

React Native

React Native

STEPS

1. 简介
2. 搭建开发环境
3. 基础语法
4. 架构原理
5. 项目实践

简介

简介

- React Native 是 Facebook 在 React.js Conf 2015 上推出了开源框架
 - React Native（简称 RN）是 React 的一个原生（Native）扩展
 - 它允许我们通过 React 语法，来开发 iOS 和 Android 原生应用
- 资源网站
 - 官网：<https://reactnative.dev/>
 - Github：<https://github.com/facebook/react-native>
 - 中文网：<https://www.reactnative.cn/>

移动 App 的开发模式

原生开发

原生 App

Android | iOS | Windows

移动应用原生开发



Android



iOS



Windows

移动 App 的开发模式

原生开发

原生 App

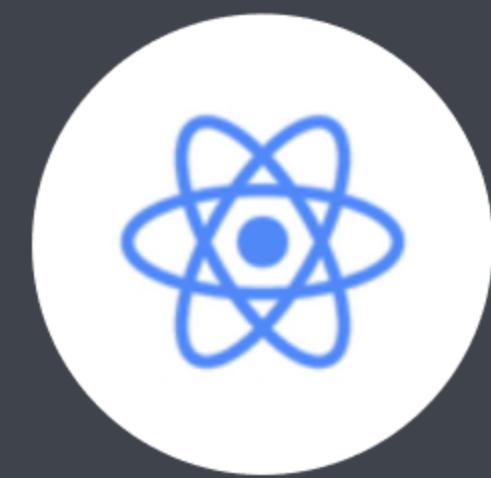
Android | iOS | Windows

混合开发

混合 App

React Native | Weex | Flutter

混合开发框架



React Native



Weex



Flutter

跨平台框架的比较

框架	React Native	Weex	Flutter
所属公司	Facebook	Alibaba	Google
编程语言	JavaScript (React)	JavaScript (Vue)	Dart
引擎	JSCore	V8	Flutter engine
支持系统	Android、iOS	Android、iOS、Web	Android、iOS、Fuchsia
性能	一般	较快	较快
适用场景	整体 App	单页面	整体 App
学习成本	难	易	一般

移动 App 的开发模式

原生开发

原生 App

Android | iOS | Windows

混合开发

混合 App

React Native | Weex | Flutter

H5 开发

Web App

HTML、CSS、JavaScript

移动 App 的开发模式

开发模式	原生开发	混合开发	Web 开发
运行环境	Android、iOS	Android、iOS	浏览器、WebView
编程语言	Java、Objective-C	JavaScript、Dart	HTML、CSS、Javascript
可移植性	差	一般	好
开发速度	慢	一般	快
性能	快	较慢	慢
学习成本	高	一般	低

React Native 的优势（为什么用 RN）

- 开发体验好
 - 用统一的代码规范开发移动端程序，不用关注移动端的差异
- 开发成本低
 - 开发一次，可以生成 Android 和 iOS 两个系统上的 App
 - Learn once, write anywhere
- 学习成本低
 - 只要掌握 JavaScript 和 React，就可以进行移动端开发了

React Native

React Native
(JavaScript)

Bridg
e

Android
(Kotlin | Java)

iOS
(Swift |
Objective-C)

React Native 的不足

- 不成熟；
 - 项目版本更新维护较频繁，学习成本高；
 - 试错成本高，有些问题较少解决方案，易耽误开发进度。
- 性能差
 - 整体性能仍不如原生；
- 兼容性差
 - 涉及底层的功能，需要针对 Android 和 iOS 双端单独开发；

搭建开发环境

搭建开发环境

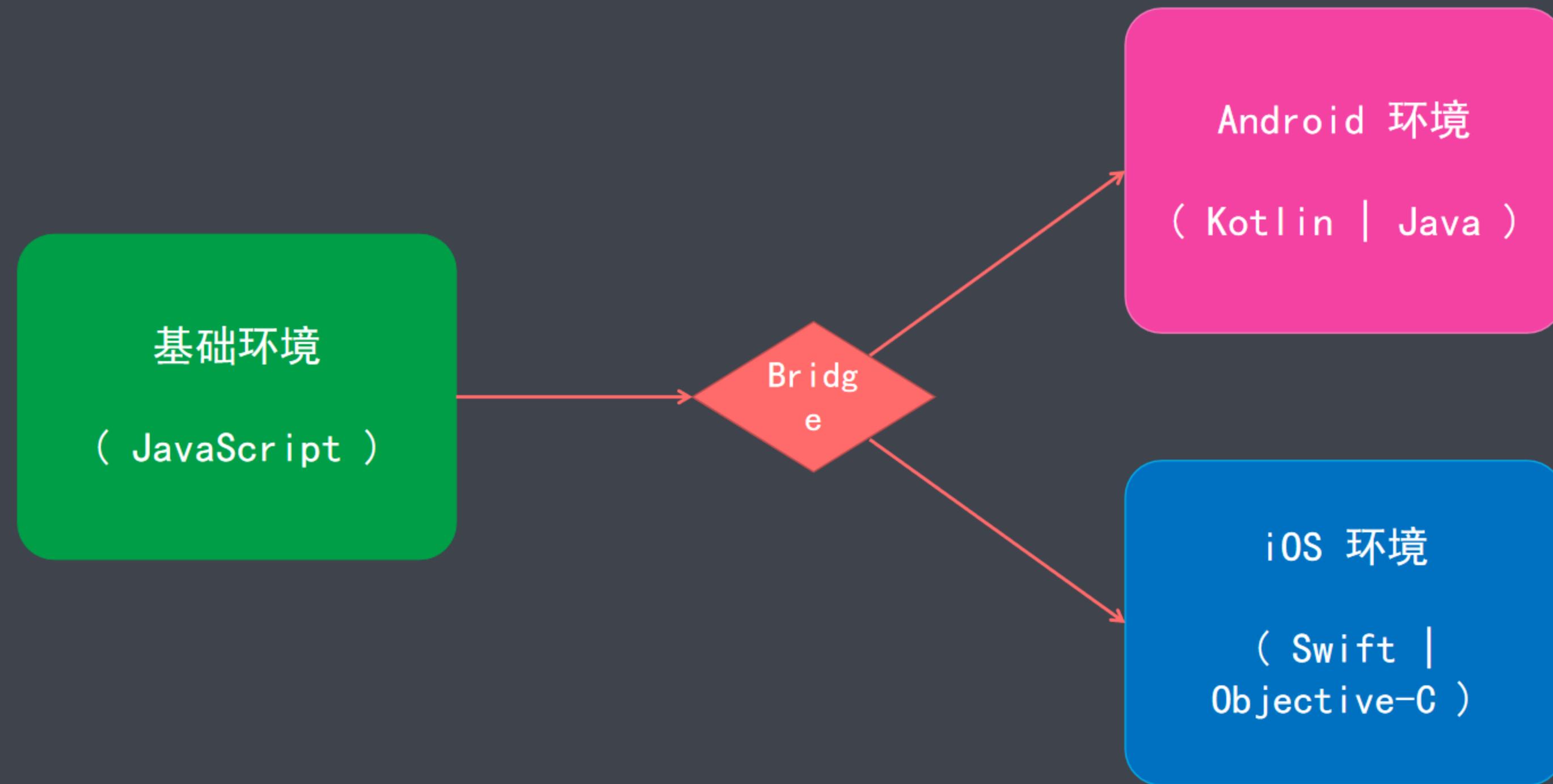
STEPS

1. 基础环境
2. 搭建安卓环境
3. 搭建 iOS 环境
4. 初始化项目

基础环境搭建

- 安装 Node.js
 - Node.js 的版本应 ≥ 12 (推荐安装 LTS 版本)
 - `npm config set registry https://registry.npm.taobao.org`
- 安装 Yarn
 - `npm install -g yarn`
- 安装 React Native 脚手架
 - `npm install -g react-native-cli`

React Native 运行环境



Windows 下只能搭建 Android 开发环境

Mac 下既能搭建 Android 开发环境，也能搭建 iOS 开发环境

搭建 React Native 开发环境



搭建开发环境

搭建安卓环境

搭建安卓环境

STEPS

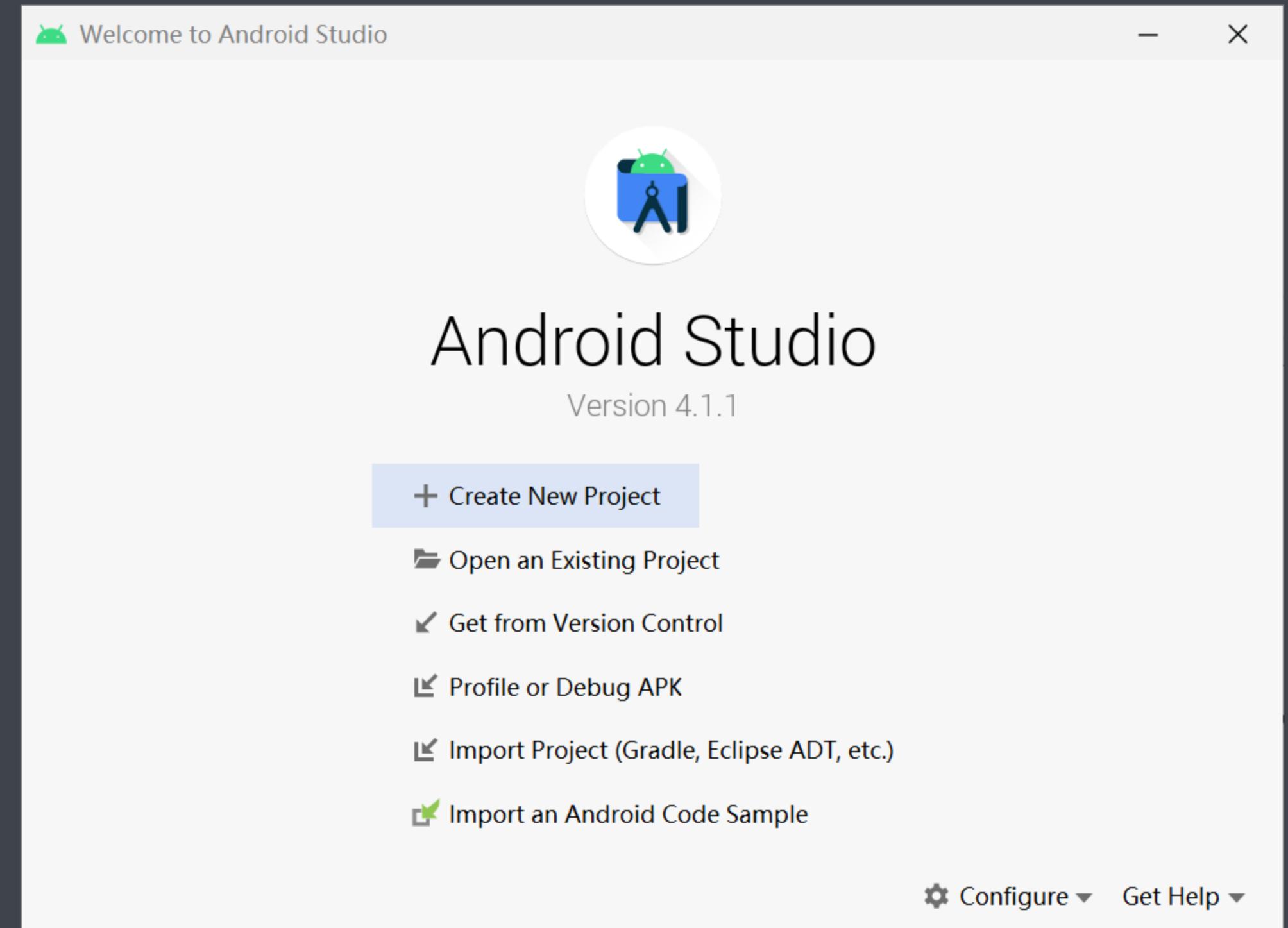
1. 安装 JDK
2. 安装 Android Studio
3. 安装 Android SDK
4. 配置环境变量

安装 JDK

- 下载 JDK (*Java SE Development Kit*)
 - <https://www.oracle.com/java/technologies/javase/javase-jdk8-downloads.html>
- JDK 的版本必须是 1.8 (1.8 版本，官方也直接称 8 版本)
 - 目前不支持高于 1.8 的 JDK 版本
- 下载时要求登陆 (请先注册 Oracle 账号)
 - 或者直接找老师，获取上面的安装包
- 安装 JDK (一直 “下一步”)
 - 命令行中，输入 `java -version`，验证安装是否成功

安装 Android Studio

- 下载 Android Studio
 - <https://developer.android.com/studio/index.html>
- 安装 Android Studio (一直“下一步”)
- 启动 Android Studio
 - 初次启动，需要安装组件（组件约 2 GB，安装后占用空间约 8 GB）
 - 安装组件的过程**巨长巨长巨长**，要有耐心
- 创建项目



Project

app
manifests
AndroidManifest.xml

Resource Manager

java
java (generated)
res
res (generated)

