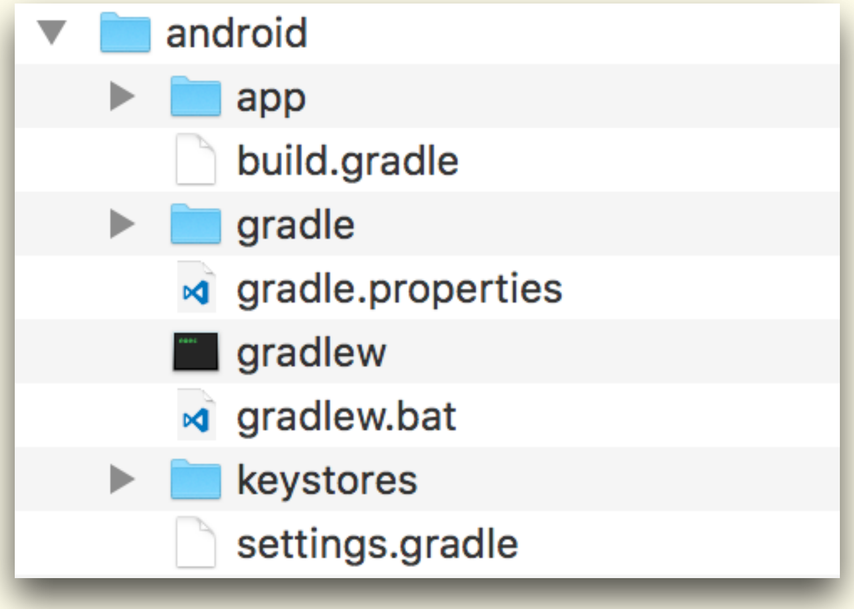


http://reactnative.cn/docs/0.41/getting-started.html#content

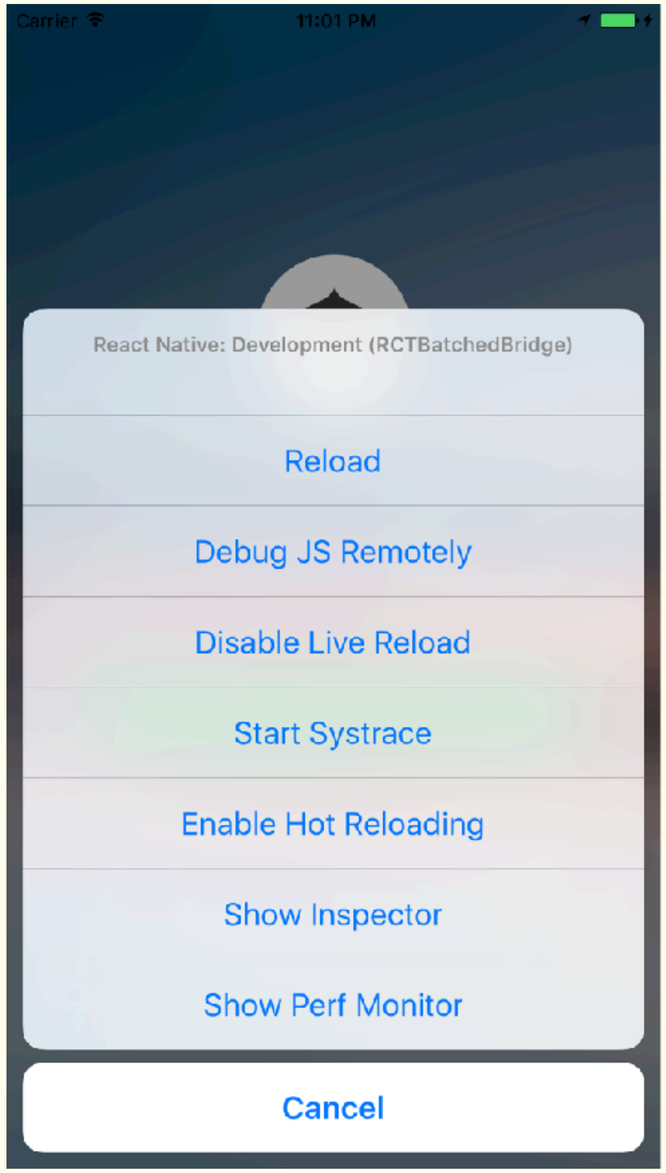
简单的APP壳



完全 RN 化的App



简单的APP壳



APP仅作为画板

强大的调试工具

集成到现有 APP

作为第三方Lib

创建View 显示在屏幕上

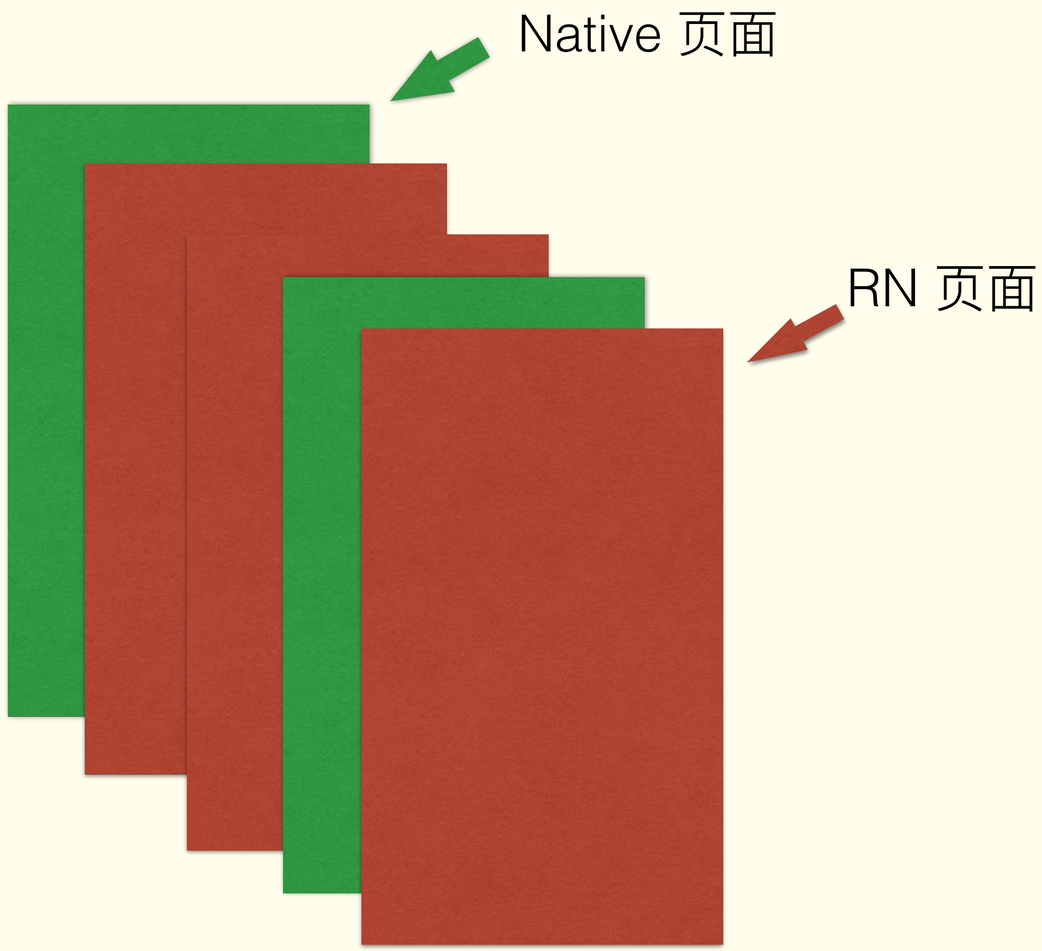
cocoapod

一个 ViewController 中创建一个 RCTRootView

gradle

一个 Activity 中创建一个 ReactRootView 对象

集成到现有 APP



部分页面使用 ReactNative

ReactNaive 作为一个布局引擎

React Native 的组件详解与分析

React Native Component

COMPONENTS

ActivityIndicator
Button
DatePickerIOS
DrawerLayoutAndroid
Image
KeyboardAvoidingView
ListView
MapView
Modal
Navigator
NavigatorIOS
Picker
PickerIOS
ProgressBarAndroid
ProgressViewIOS
RefreshControl
ScrollView
SegmentedControlIOS
Slider
SnapshotViewIOS
StatusBar
Switch
TabBarIOS
TabBarIOS.Item
Text
TextInput
ToolbarAndroid
TouchableHighlight
TouchableNativeFeedback
TouchableOpacity
TouchableWithoutFeedback
View
ViewPagerAndroid
WebView