Gradle Scripts

I: Structure

Favorites

2: Favorites

Build Variants

2: Favorites

activity_main.xml MainActivity.java AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.rntutorial">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="rntutorial"
9          android:roundIcon="@mipmap/ic_launcher_round"
10         android:supportsRtl="true"
11         android:theme="@style/Theme.Rntutorial">
12             <activity android:name=".MainActivity">
13                 <intent-filter>
14                     <action android:name="android.intent.action.MAIN" />
15
16                     <category android:name="android.intent.category.LAUNCHER" />
17                 </intent-filter>
18             </activity>
19         </application>
20
21     </manifest>
```

manifest > application

Text

Merged Manifest

安装 Android SDK

- What
 - Android SDK 是针对安卓开发的套件
- Why
 - 虽然 Android Studio 默认会安装最新版本的 Android SDK
 - 但是，目前编译 React Native 应用需要的是 Android 10 (Q) 版本的 SDK
- How
 - 打开 Android Studio，在菜单 Tools 下找到 "SDK Manager"

Settings for New Projects

Appearance & Behavior > System Settings > Android SDK

Manager for the Android SDK and Tools used by Android Studio

Android SDK Location: C:\Users\changtaoliu\AppData\Local\Android\Sdk [Edit](#) [Optimize disk space](#)

SDK Platforms [Edit](#) [Optimize disk space](#) [SDK Tools](#) [SDK Update Sites](#)

Each Android SDK Platform package includes the Android platform and sources pertaining to an API level by default. Once installed, Android Studio will automatically check for updates. Check "show package details" to display individual SDK components.

Name	API Level	Revision	Status
Android 11.0 (R)			
Android SDK Platform 30	30	3	Installed
Sources for Android 30	30	1	Installed
Google Play Intel x86 Atom System Image	30	9	Installed
Android R Preview			
Google APIs Intel x86 Atom System Image	R	2	Not installed
Google APIs Intel x86 Atom_64 System Image	R	2	Not installed
Google Play Intel x86 Atom System Image	R	2	Not installed
Google Play Intel x86 Atom_64 System Image	R	2	Not installed
Android 10.0 (Q)			
Android SDK Platform 29	29	5	Not installed
Sources for Android 29	29	1	Not installed
Intel x86 Atom System Image	29	7	Not installed
Intel x86 Atom_64 System Image	29	7	Not installed
Google APIs Intel x86 Atom System Image	29	9	Not installed
Google APIs Intel x86 Atom_64 System Image	29	9	Not installed
Google Play Intel x86 Atom System Image	29	8	Not installed
Google Play Intel x86 Atom_64 System Image	29	8	Not installed

安装 React Native 必须的 Android SDK Platform 29

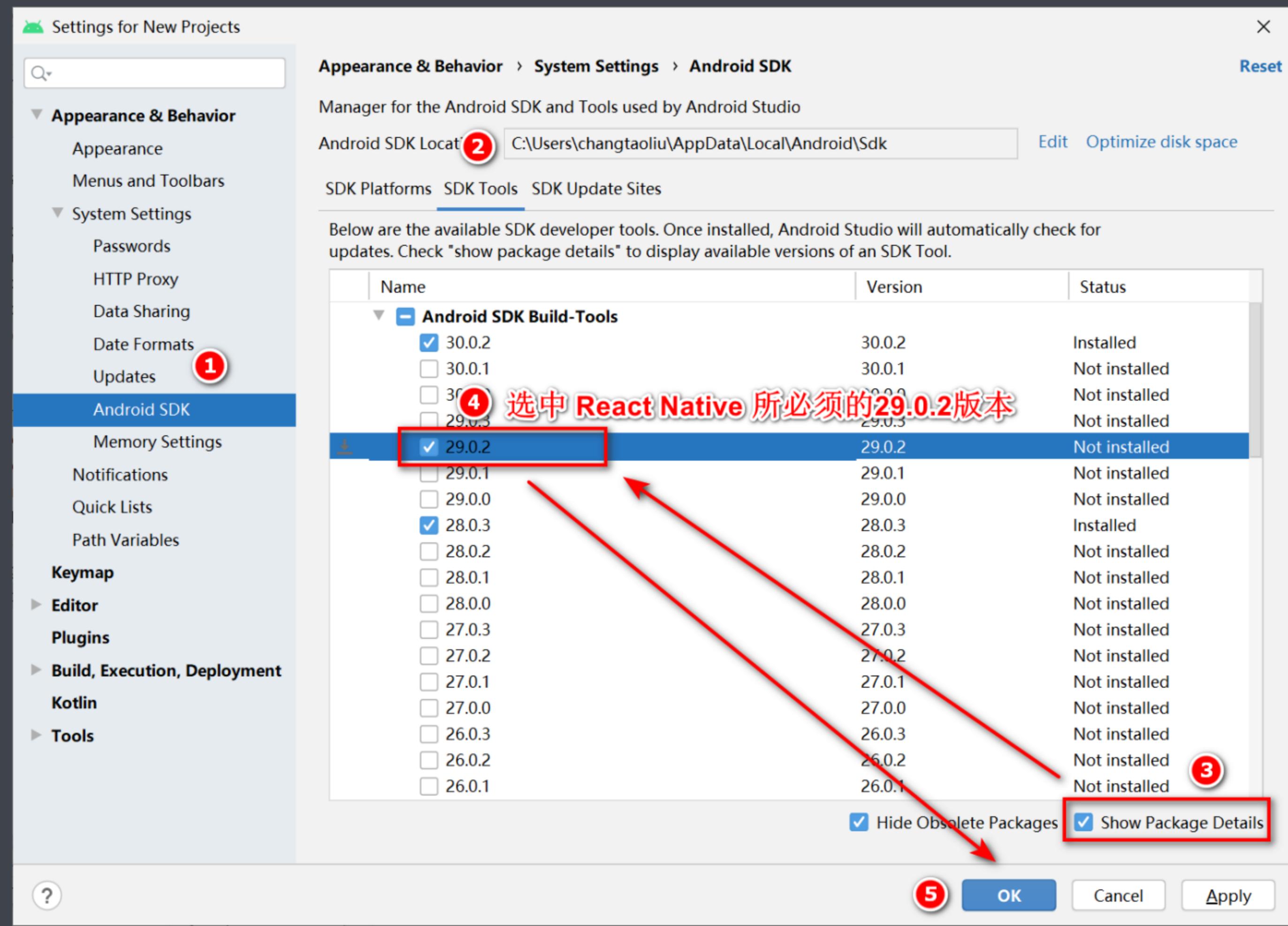
① Updates [Edit](#) [Optimize disk space](#) [SDK Tools](#) [SDK Update Sites](#)

② Android SDK Location: C:\Users\changtaoliu\AppData\Local\Android\Sdk [Edit](#) [Optimize disk space](#)

③ Not installed

④ 安装 React Native 必须的 Android SDK Platform 29

⑤ Hide Obsolete Packages Show Package Details [OK](#) [Cancel](#) [Apply](#)

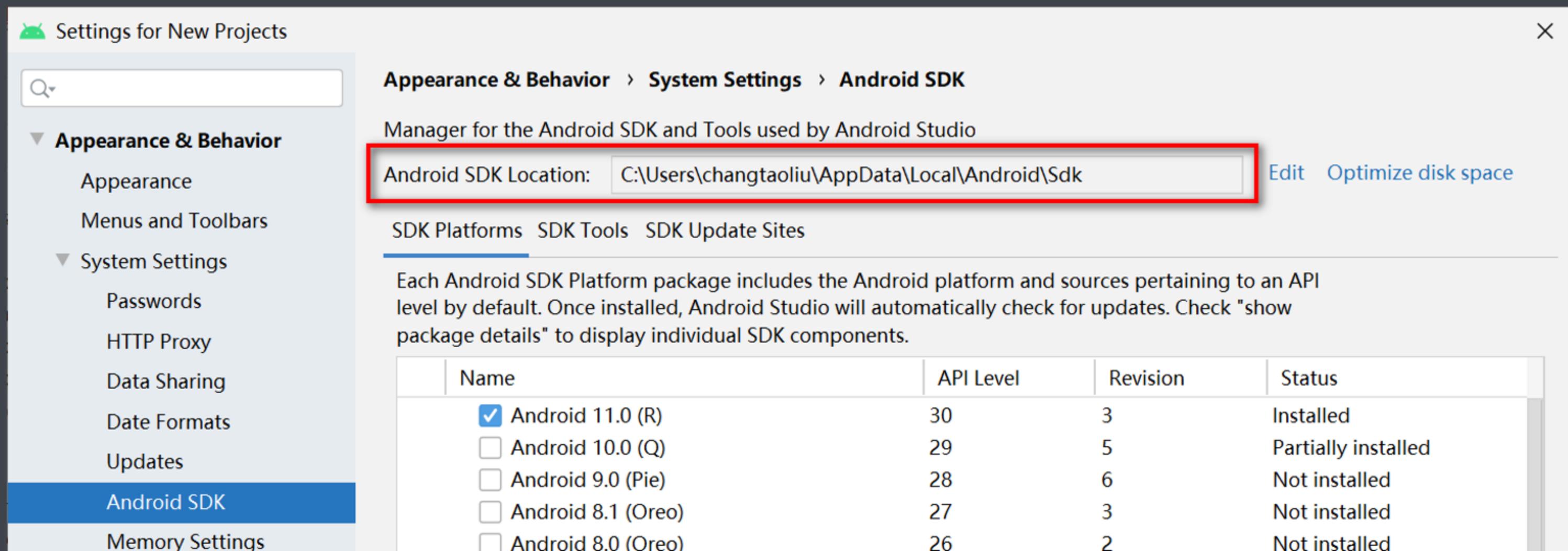


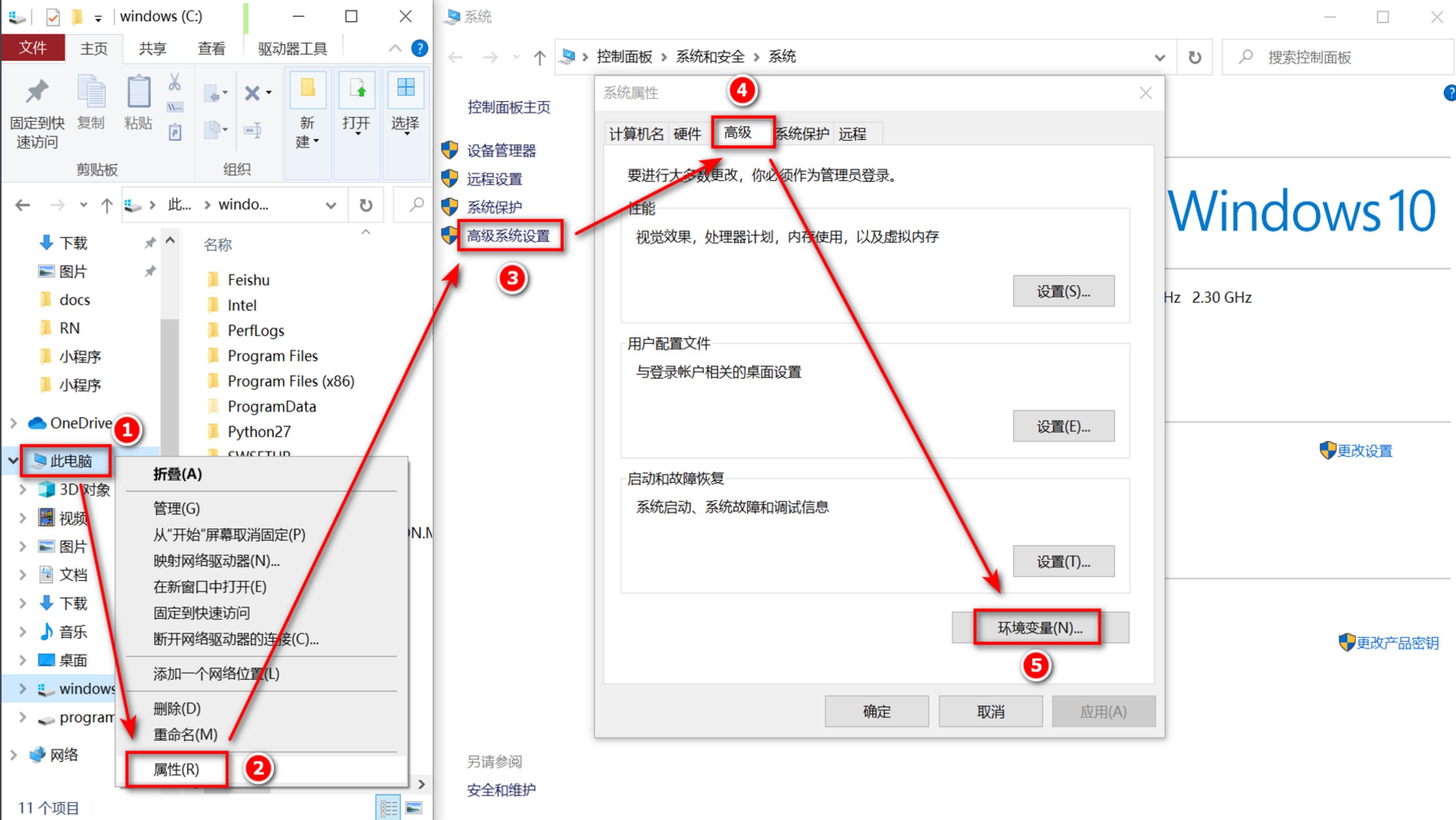
配置环境变量

- 配置 ANDROID_HOME 环境变量

- 打开 Android Studio, 点击菜单 Tools → SDK Manager, 找到

- Appearance & Behavior → System Settings → Android SDK





环境变量

changtaoliu 的用户变量(U)

新建系统变量

7 填写本地

变量	值
OneDrive	C:\Users\ch
Path	C:\Users\ch
TEMP	C:\Users\ch
TMP	C:\Users\ch

变量名(N): ANDROID_HOME

变量值(V): C:\Users\changtaoliu\AppData\Loca

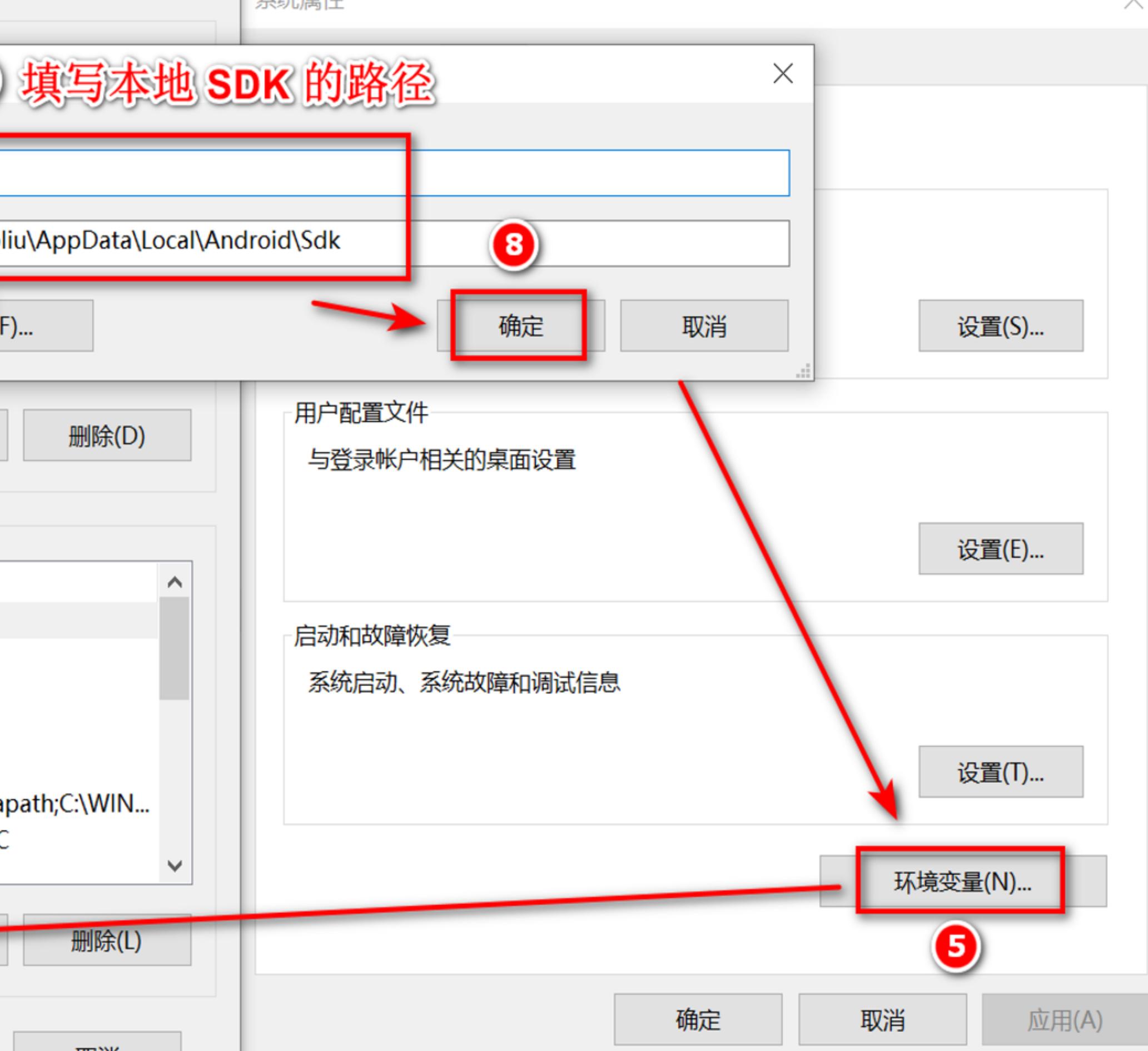
浏览目录(D)... 浏览文件(F)...

新建(N)... 编辑(E)... 删除(D)

系统变量(S)

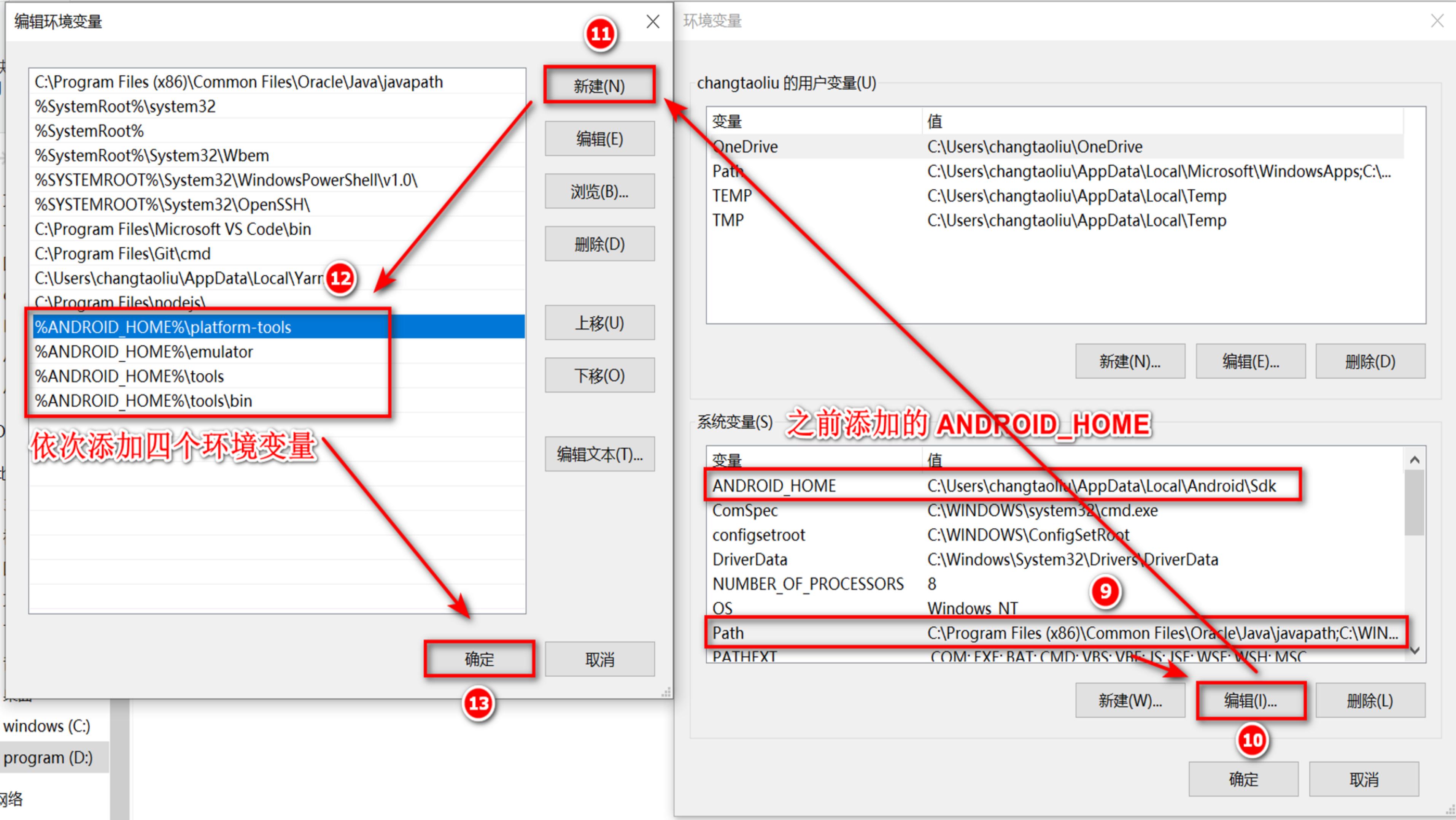
变量	值
ComSpec	C:\WINDOWS\system32\cmd.exe
configsetroot	C:\WINDOWS\ConfigSetRoot
DriverData	C:\Windows\System32\Drivers\DriverData
NUMBER_OF_PROCESSORS	8
OS	Windows_NT
Path	C:\Program Files (x86)\Common Files\Oracle\Java\javapath;C:\WIN...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC
PROCESSOR_ARCHITECTURE	AMD64

The screenshot shows a horizontal toolbar with three buttons: '新建(W)...' (New), '编辑(I)...' (Edit), and '删除(L)' (Delete). A red box highlights the '新建(W)...' button. A red arrow points from the '新建(W)...' button towards the '编辑(I)...' button. A red circle with the number '6' is positioned below the toolbar.



配置环境变量

- 配置 ANDROID_HOME 环境变量
 - 打开 Android Studio，点击菜单 Tools → SDK Manager，找到 Appearance & Behavior → System Settings → Android SDK
- 跟 ANDROID_HOME 相关的环境变量
 - %ANDROID_HOME%\platform-tools
 - %ANDROID_HOME%\emulator
 - %ANDROID_HOME%\tools
 - %ANDROID_HOME%\tools\bin



RN 安卓环境搭建.md

搭建开发环境

搭建 iOS 环境

搭建 iOS 环境

STEPS

1. 安装 Watchman
2. 安装 Xcode
3. 安装 CocoaPods

iOS 环境搭建

- Watchman
 - brew install watchman
- Xcode
 - React Native 目前需要 Xcode 10 或更高版本。你可以通过 App Store 下载
- CocoaPods
 - brew install cocoapods

RN iOS 环境搭建.md

搭建开发环境

初始化项目

初始化项目

STEPS

1. 创建项目
2. 安装 VS Code 插件
3. 调试工具

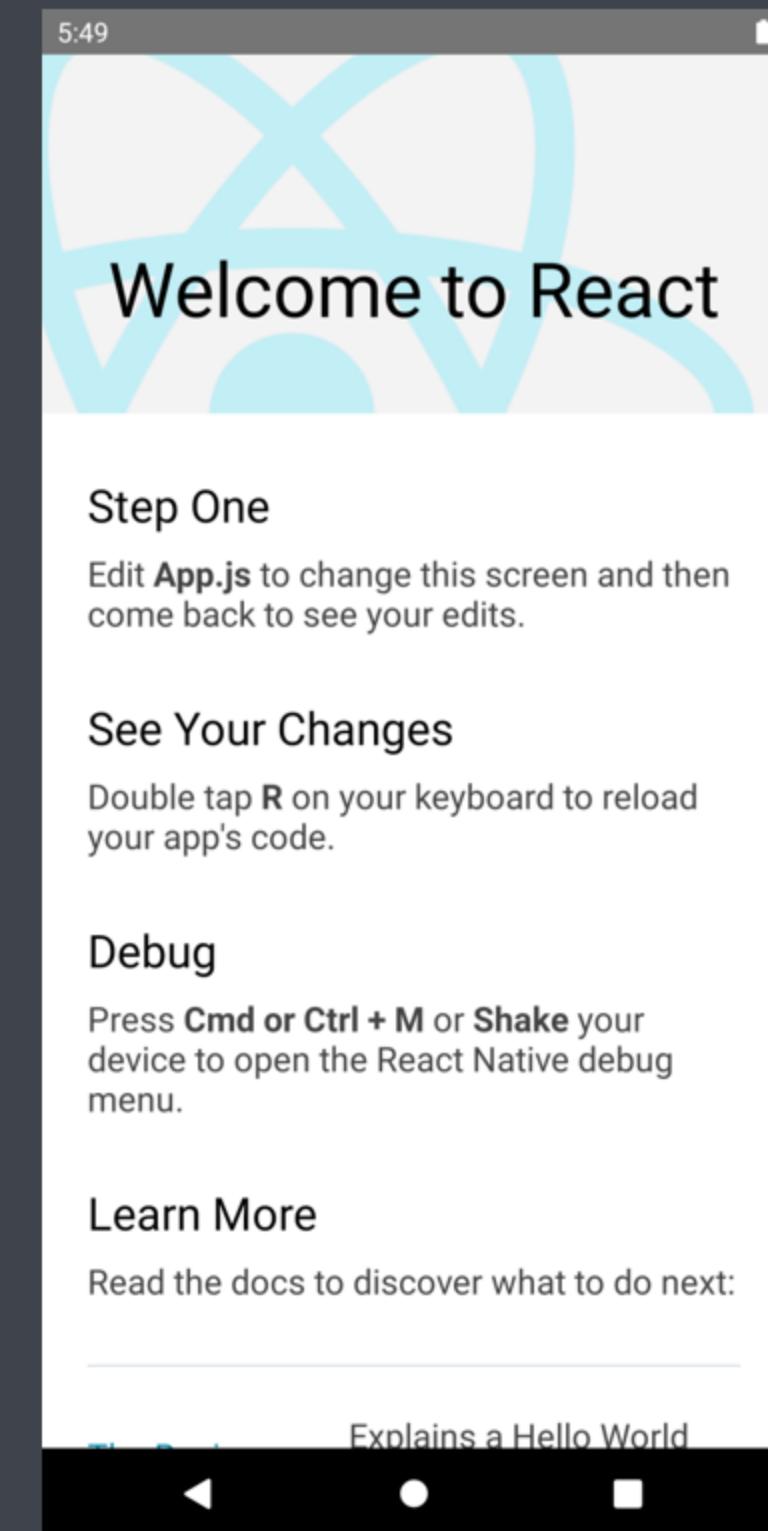
创建项目

- 初始化项目
 - `react-native init myproject`
- 进入项目
 - `cd myproject`
- 运行项目
 - Android
 - `yarn android`
 - iOS
 - `cd ios && pod install && cd ..`
 - `yarn ios`

创建项目

名称	类型	大小
__tests__	文件夹	
android	文件夹	
ios	文件夹	
node_modules	文件夹	
.buckconfig	BUCKCONFIG 文件	1 KB
.eslintrc.js	JavaScript 文件	1 KB
.flowconfig	FLOWCONFIG 文件	2 KB
.gitattributes	文本文档	1 KB
.gitignore	文本文档	1 KB
.prettierrc.js	JavaScript 文件	1 KB
.watchmanconfig	WATCHMANCONFI...	1 KB
App.js	JavaScript 文件	3 KB
app.json	JSON 文件	1 KB
babel.config.js	JavaScript 文件	1 KB
index.js	JavaScript 文件	1 KB
metro.config.js	JavaScript 文件	1 KB
package.json	JSON 文件	1 KB
yarn.lock	LOCK 文件	310 KB

yarn android
或
yarn ios



安装 VS Code 插件

The screenshot shows the Visual Studio Code interface. On the left, the Extensions view is open, displaying a list of installed extensions. One extension, 'ES7 React/Redux/GraphQL/React-Native snippets' by dsznajder, is highlighted with a red oval and a red arrow pointing to it from the bottom-left. The main panel shows a detailed view of this extension. The title is 'Extension: ES7 React/Redux/GraphQL/React-Native snippets - rntutorial - Visual Studio Code'. Below the title is another tab labeled 'Extension: ES7 React/Redux/GraphQL/React-Native snippets X'. The extension details include:

- ES7 React/Redux/GraphQL/React-Native snippets**
- dsznajder | 2,036,816 | ★★★★☆ | Repository | v3.0.0
- Simple extensions for React, Redux and GraphQL in JS/TS with ES7 syntax
- Disable | Uninstall | This extension is enabled globally.

Below the extension details, there is a section titled 'VS Code ES7 React/Redux/React-Native/JS snippets' with the following information:

- Visual Studio Marketplace | v3.0.0 | installs 2038918 | downloads 9822230 | rating 4.19/5 (27)
- This extension provides you JavaScript and React/Redux snippets in ES7 with Babel plugin features for VS Code
- Installation**
 - Visual Studio Marketplace
- Launch Quick Open:

快捷命令: rnc (react native class)