React Native Component

COMPONENTS

- ActivityIndicator
- Button
- DatePickerIOS
- DrawerLayoutAndroid
- Image
- KeyboardAvoidingView
- ListView
- MapView
- Modal
- Navigator
- NavigatorIOS
- Picker
- PickerIOS
- ProgressBarAndroid
- ProgressViewIOS
- RefreshControl
- ScrollView
- SegmentedControlIOS
- Slider
- SnapshotViewIOS
- StatusBar
- Switch
- TabBarIOS
- TabBarIOS.Item
- Text
- TextInput
- ToolbarAndroid
- TouchableHighlight
- TouchableNativeFeedback
- TouchableOpacity
- TouchableWithoutFeedback
- View
- ViewPagerAndroid
- WebView

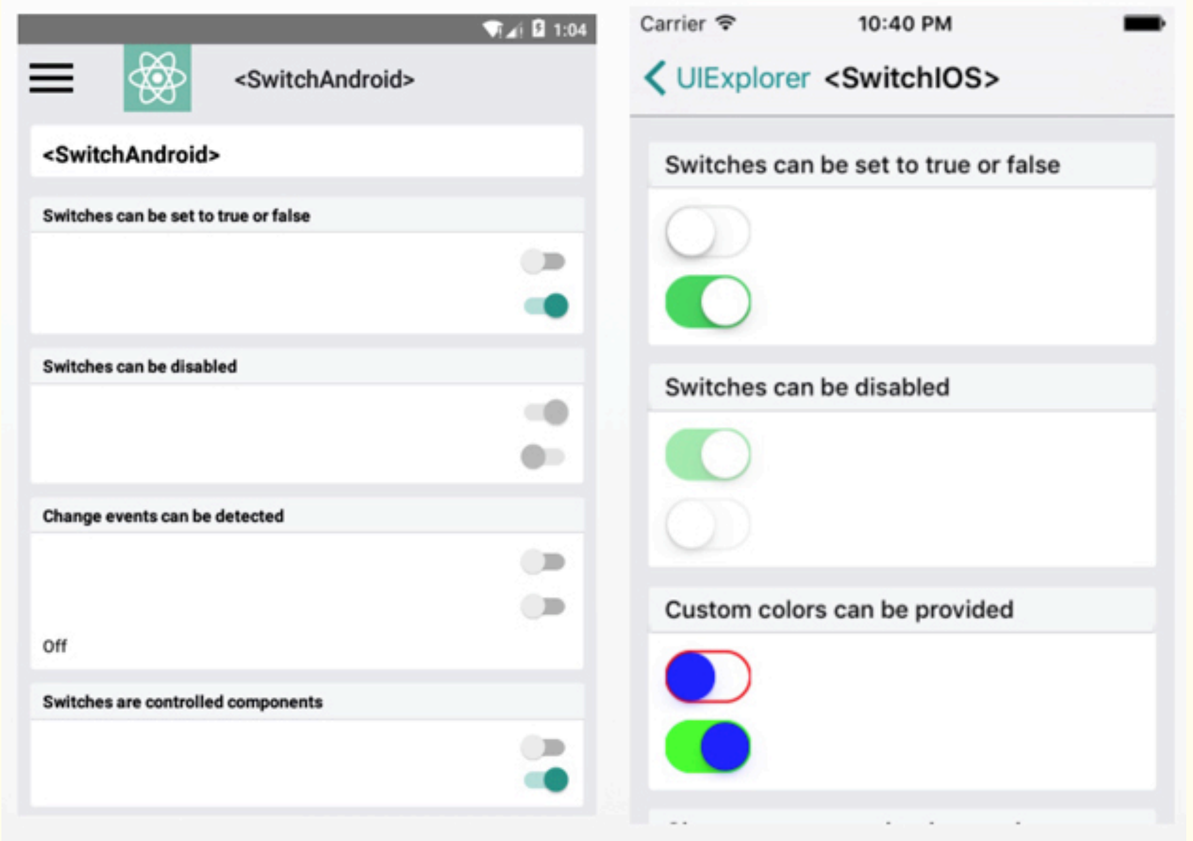
React Native Component

基础组件	View	Image	Text	TextInput	ActivityIndicator
核心组件	ScrollView	ListView	Navigator	Modal	Slider	WebView
手势组件	TouchableWithoutFeedback	TouchableOpacity	TouchableHighlight		
平台特征	Switch	Picker	RefreshControl		