```
import React, { Component } from 'react'
import { Text, View } from 'react-native'

export default class FileName extends Component {
  render() {
    return (
      <View>
        <Text> $2 </Text>
      </View>
    )
  }
}
```

快捷命令: rnf (react native function)



```
import React from 'react'
import { View, Text } from 'react-native'

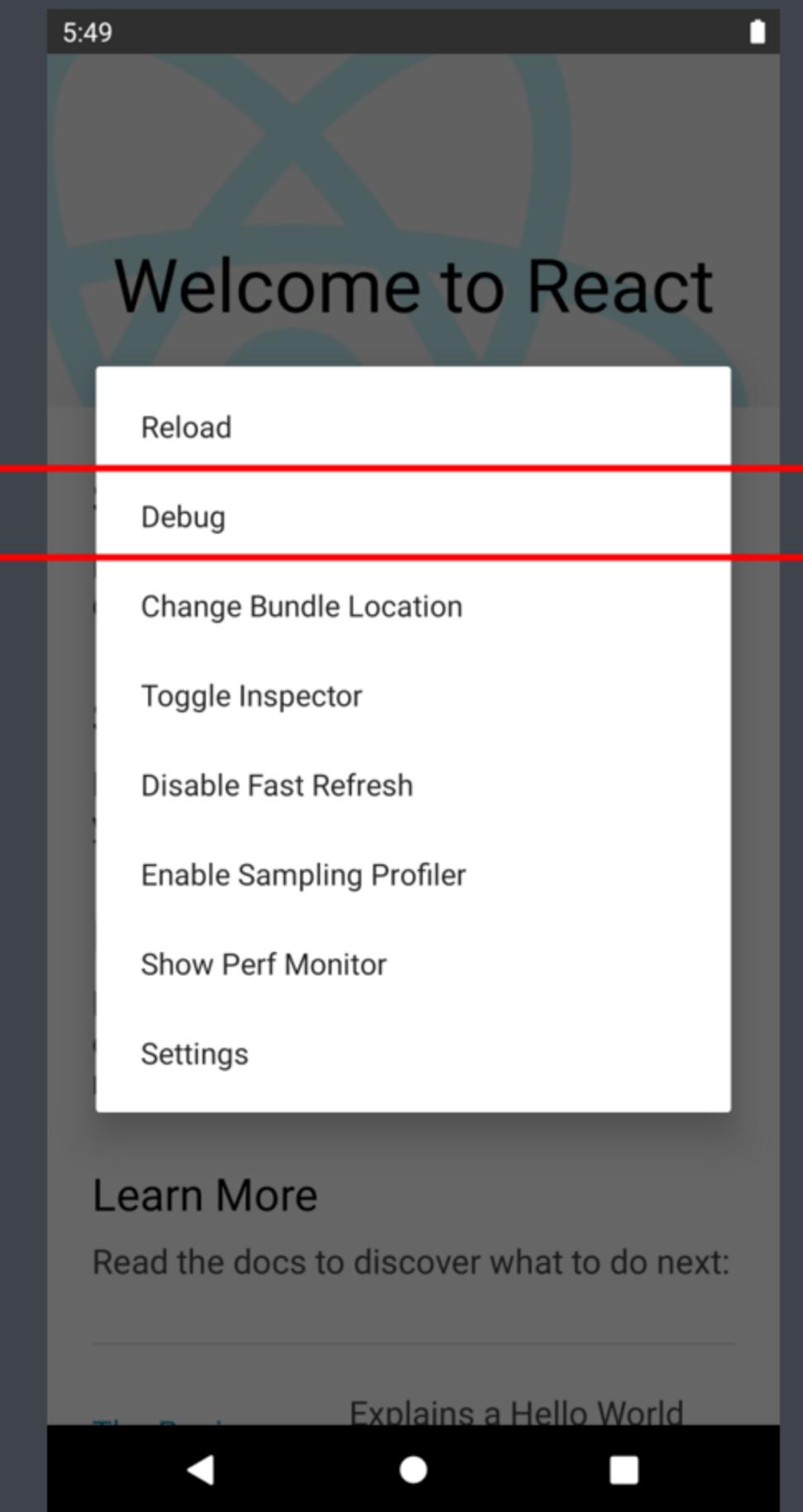
export default function $1() {
  return (
    <View>
      <Text> $2 </Text>
    </View>
  )
}
```

调试工具

- 模拟器调试
 - 模拟器是安装在电脑上的，虚拟的手机界面
 - 模拟器一般跟随 Android Studio 和 Xcode 一起安装
 - 启动应用，模拟器会一起启动
- 真机调试
 - 打开 USB 调试模式
 - 通过 USB 线将电脑和手机连起来
 - 启动应用，在手机上安装应用

模拟器调试

- 点击模拟器（使模拟器获取焦点）
- 快捷键 `ctrl + m`
- 点选 `debug`
- 自动跳转到浏览器



环境和软件版本

- 操作系统: Windows 10
- 开发工具: VS Code
- React Native 版本: 0.63.3
- React 版本: 16.13.1

基础语法

基础语法

STEPS

1. 掌握 React
2. StyleSheet
3. Flexbox
4. 组件和 API
5. 路由与导航

React 语法

- JSX 语法
- 组件（分类，传参，属性，状态）
- 生命周期
- Hook API
- Redux
- 常用安装包

基础语法

StyleSheet

StyleSheet 是 RN 中声明样式的 API

RN 中的样式与 CSS 的不同

- 没有继承性
 - RN 中的继承只发生在 `Text` 组件上
- 样式名采用小驼峰命名
 - `fontSize` VS `font-size`
- 所有尺寸都是没有单位
 - `width: 100`
- 有些特殊的样式名
 - `marginHorizontal` (水平外边距) , `marginVertical` (垂直外边距)

RN 样式的声明方式

- 通过 style 属性直接声明
 - 属性值为对象: <组件 style={ {样式} } />
 - 属性值为数组: <组件 style={ [{样式1}, ..., {样式N}] } />
- 在 style 属性中调用 StyleSheet 声明的样式
 - 引入: import { StyleSheet, View } from 'react-native'
 - 声明: const styles = StyleSheet.create({ foo: {样式1}, bar: {样式2} })
 - 使用: <View style={ [, styles.foo, styles.bar] } >内容</View>

```
● ● ●  
import React, { Component } from 'react'  
import { Text, StyleSheet, View } from 'react-native'  
  
export default class App extends Component {  
  render() {  
    return (  
      <View style={ {  
        marginHorizontal: 20,  
        backgroundColor: '#dfb'  
      } }>  
        <Text style={[styles.red, styles.fontLarge]}>Hello RN</Text>  
      </View>  
    )  
  }  
  
  const styles = StyleSheet.create({  
    red: {  
      color: '#e33'  
    },  
    fontLarge: {  
      fontSize: 40,  
    }  
  })  
}
```

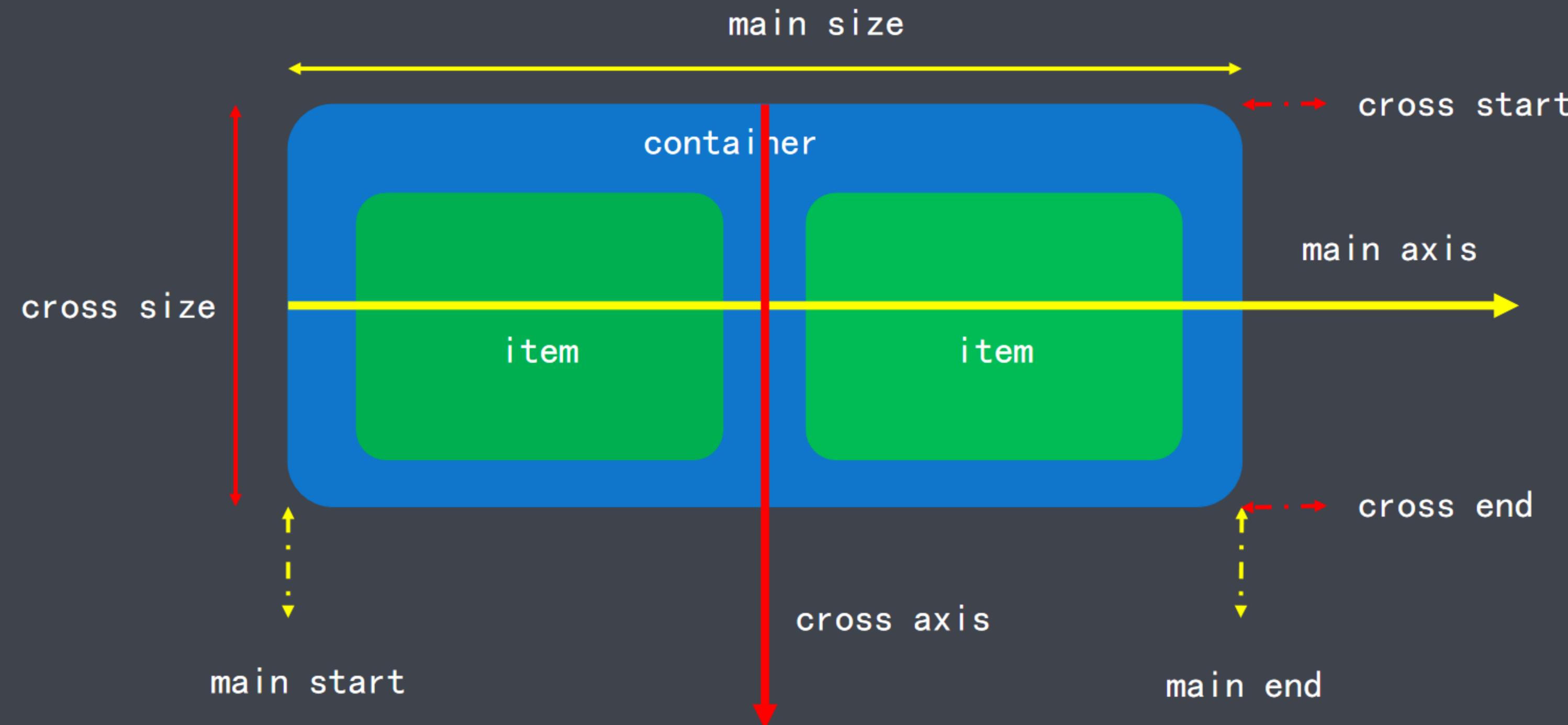
基础语法

Flexbox

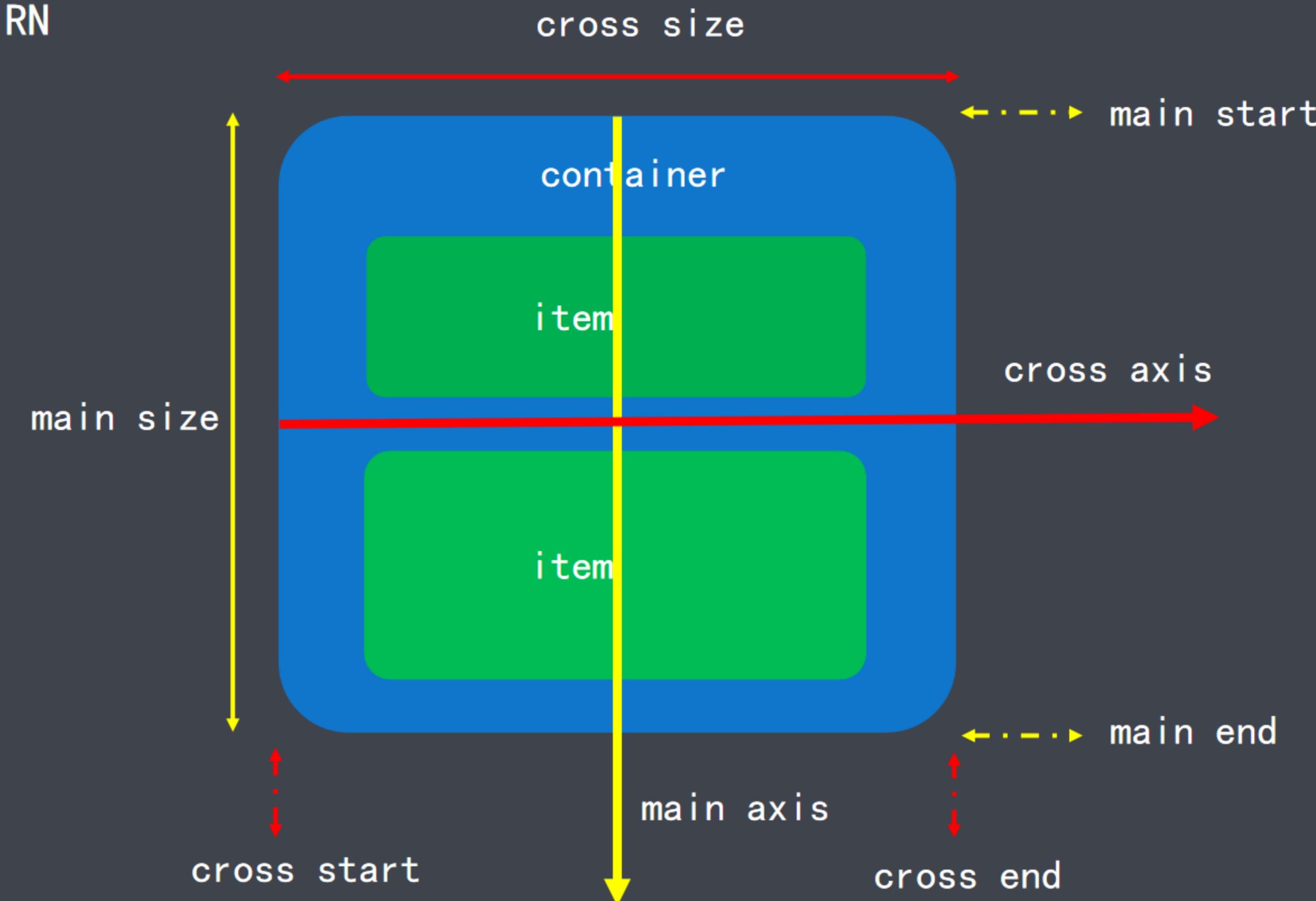
Flexbox - 术语

- 容器 (container)
 - 采用 Flex 布局的元素，称为 Flex 容器 (flex container)，简称"容器"
- 项目 (item)
 - 容器所有子元素，称为 Flex 项目 (flex item)，简称"项目"
- 主轴 (main axis)
- 交叉轴 (cross axis)

Flexbox – Web



Flexbox – RN



Flexbox - 属性

- `flexDirection`
 - 声明主轴方向: `row` (Web默认) | `column` (RN默认)
- `justifyContent`
 - 声明项目在主轴上的对齐方式
- `alignItems`
 - 声明项目在交叉轴上的对齐方式
- `flex`
 - 声明项目在主轴上的尺寸比例

响应式布局

- Flexbox
- Dimensions
 - import { Dimensions } from 'react-native';
 - const windowHeight = Dimensions.get('window').width;
 - const windowHeight = Dimensions.get('window').height;

基础语法

组件 和 API

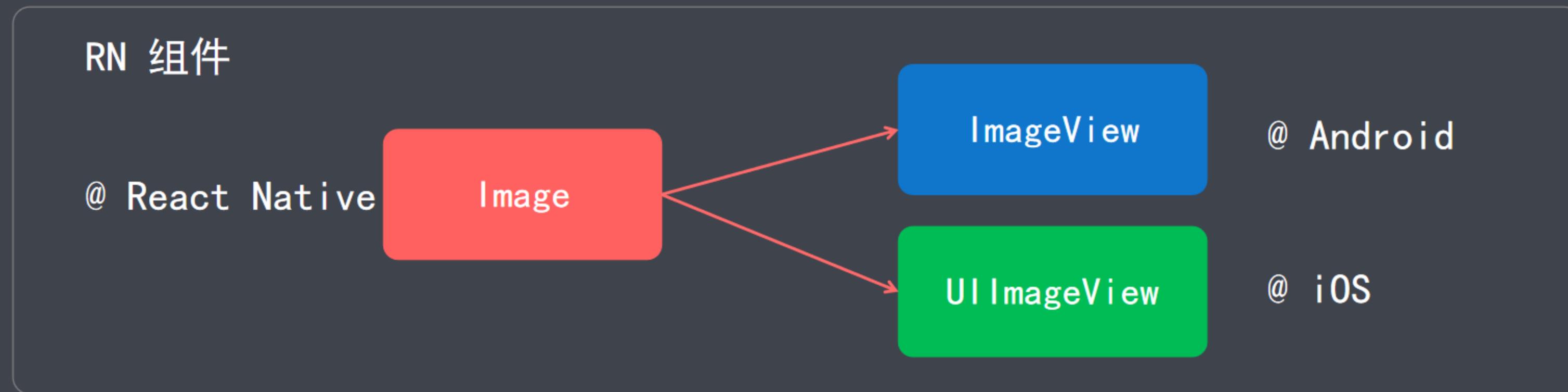
组件 和 API

STEPS

1. 简介
2. 核心组件
3. 第三方组件
4. 自定义组件

简介

- RN 中的核心组件，是对原生组件的封装
 - 原生组件：Android 或 iOS 内的组件
 - 核心组件：RN 中最常用的，来在 react-native 的组件

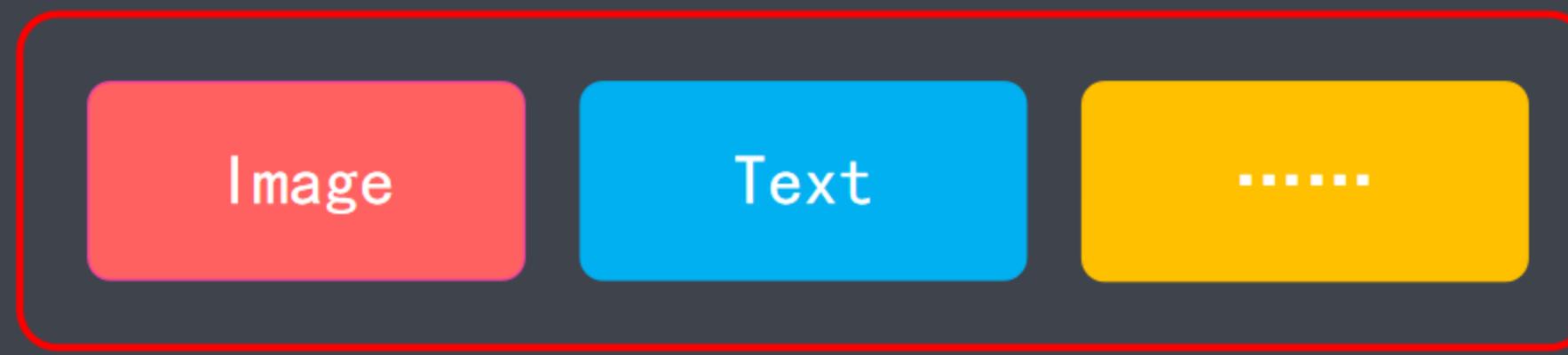


RN 组件的封装原理

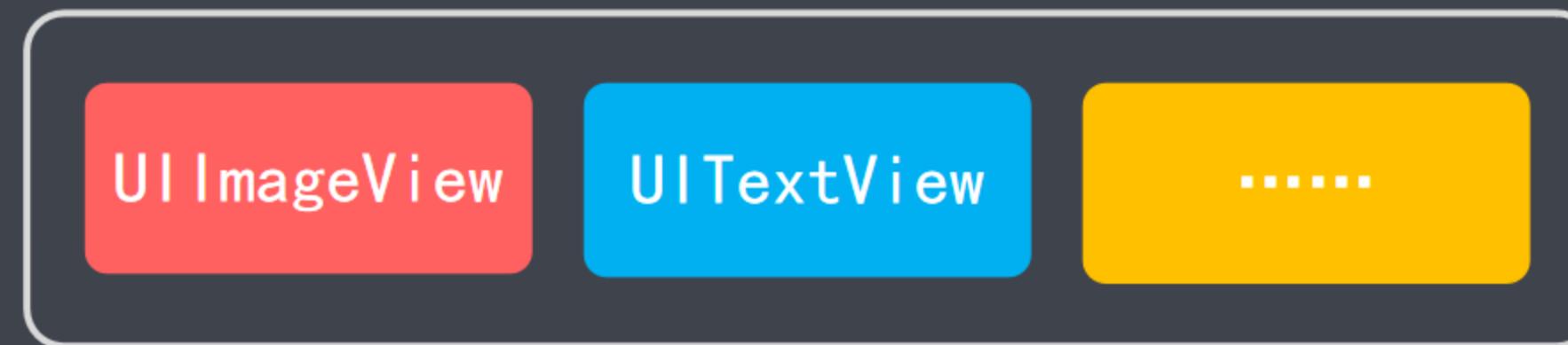
@ Android



@ React Native



@ iOS



RN 组件

作用	RN 组件	安卓视图	iOS 视图	HTML 标签
展示区块	View	ViewGroup	UIView	div
展示图片	Image	ImageView	UIImageView	img
展示文本	Text	TextView	UITextView	p
.....

核心组件

原生组件

核心组件

- 基础组件
- 交互控件
- 列表视图
- iOS 独有组件
- Android 独有组件
- 其他