React Native Component

Switch

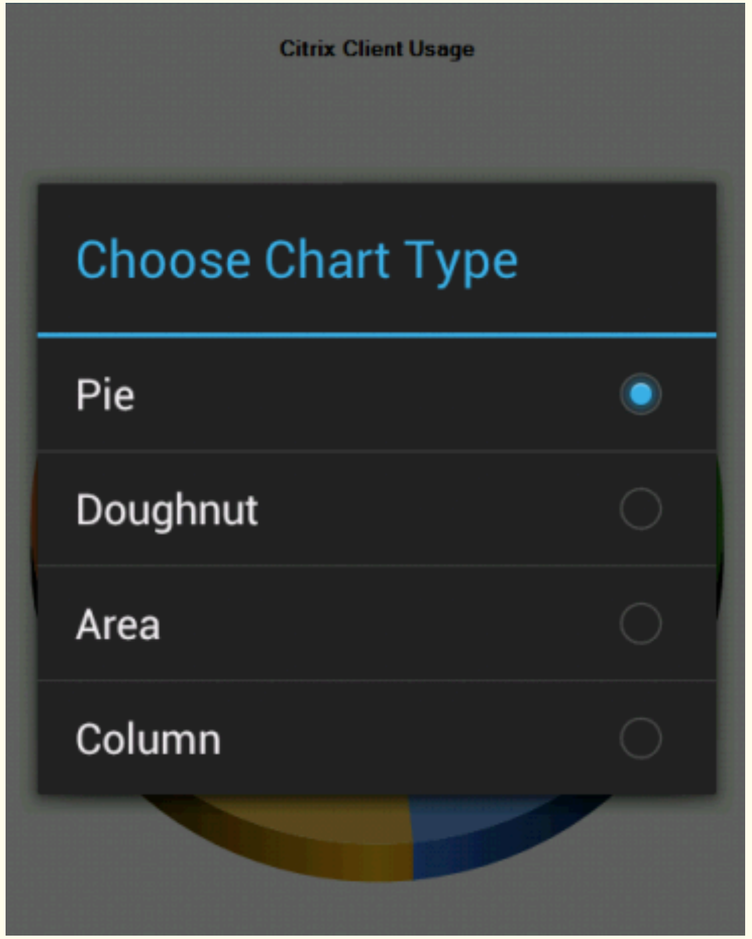
平台特征



React Native Component

Picker

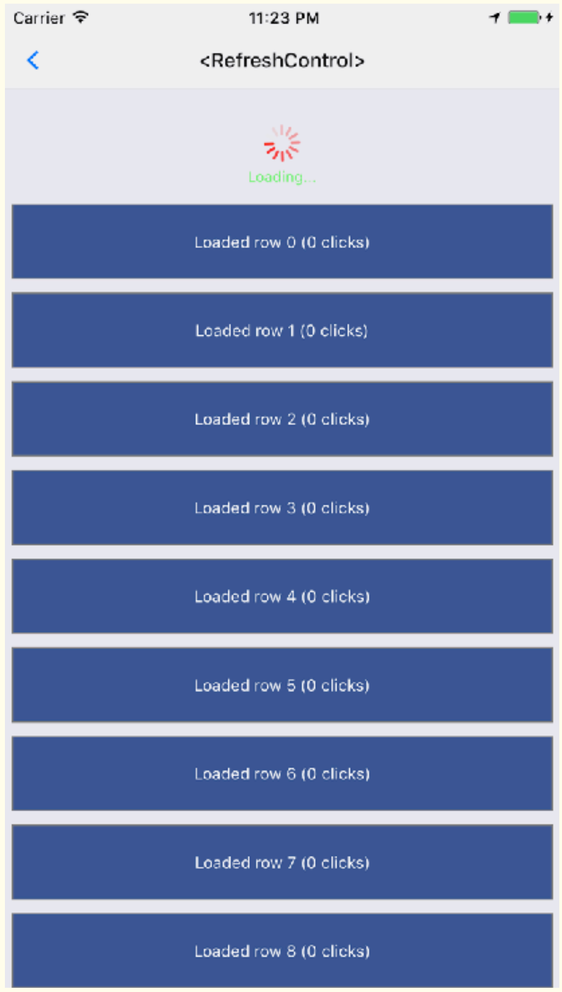
平台特征

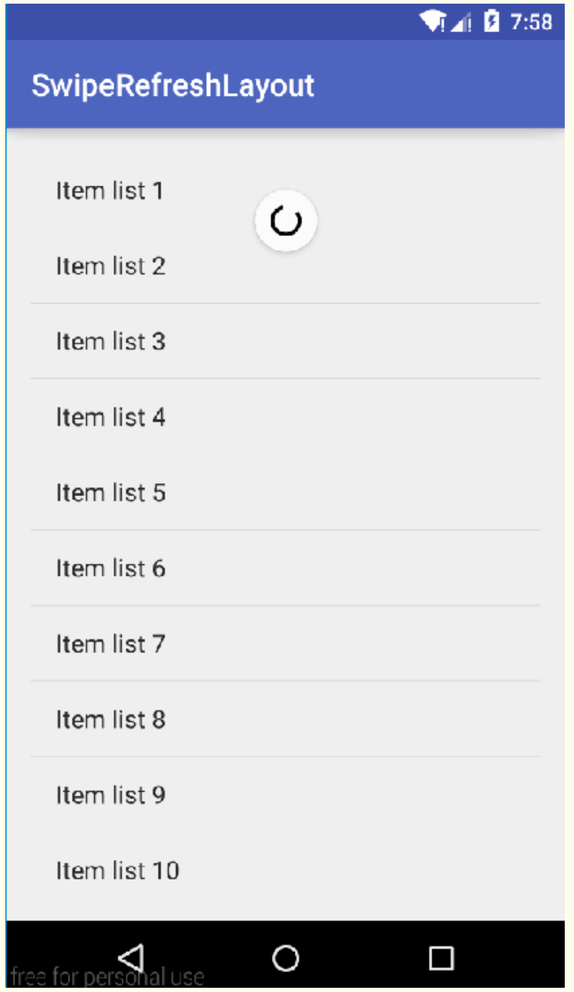


React Native Component

RefreshControl

平台特征

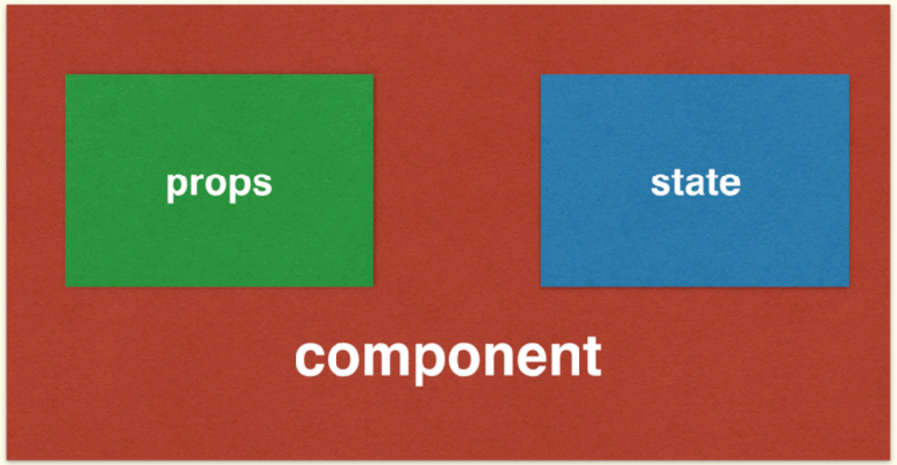




状态管理框架 Redux 的应用

props 与 state

props
父组件 => 子组件
数据流

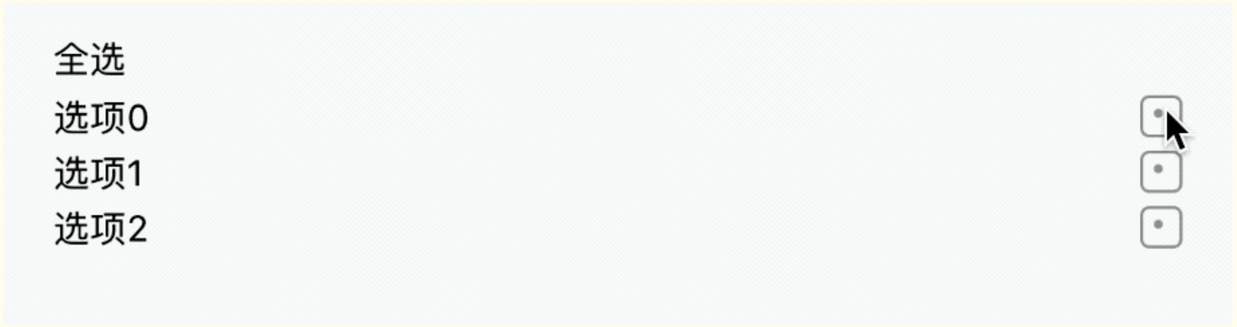


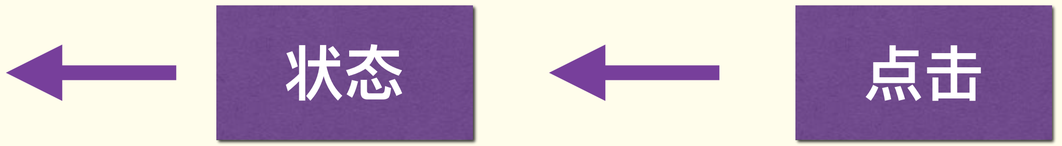
state
组件内部状态

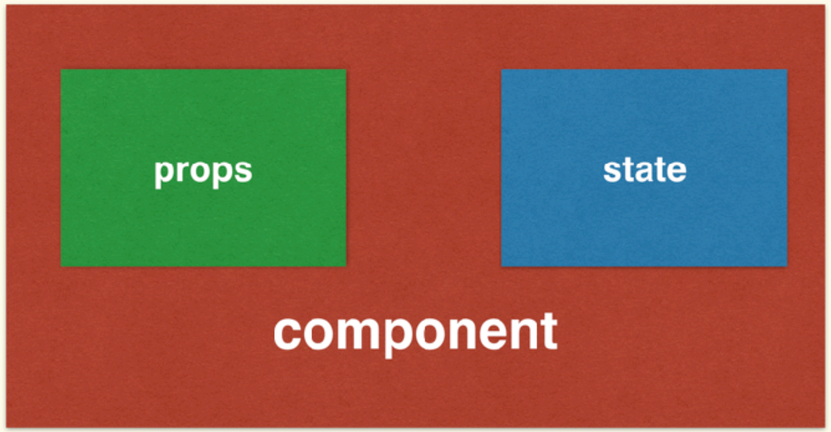
props 与 state

视图

操作





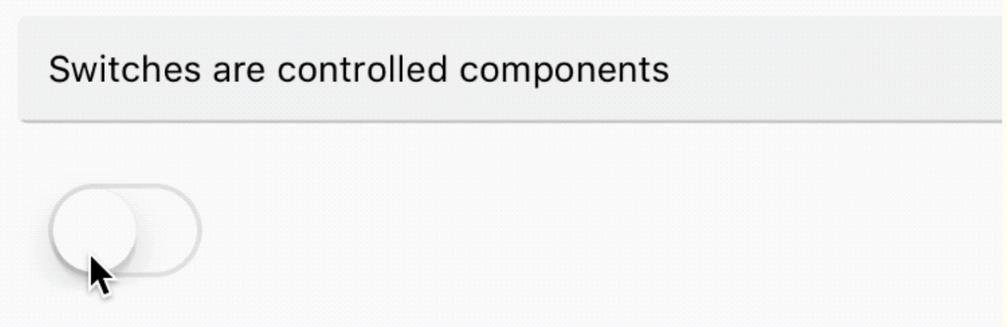


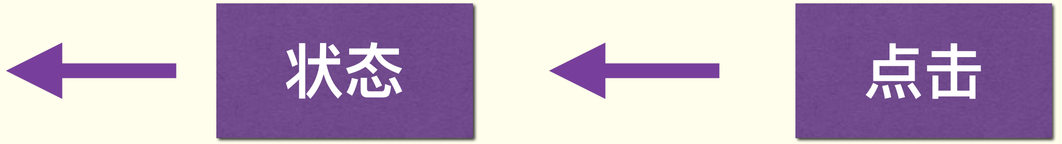
react 中的单向数据流

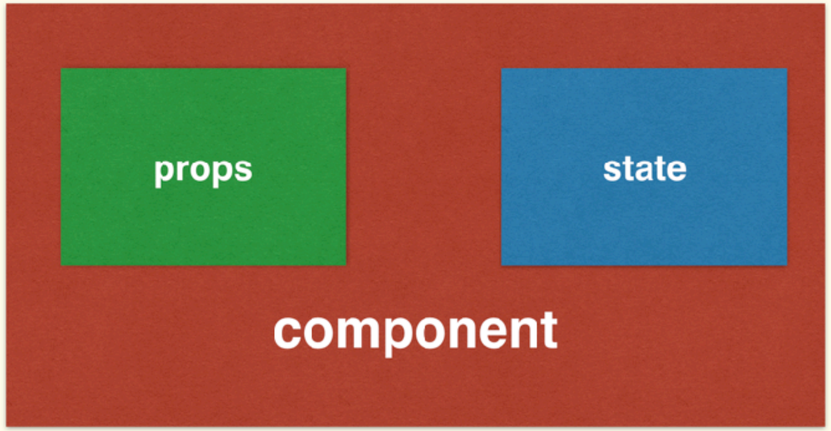
props 与 state

视图

操作



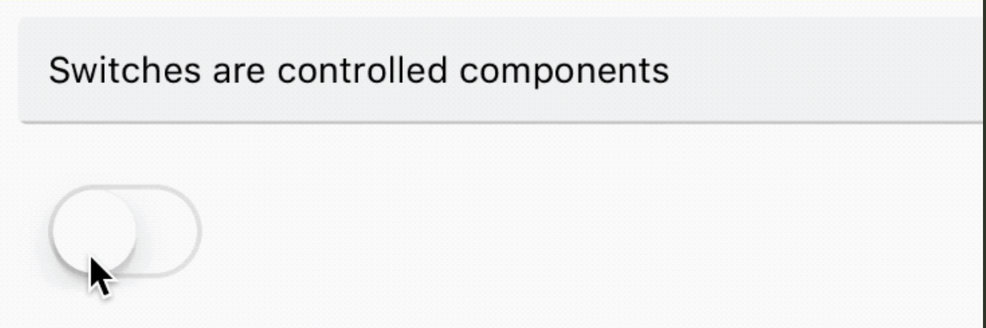


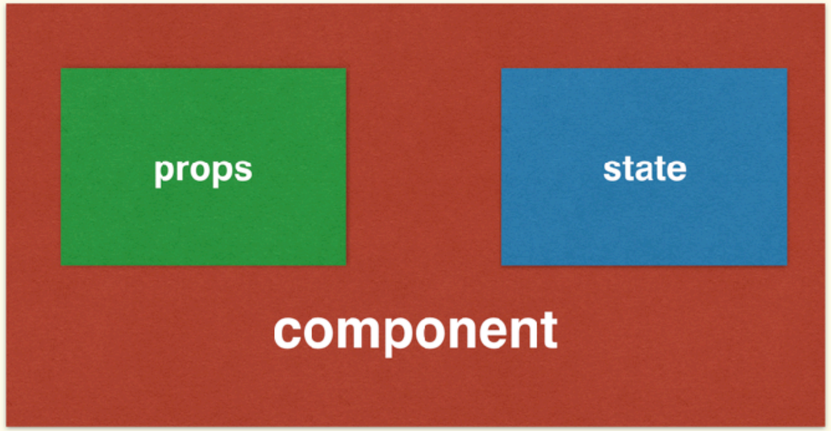


视图需要是受控的组件

props 与 state