```
import {  
    View,  
    Text,  
    TouchableOpacity,  
    Dimensions,  
    StyleSheet,  
    StatusBar,  
    Image,  
    ImageBackground  
} from 'react-native';
```

核心组件

View

视图组件

Text

文本组件

Alert

警告框组件

Button

按钮组件

Switch

开关组件

StatusBar

状态栏组件

ActivityIndicator

加载指示器组件

核心组件

Image

图片组件

TextInput

输入框组件

Touchable

触碰组件（共三个）

ScrollView

滚动视图组件

SectionList

分组列表组件

FlatList

高性能列表组件

Animated

动画组件

核心组件

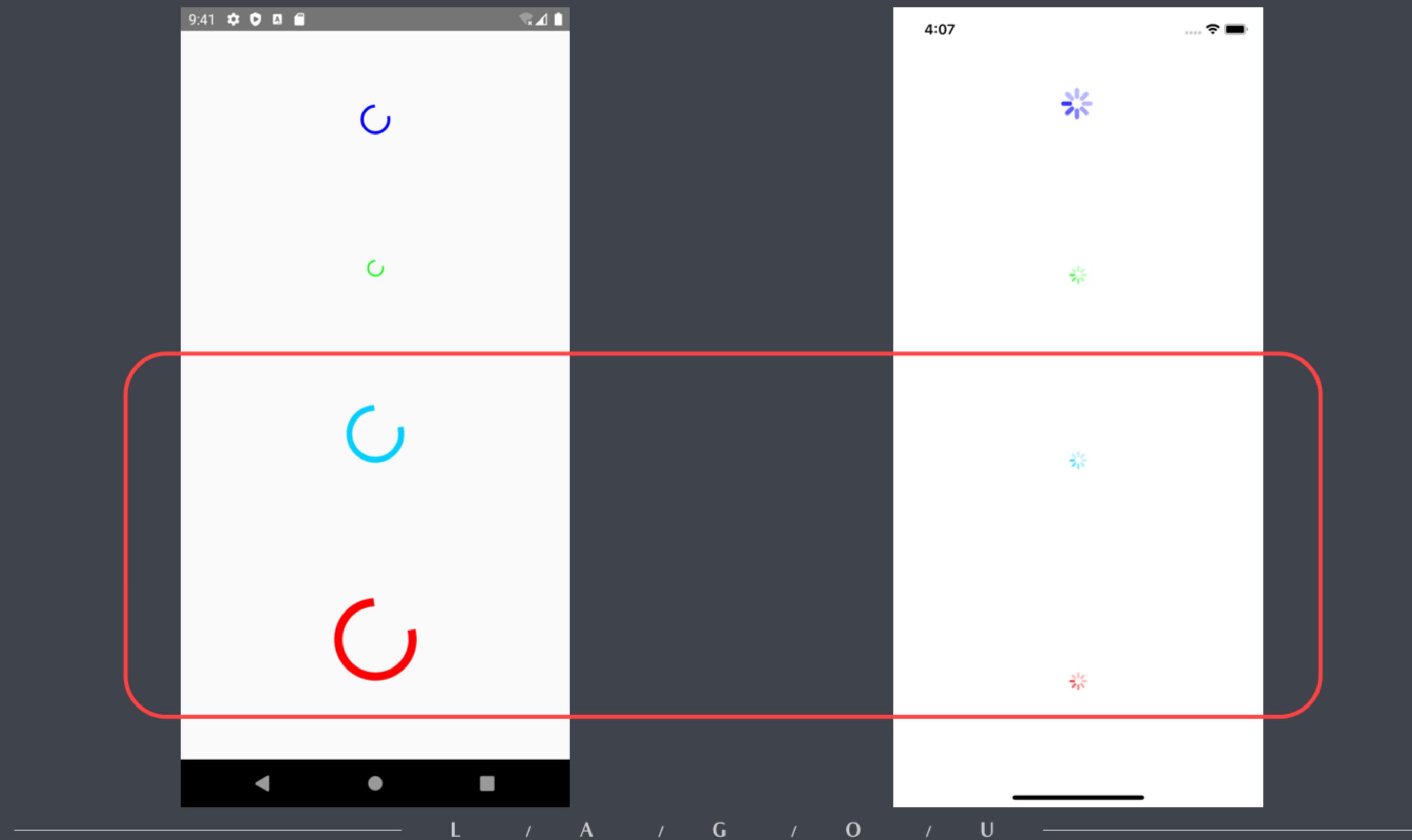
Alert & Button

核心组件

Switch && StatusBar

核心组件

ActivityIndicator



核心组件

Image

Image 组件

- 作用
 - 加载图片
- 加载方式
 - 本地路径
 - 图片的 URI 地址
 - 图片的 Base64 字符串

核心组件

TextInput

核心组件

Touchable 组件

Touchable 组件

- TouchableHighlight
 - 触碰后，高亮显示
- TouchableOpacity
 - 触碰后，透明度降低（模糊显示）
- TouchableWithoutFeedback
 - 触碰后，无任何响应

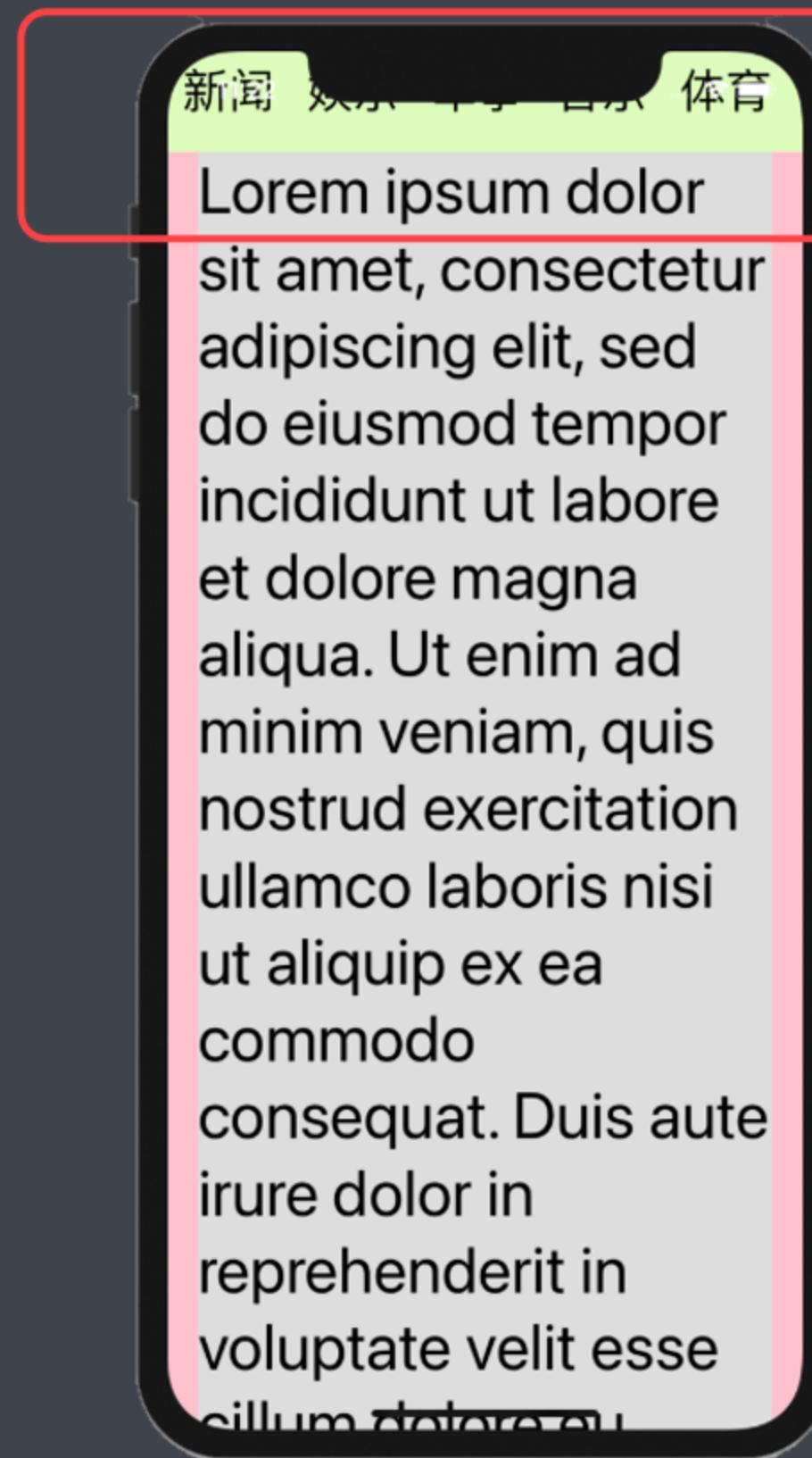
核心组件

ScrollView

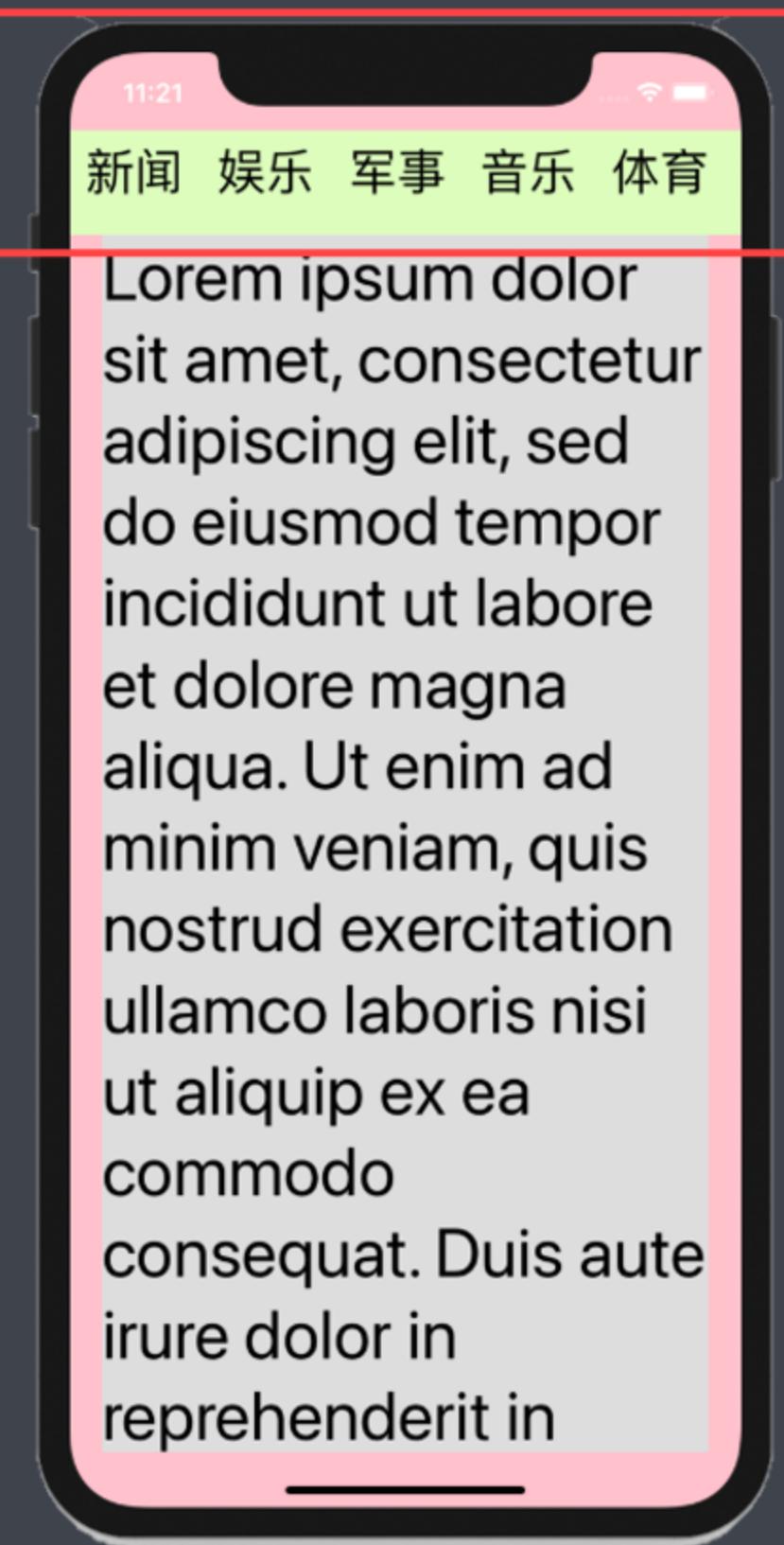
核心组件

SafeAreaView

View



SafeAreaView



核心组件

SectionList

核心组件

FlatList

核心组件

Animated

组件必须经过特殊处理才能用于动画

RN 中可以直接使用的动画组件

- Animated.View
- Animated.Text
- Animated.ScrollView
- Animated.Image

如何创建动画（步骤）

- 创建初始值
 - `Animated.Value()` 单个值
 - `Animated.ValueXY()` 向量值
- 将初始值绑定的动画组件上
 - 一般将其绑定到某个样式属性下，例如：`opacity`、`translate`
- 通过动画类型API，一帧一帧地更改初始值
 - `Animated.decay()` 加速效果
 - `Animated.spring()` 弹跳效果
 - `Animated.timing()` 时间渐变效果

第三方组件

第三方组件

- 需要单独安装的组件
- 使用步骤
 - 安装
 - 配置
 - 使用

第三方组件

WebView

相当于内置浏览器

Picker

下拉框

Swiper

展示轮播效果

AsyncStorage

持久化存储系统

Geolocation

获取定位信息

Camera

调用摄像头

第三方组件

WebView

WebView

- 安装
 - `yarn add react-native-webview`
- 配置
 - <https://github.com/react-native-webview/react-native-webview>
- 使用
 - 直接指定 `uri` 地址
 - 直接渲染 `html` 代码

第三方组件

Picker

Picker (下拉框)

- 安装
 - `yarn add @react-native-picker/picker`
- 配置
 - <https://github.com/react-native-picker/picker>
 - 注意：不同版本的配置方式不同
- 使用
 - 注意平台之间的差异 (Android | iOS)

第三方组件

Swiper

Swiper (轮播效果)

- 安装
 - `yarn add react-native-swiper`
- 使用
 - <https://github.com/leecade/react-native-swiper>

第三方组件

AsyncStorage

AsyncStorage

- 安装
 - `yarn add @react-native-async-storage/async-storage`
- 配置
 - <https://github.com/react-native-async-storage/async-storage>
- 使用
 - 增删改查

第三方组件

AsyncStorage

封装 AsyncStorage

- 增
 - `set(key, value)`
- 删
 - `delete(key) | clear()`
- 改
 - `update(key, value)`
- 查
 - `get(key)`

第三方组件

Geolocation

Geolocation

- 安装
 - `yarn add @react-native-community/geolocation`
- 配置
 - <https://github.com/react-native-geolocation/react-native-geolocation>
 - 添加获取定位信息的授权许可
- 使用
 - 通过手机，获取经纬度信息

第三方组件

Camera

Camera (摄像头)

- 安装
 - `npm install react-native-camera --save`
- 配置
 - <https://github.com/react-native-camera/react-native-camera>
 - 添加使用摄像头的授权许可
- 使用
 - 拍照、扫码、人脸识别……

第三方组件

react-native-image-picker

React-native-image-picker

- 安装
 - `yarn add react-native-image-picker`
- 配置
 - <https://github.com/react-native-image-picker/react-native-image-picker>
 - 与 Camera 一直
- 使用
 - 调用摄像头
 - 访问相册

自定义组件

自定义组件

- 工程师自己写的组件
 - React 是面向组件开发的（所有内容都是组件）
 - 这里的自定义组件是指：具有特定功能的，可以重复使用的公共组件

基础语法

路由与导航

路由与导航

STEPS

1. 简介
2. 基础组件
3. Stack 导航
4. BottomTab 导航
5. Drawer 导航
6. MaterialTopTab 导航

简介

- RN 中的路由是通过 React-Navigation 来完成的
 - React 中通过 React-Router 实现路由
 - RN 0.44 之前，React-Navigation 在核心中维护，0.44 之后，独立维护
- 本节使用 React-Navigation 5.x 版本
 - 官网：<https://reactnavigation.org/>

基础组件

- 安装
 - *yarn add @react-navigation/native*
 - *yarn add react-native-reanimated react-native-gesture-handler react-native-screens react-native-safe-area-context @react-native-community/masked-view*
- 链接
 - RN 0.60 后安卓环境自动链接路由（Android 无需任何操作）
 - iOS 下需要手动链接路由（*npx pod-install ios*）

基础组件

- 添加头部组件
 - 将如下代码，放到应用的头部（例如：放到 index.js 或 App.js 文件的头部）