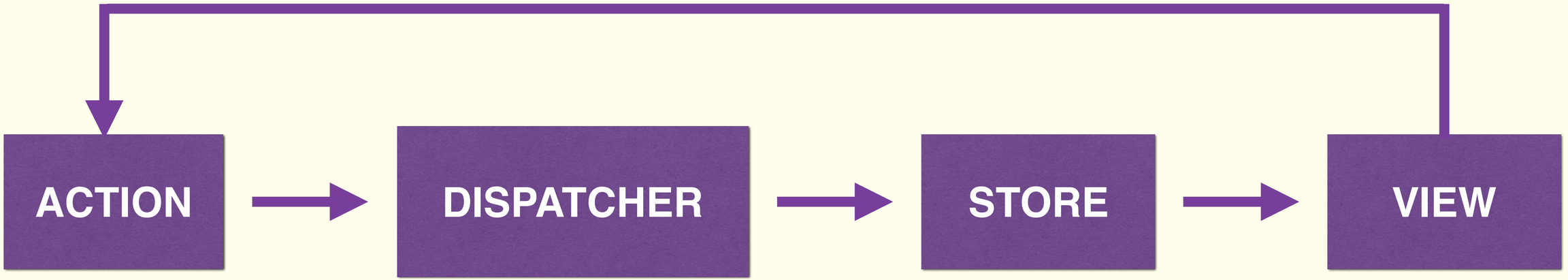
视图






```
88    _onChange: function(event: Object) {
89      if (Platform.OS === 'android') {
90        this._rctSwitch.setNativeProps({on: this.props.value});
91      } else {
92        this._rctSwitch.setNativeProps({value: this.props.value});
93      }
94      //Change the props after the native props are set in case the props change ren
    the component
95      this.props.onChange && this.props.onChange(event);
96      this.props.onValueChange && this.props.onValueChange(event.nativeEvent.value);
97    },
98
99    render: function() {
100     var props = {...this.props};
101     props.onStartShouldSetResponder = () => true;
102     props.onResponderTerminationRequest = () => false;
103     if (Platform.OS === 'android') {
104       props.enabled = !this.props.disabled;
105       props.on = this.props.value;
106       props.style = this.props.style;
107     } else if (Platform.OS === 'ios') {
108       props.style = [styles.rctSwitchIOS, this.props.style];
109     }
110     return (
111       <RCTSwitch
112         {...props}
113         ref={(ref) => { this._rctSwitch = ref; }}
114         onChange={this._onChange}
115       />
116     );
```

单向数据流的架构模式 — Flux



Redux 三大原则

单一数据源Store

State 是只读的

使用纯函数修改

Redux 的引入

解决多交互、多数据源问题

负责的用户交互逻辑

多个数据来源控制View

React Native 跨平台的定制与优化

组件的优化

平台特征

Switch　　Picker　　RefreshControl

组件优化

ScrollView　　ListView　　Navigator

组件的优化

ScrollView

组件优化

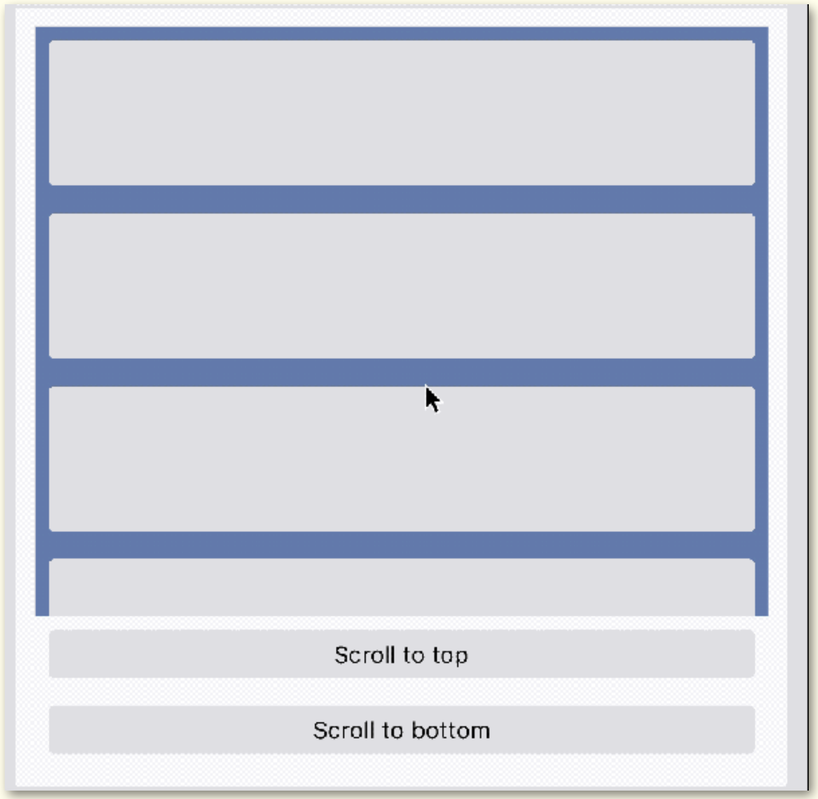
	官方 ScrollView	QRN ScrollView	React Native 普通组件
触摸事件 处理	**Native**	JS	JS