```
import 'react-native-gesture-handler';
```
- 添加导航容器
 - 我们需要在入口文件中，把整个应用，包裹在导航容器（NavigationContainer）中（例如：在 index.js 或 App.js 文件中）

添加导航



```
import 'react-native-gesture-handler';
import * as React from 'react';
import { NavigationContainer } from '@react-navigation/native';

export default function App() {
  return (
    <NavigationContainer>
      {/* 具体的应用代码 */}
    </NavigationContainer>
  );
}
```

路由与导航

Stack 导航

Stack 导航

- 简介
 - RN 中默认没有类似浏览器的 history 对象
 - 在 RN 中跳转之前，先将路由声明在 Stack 中
- 安装
 - `yarn add @react-navigation/stack`
- 使用
 - `import { createStackNavigator } from '@react-navigation/stack'`
 - `const Stack = createStackNavigator();`



```
import * as React from 'react';
import { View, Text, Button } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';

function HomeScreen({ navigation }) {
  return ( <Button onPress={() => navigation.navigate('Details')} title="跳转到详情页" /> );
}

function DetailsScreen({ navigation }) {
  return ( <Button onPress={() => navigation.navigate('Home')} title="回首页" /> );
}

const Stack = createStackNavigator();

function App() {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="Details">
        <Stack.Screen name="Home" component={HomeScreen} />
        <Stack.Screen name="Details" component={DetailsScreen} />
      </Stack.Navigator>
    </NavigationContainer>
  );
}

export default App;
```

导航属性

```
<Stack.Navigator  
    ...属性  
/>
```

作用于整个导航（包含多个屏幕）

```
<Stack.Screen  
    ...属性  
/>
```

仅仅作用于当前屏幕

Navigator 属性

- `initialRouteName`
 - 初始化路由，即默认加载的路由
- `headerMode`
 - `float`: iOS 头部效果
 - `screen`: Android 头部效果
 - `none`: 不显示头部
- `screenOptions`

Screen 属性

- options
 - title
 - headerTitleStyle
 - headerStyle
 - headerLeft
 - headerRight
 - headerTintColor



```
<Stack.Screen
  name="Home"
  component={HomeScreen}
  options={{
    title: '首页',
    headerStyle: {
      backgroundColor: '#00b38a',
      borderBottomColor: '#00b38a',
      borderBottomWidth: 0,
      elevation: 0, // 删除 Android 上的阴影
      shadowOpacity: 0, // 删除 iOS 上的阴影
    },
    headerTintColor: '#fff',
    headerTitleStyle: {
      fontSize: 18,
      fontWeight: 'bold',
    },
    headerRight: () => ( // 声明头部右侧内容
      <TouchableOpacity onPress={() => this.takePicture()} >
        <Text style={{fontSize: 18}}>拍照</Text>
      </TouchableOpacity>
    ),
  }}>
</Stack.Screen>
```

路由与导航

BottomTab 导航

BottomTab 导航

- 安装
 - `yarn add @react-navigation/bottom-tabs`
- 使用
 - `import { createBottomTabNavigator } from '@react-navigation/bottom-tabs'`
 - `const Tab = createBottomTabNavigator();`



```
import * as React from 'react';
import { Text, View } from 'react-native';
import { NavigationContainer } from '@react-navigation/native';
import { createBottomTabNavigator } from '@react-navigation/bottom-tabs';

function HomeScreen() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Home!</Text>
    </View>
  );
}

function SettingsScreen() {
  return (
    <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
      <Text>Settings!</Text>
    </View>
  );
}

const Tab = createBottomTabNavigator();