复杂组件的嵌套会出现行为怪异

组件的优化

ScrollView

组件优化



滑动手势发送给JS

JS设置 Scroll Position

JS
Core

组件的优化

ScrollView

类似于iOS的下拉风格

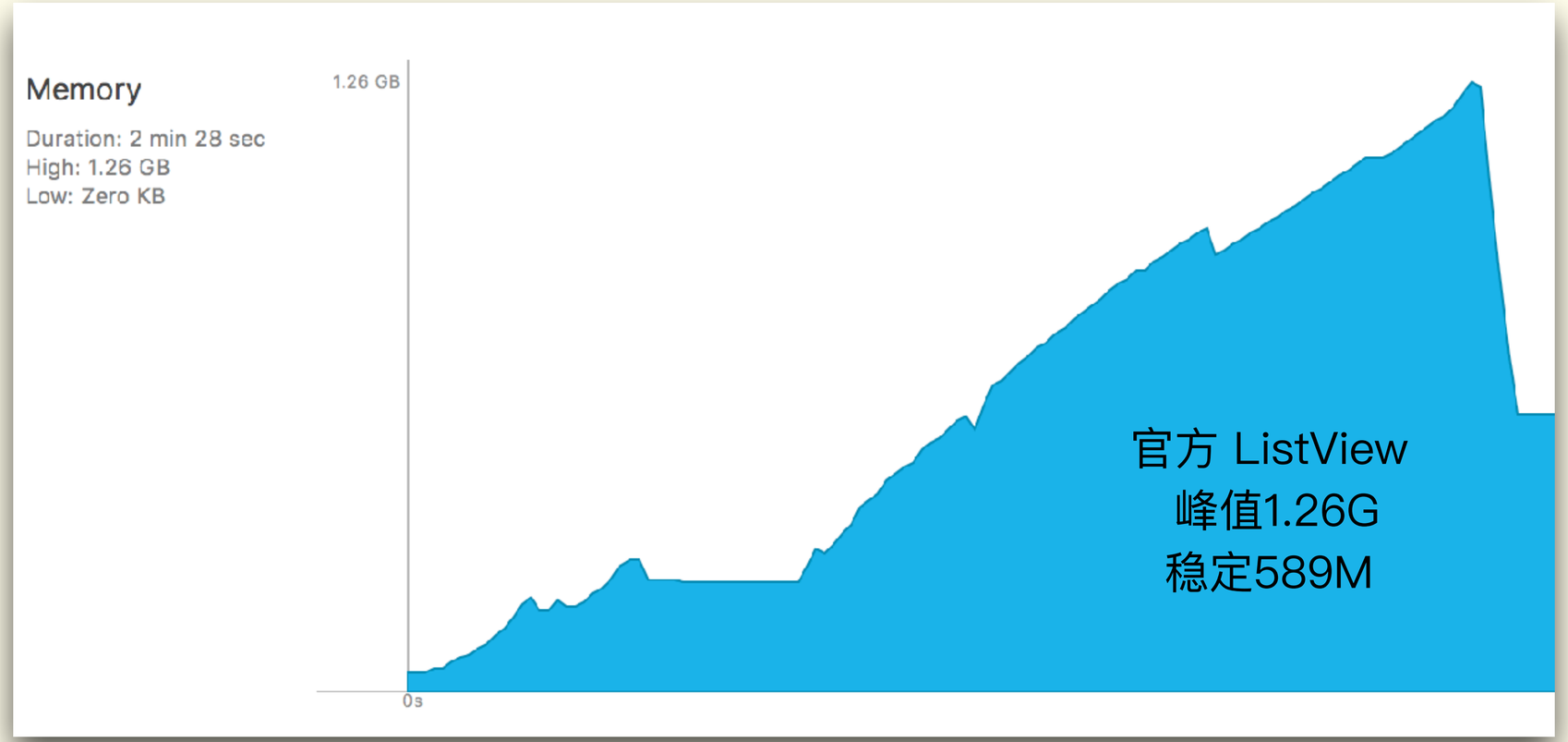
组件优化

更加流弊的上拉、下拉组件

组件的优化

ListView

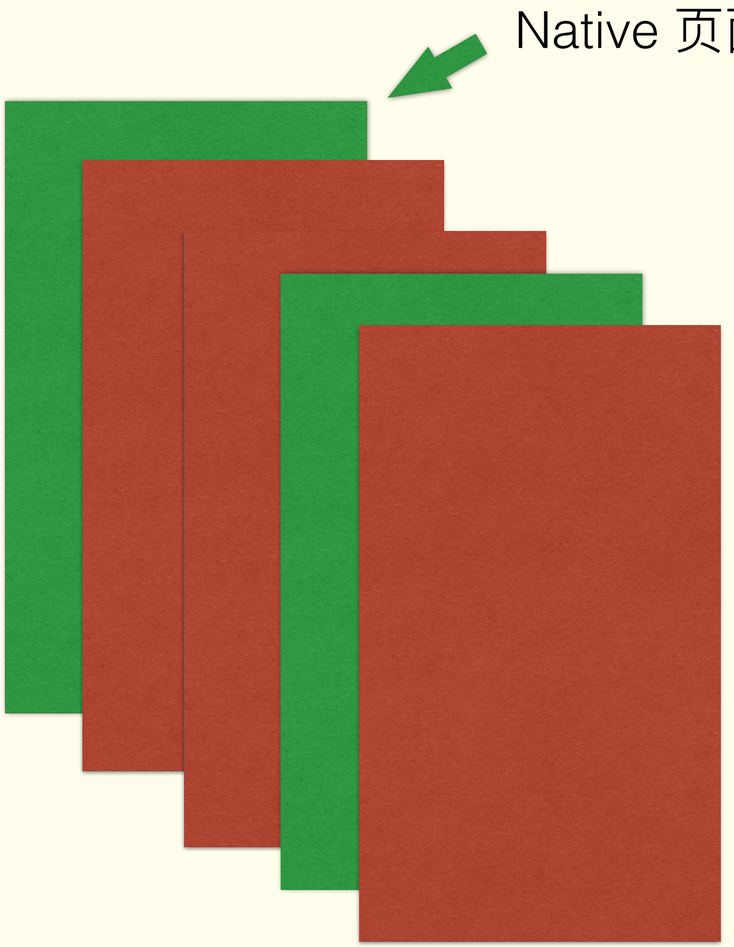
组件优化



2000行数据

组件的跨平台性

Navigator

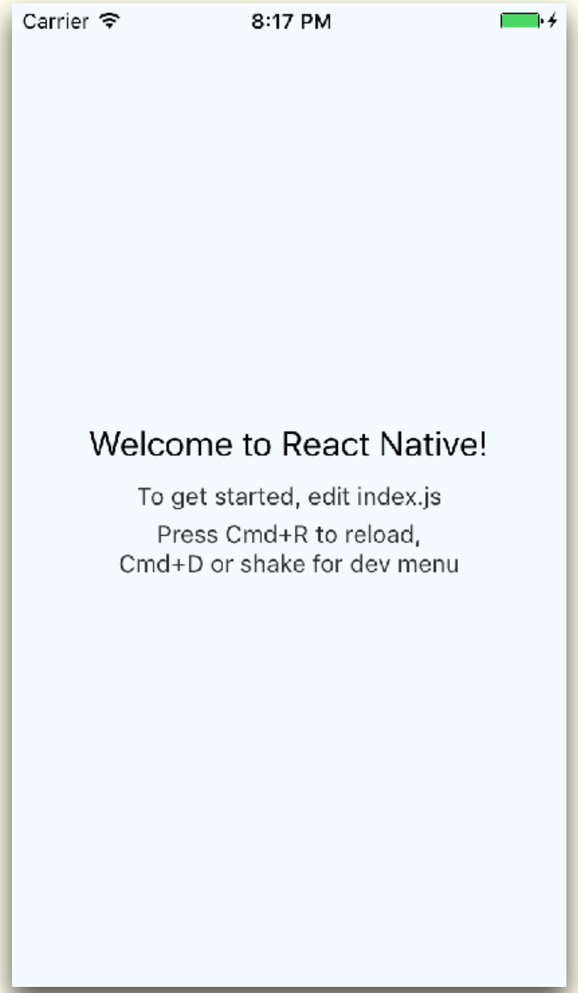


局限于JS页面栈

组件优化

需要提供更强大路由

React Native 的 Bridge



RootView

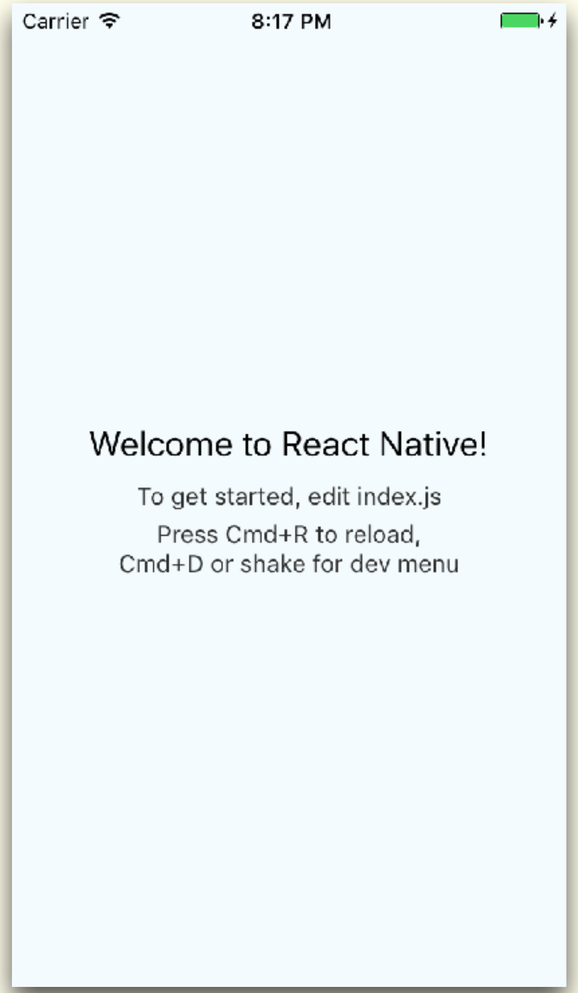
Bridge

RN 环境

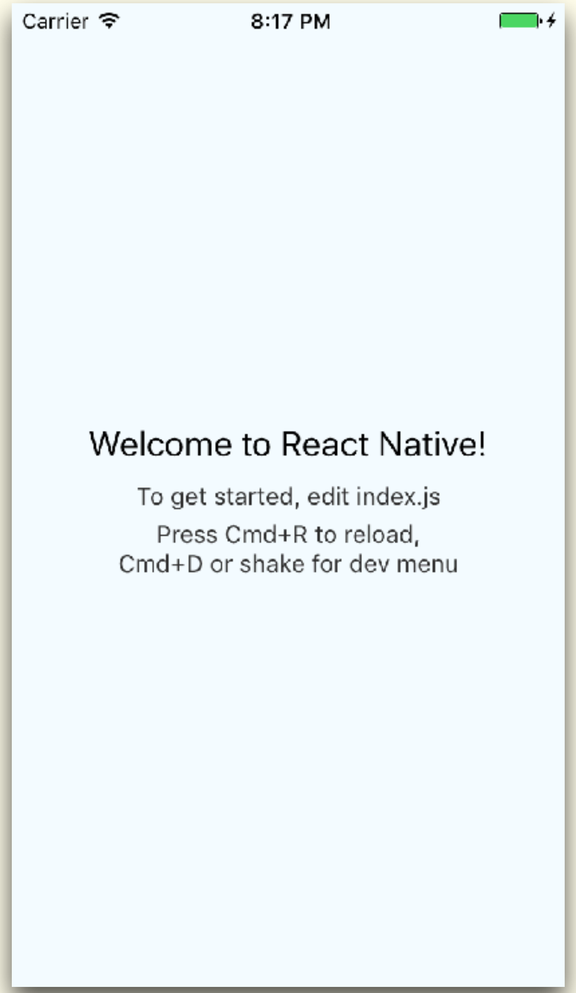
JS Core

插件

React Native 的 Bridge

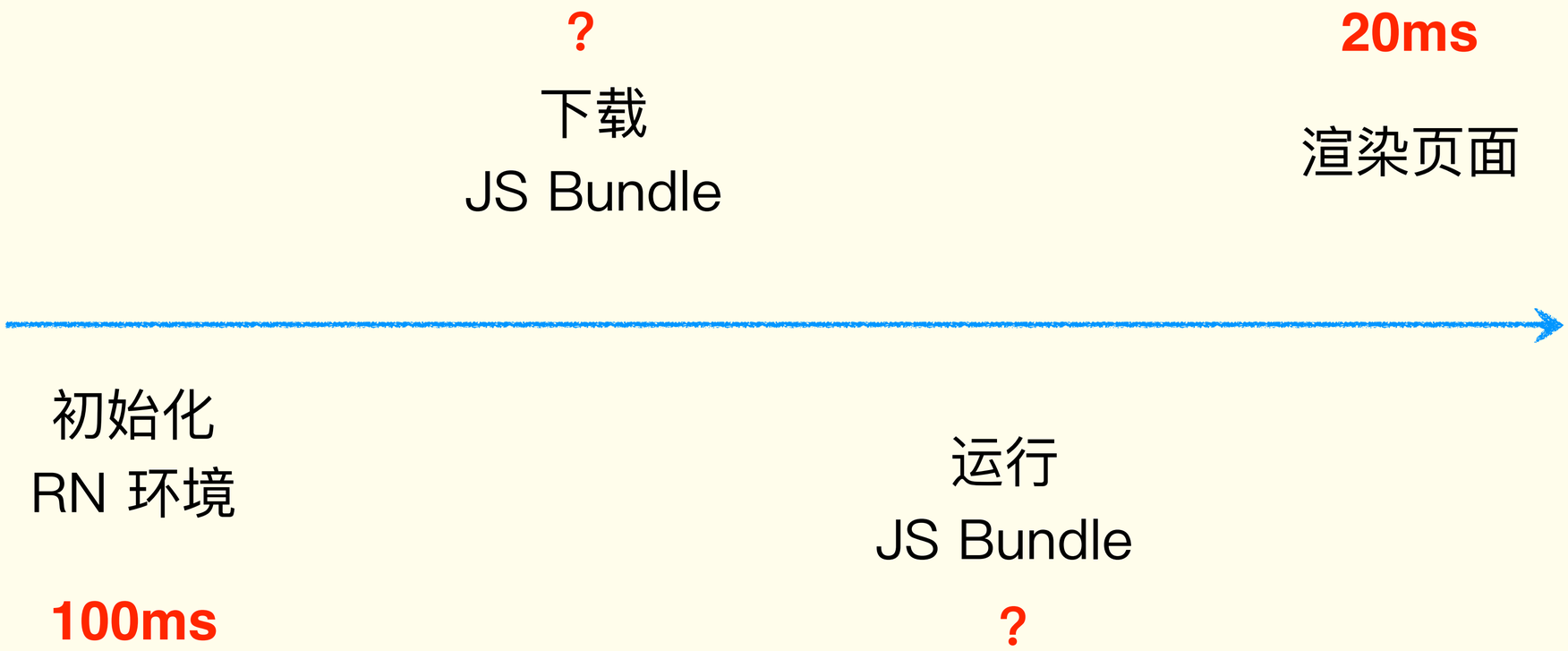


RootView

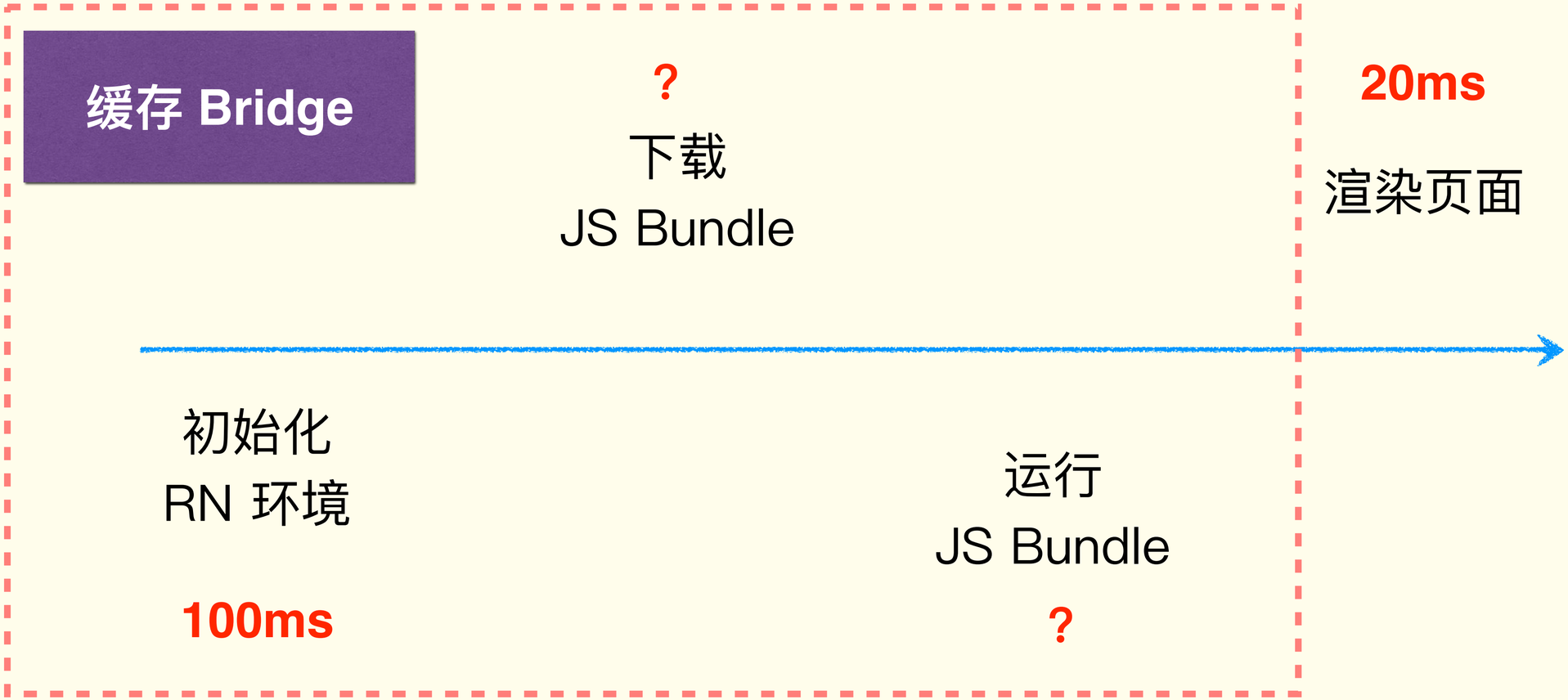


RootView

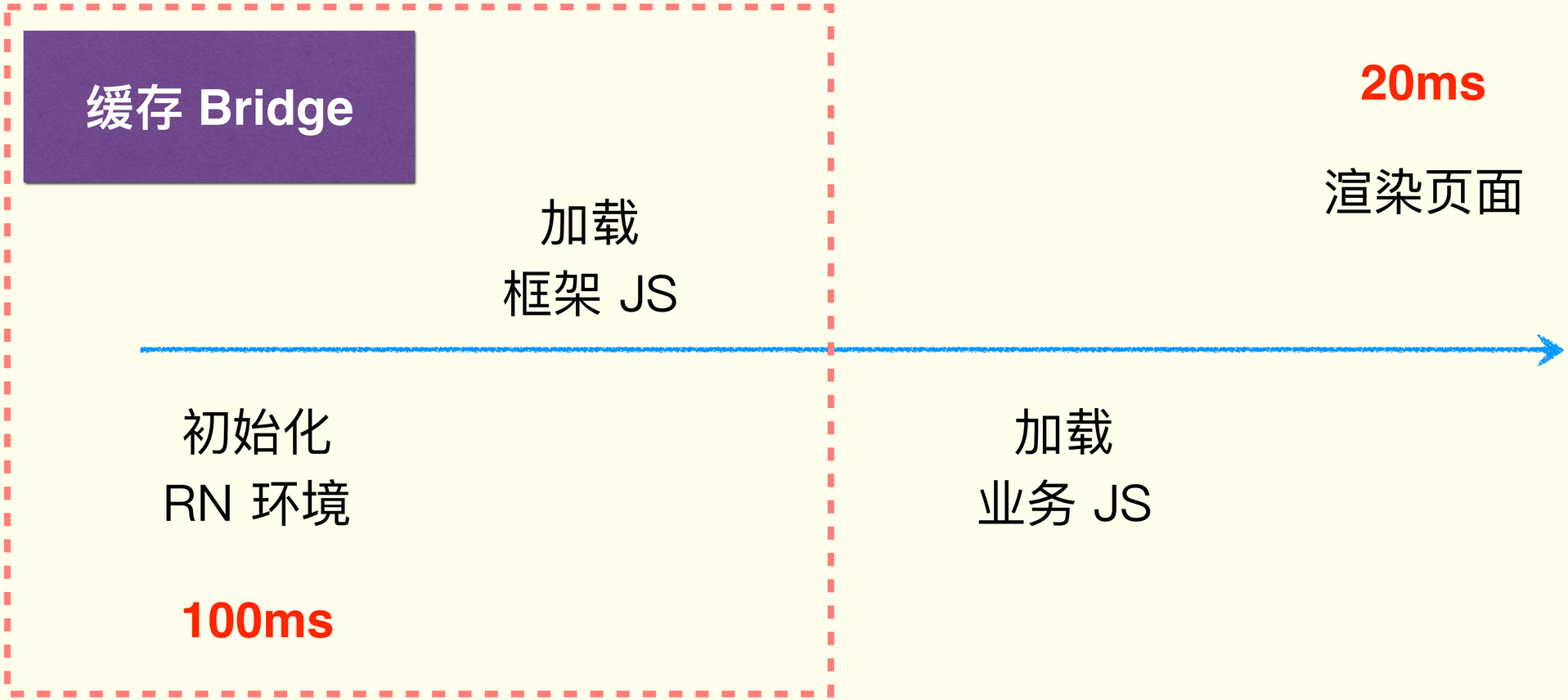
ReactNative 加载流程



ReactNative 加载流程



ReactNative 加载流程



减少运行 JS Bundle 时间

```
29 const ReactNative = {
30   // Components
31   get ActivityIndicator() { return require('ActivityIndicator'); },
32   get ActivityIndicatorIOS() { return require('ActivityIndicatorIOS'); },
33   get ART() { return require('ReactNativeART'); },
34   get DatePickerIOS() { return require('DatePickerIOS'); },
35   get DrawerLayoutAndroid() { return require('DrawerLayoutAndroid'); },
36   get Image() { return require('Image'); },
37   get ImageEditor() { return require('ImageEditor'); },
38   get ImageStore() { return require('ImageStore'); },
39   get KeyboardAvoidingView() { return require('KeyboardAvoidingView'); },
40   get ListView() { return require('ListView'); },
```

...

```
182 };
```

```
const ReactNative = {
  Image: require('Image'),
  View: require('View'),
  Text: require('Text'),
}
```

Libraries/react-native/react-native.js

减少运行 JS Bundle 时间

```
94  global.require = require;
```

只初始化一次

```
25  function require(id) {
26    var mod = modules[id];
27    if (mod && mod.isInitialized) {
28      return mod.module.exports;
29    }
30
31    return requireImpl(id);
32  }
```

packager/react-pakager/src/Resolve/polyfills/require.js

注册页面的方式

```
6  import React, {
7    View,
8    Component,
9    AppRegistry
10 } from 'react-native';
11
12 import { Provider } from 'react-redux/native';
13
14 import MyHotel from './src/containers/MyHotel/MyHotelContainer';
15 import Hotel from './src/containers/Hotel/HotelContainer';
16 import HotelDetailMap from './src/containers/HotelDetailMap';
17
18 class HotelDetailMapRoot extends Component {
19     render() {
20         return (
21             <Provider store={store}>
22                 <HotelDetailMap {...this.props}/>
23             </Provider>
24         )
25     }
26 }
```

基础组件

Redux

页面

注册页面的方式

```
118  AppRegistry.registerComponent('MyHotel',
119      () => MyHotel);
120  AppRegistry.registerComponent('Hotel',
121      () => Hotel);
122  AppRegistry.registerComponent('HKeyword',
123      () => HKeywordRoot);
124  AppRegistry.registerComponent('DetailImage',
125      () => DetailImageRoot);
126  AppRegistry.registerComponent('RecommendList',
127      ()=>HotelRecommandListRoot);
```

注册多达20个页面

注册页面的方式

渲染页面的开始

moduleName

```
AppRegistry.runApplication('DetailImage', {
  initialProps:{}
});
```

```
118  AppRegistry.registerComponent('MyHotel',
119      () => MyHotel);
120  AppRegistry.registerComponent('Hotel',
121      () => Hotel);
122  AppRegistry.registerComponent('HKeyword',
123      () => HKeywordRoot);
124  AppRegistry.registerComponent('DetailImage',
125      () => DetailImageRoot);
126  AppRegistry.registerComponent('RecommendList',
127      ()=>HotelRecommandListRoot);
```

异步加载

```
AppRegistry.registerComponent('DetailImage', () => {
    const DetailImage  = require('./src/containers/DetailImage/
    DetailImage');

    class DetailImageRoot extends Component {
        render() {
            return (
                <Provider store={store}>
                    <DetailImage {...this.props}/>
                </Provider>
            )
        }
    }

    return DetailImageRoot;
});
```

只有渲染的页面
才初始化

现有 App 快速引入 React Native 实践

谢谢大家~　　Q&A