export default function App() {
  return (
    <NavigationContainer>
      <Tab.Navigator>
        <Tab.Screen name="Home" component={HomeScreen} />
        <Tab.Screen name="Settings" component={SettingsScreen} />
      </Tab.Navigator>
    </NavigationContainer>
  );
}
```

6:08



6:09



拉勾教育

—互联网人实战大学—

Home!

Settings!

Home

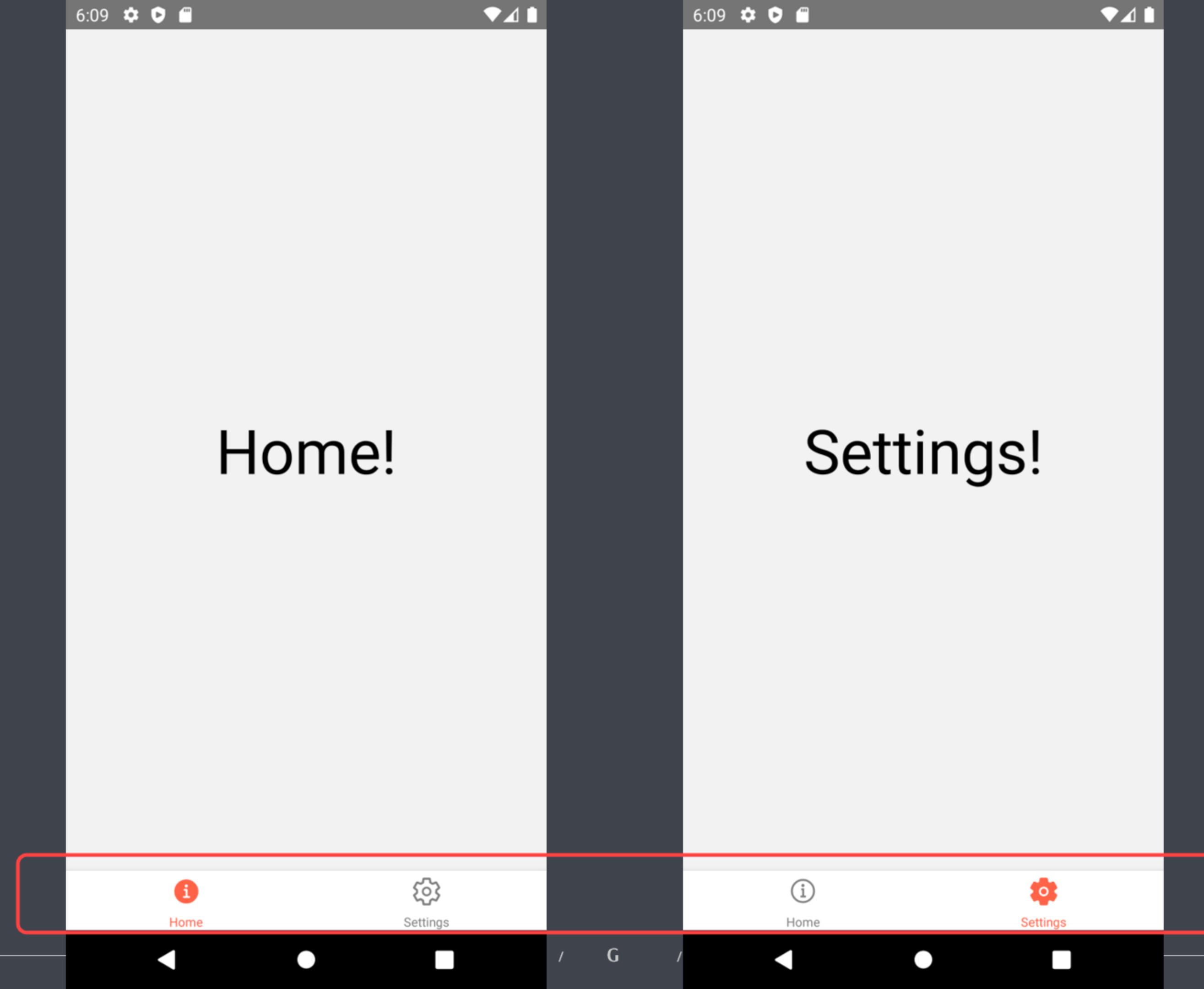
Settings

Settings

为 BottomTab 导航菜单设置图标

React-native-vector-icons（图标组件库）

- 安装
 - `npm install --save react-native-vector-icons`
- 将图标链接到应用（环境问题较多）
 - <https://github.com/oblador/react-native-vector-icons>
- 使用
 - 需要到具体的图标库官网查看
 - 例如： Ionicons、FontAwesome、AntDesign



路由与导航

Drawer 导航

Drawer 导航

- 安装

- `npm install @react-navigation/drawer`

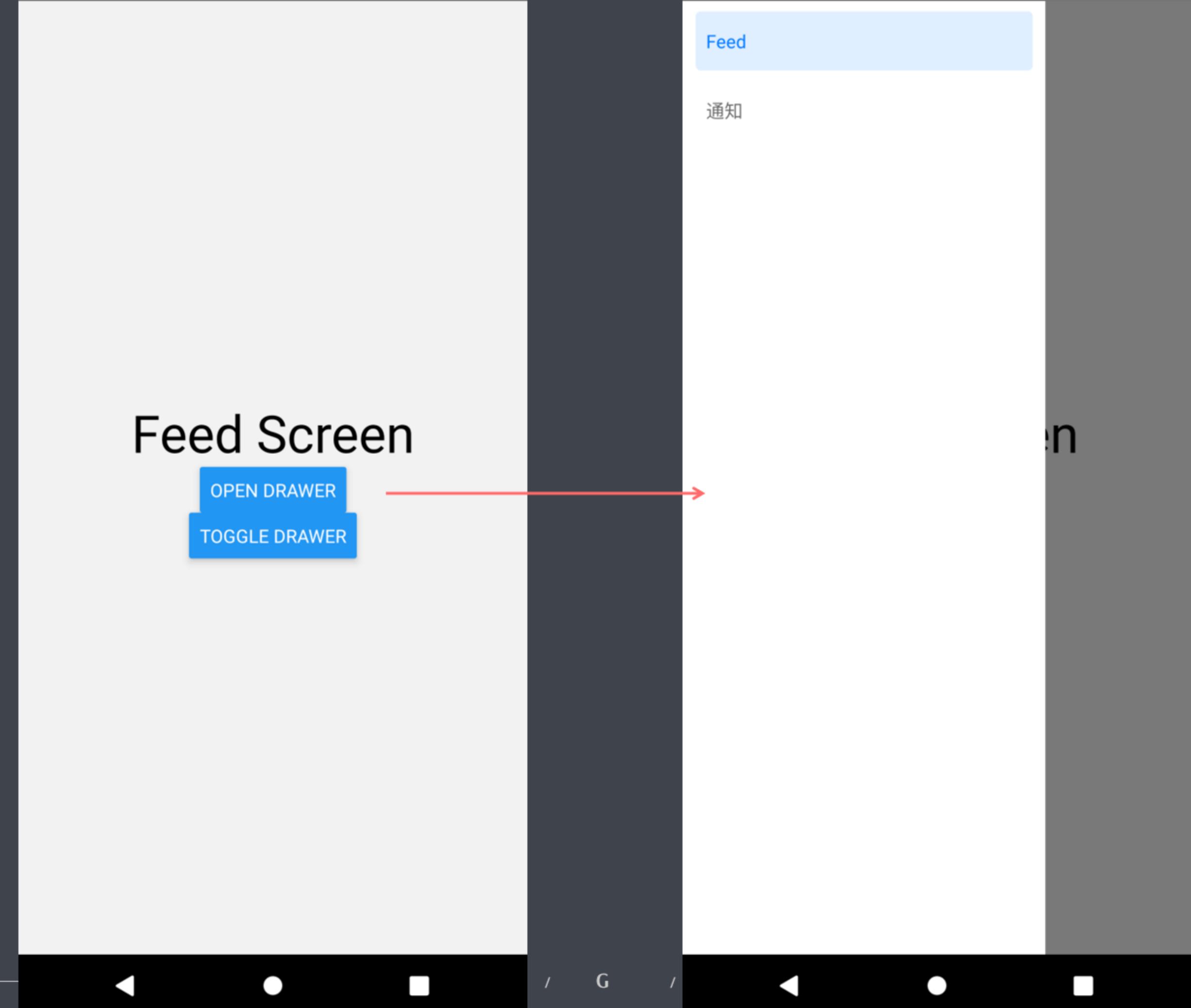
- 使用

- `import { createDrawerNavigator } from '@react-navigation/drawer';`
 - `const Drawer = createDrawerNavigator();`
 - 声明 `Drawer.Navigator` 和 `Drawer.Screen`

8:24



8:08



拉勾教育

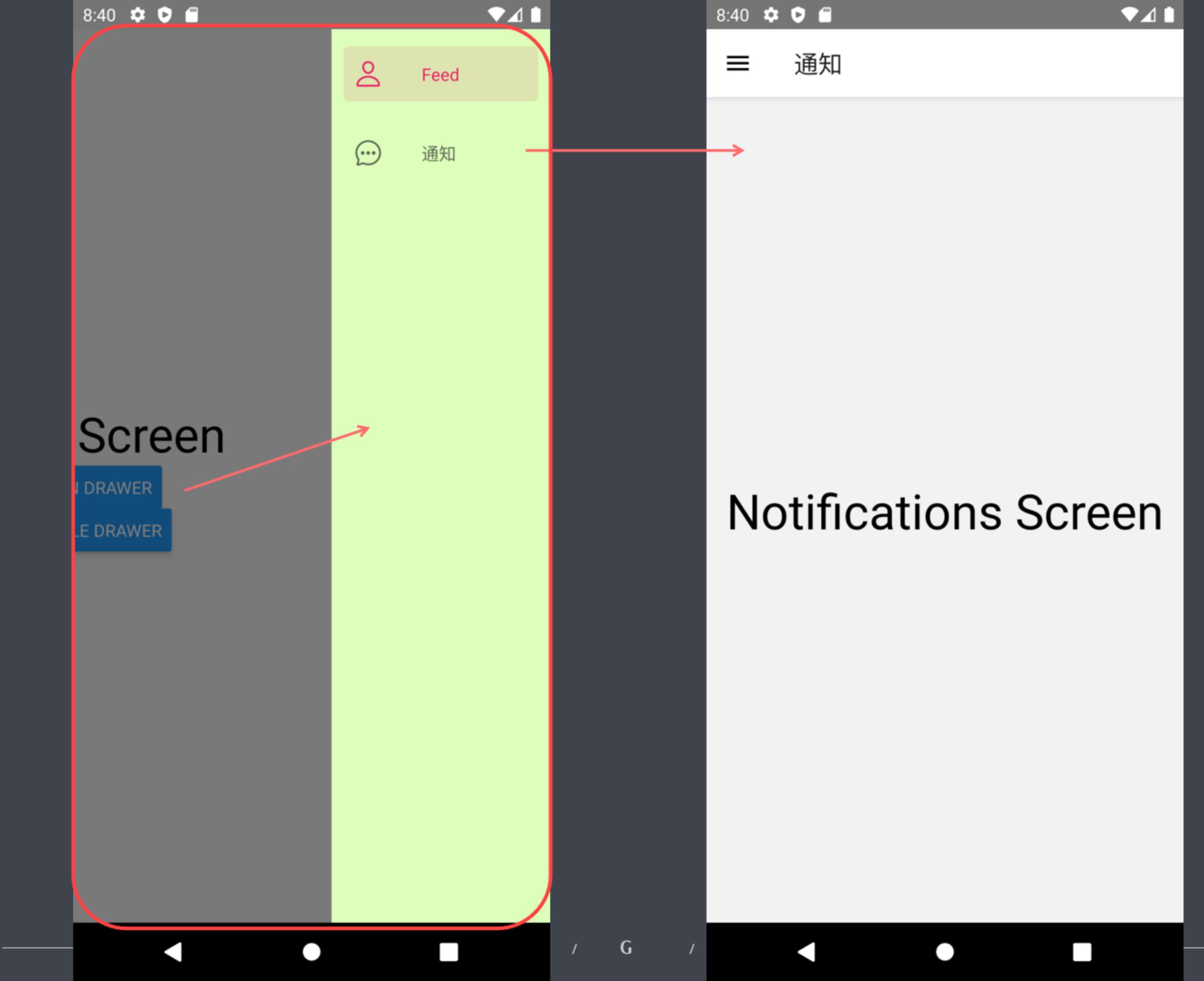
—互联网人实战大学—

配置 Drawer 导航

- Navigator 属性
 - drawerPosition: 菜单显示位置, left (默认) | right
 - drawerType: 菜单动画效果, front | back | slide | permanent
 - drawerStyle: 菜单样式
 - backgroundColor、width...
 - drawerContentOptions: 选中菜单项样式
 - activeTintColor: 当前选中菜单字体颜色
 - itemStyle: 所有菜单项样式

配置 Drawer 导航

- Screen 属性
 - options
 - title: 菜单标题
 - drawerLabel: 替代 title, 返回复杂的组件。{ focused: boolean, color: string }
 - drawerIcon: 返回图标的函数。{ focused: boolean, color: string, size: number }
 - headerShown: 是否显示 header。布尔型, 默认 false 不显示
 - headerLeft: 函数, 声明 header 左侧的显示内容
 - headerRight: 函数, 声明 header 右侧的显示内容

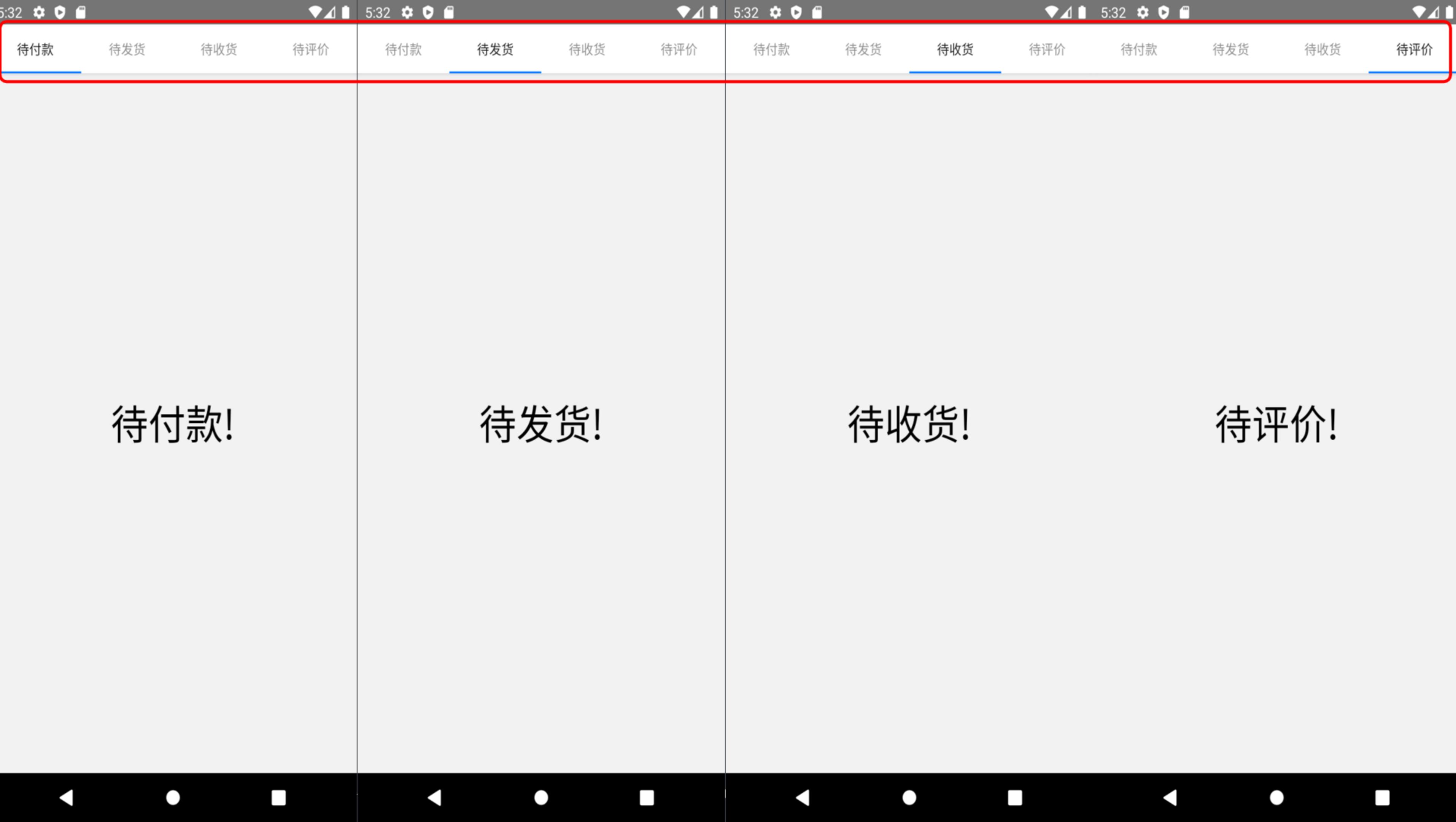


路由与导航

MaterialTopTab 导航

MaterialTopTap 导航

- 安装
 - `yarn add @react-navigation/material-top-tabs react-native-tab-view`
- 使用
 - `import { createMaterialTopTabNavigator } from '@react-navigation/material-top-tabs' ;`
 - `const Tab = createMaterialTopTabNavigator () ;`
 - 声明 Tab.Navigator 和 Tab.Screen



配置 MaterialTopTap 导航

- Navigator 属性
 - tabBarPosition: Tab显示位置。默认 top，可以设置 bottom
 - tabBarOptions: 包含 tabBar 组件属性的对象
 - activeTintColor - 当前菜单的标题或图标颜色。
 - inactiveTintColor - 非当前菜单的标题或图标颜色。
 - showIcon - 是否显示图标，默认是 false。
 - showLabel - 是否显示文字，默认是 true。
 - tabStyle - 标签样式对象。
 - labelStyle - 标签文字样式对象。这里指定的颜色，会覆盖 activeTintColor 和 inactiveTintColor 的值。
 - iconStyle - 图标样式对象。

配置 MaterialTopTap 导航

- Screen 属性
 - options: 设置 Screen 组件的对象
 - title - 设置显示标题
 - tabBarIcon - 设置标签图标（需要现在 Navigator 中指定 showIcon: true ）
 - 其值为函数，包含两个参数: { focused: boolean, color: string }
 - focused 用来判断标签是否获取焦点，color 为当前标签的颜色
 - tabBarLabel - 设置标签文字内容（当未定义时，会使用 title ）
 - 其值为函数，包含两个参数: { focused: boolean, color: string }
 - focused 用来判断标签是否获取焦点，color 为当前标签的颜色

待付款!



待付款



待发货



待收货



待评价

路由与导航

嵌套

路由嵌套

- 在一个导航的内部，渲染另一个导航
- 示例：
 - Stack.Navigator
 - Home (Tab.Navigator)
 - Feed (Screen)
 - Messages (Screen)
 - Profile (Screen)
 - Settings (Screen)



```
function Home() {  
  return (  
    <Tab.Navigator>  
      <Tab.Screen name="Feed" component={Feed} />  
      <Tab.Screen name="Messages" component={Messages} />  
    </Tab.Navigator>  
  );  
}
```

```
function App() {  
  return (  
    <NavigationContainer>  
      <Stack.Navigator>  
        <Stack.Screen name="Home" component={Home} />  
        <Stack.Screen name="Profile" component={Profile} />  
        <Stack.Screen name="Settings" component={Settings} />  
      </Stack.Navigator>  
    </NavigationContainer>  
  );  
}
```

路由与导航

路由传参

路由传参

- 传递参数

- `navigation.navigate('路由名称', {KEY:123})`

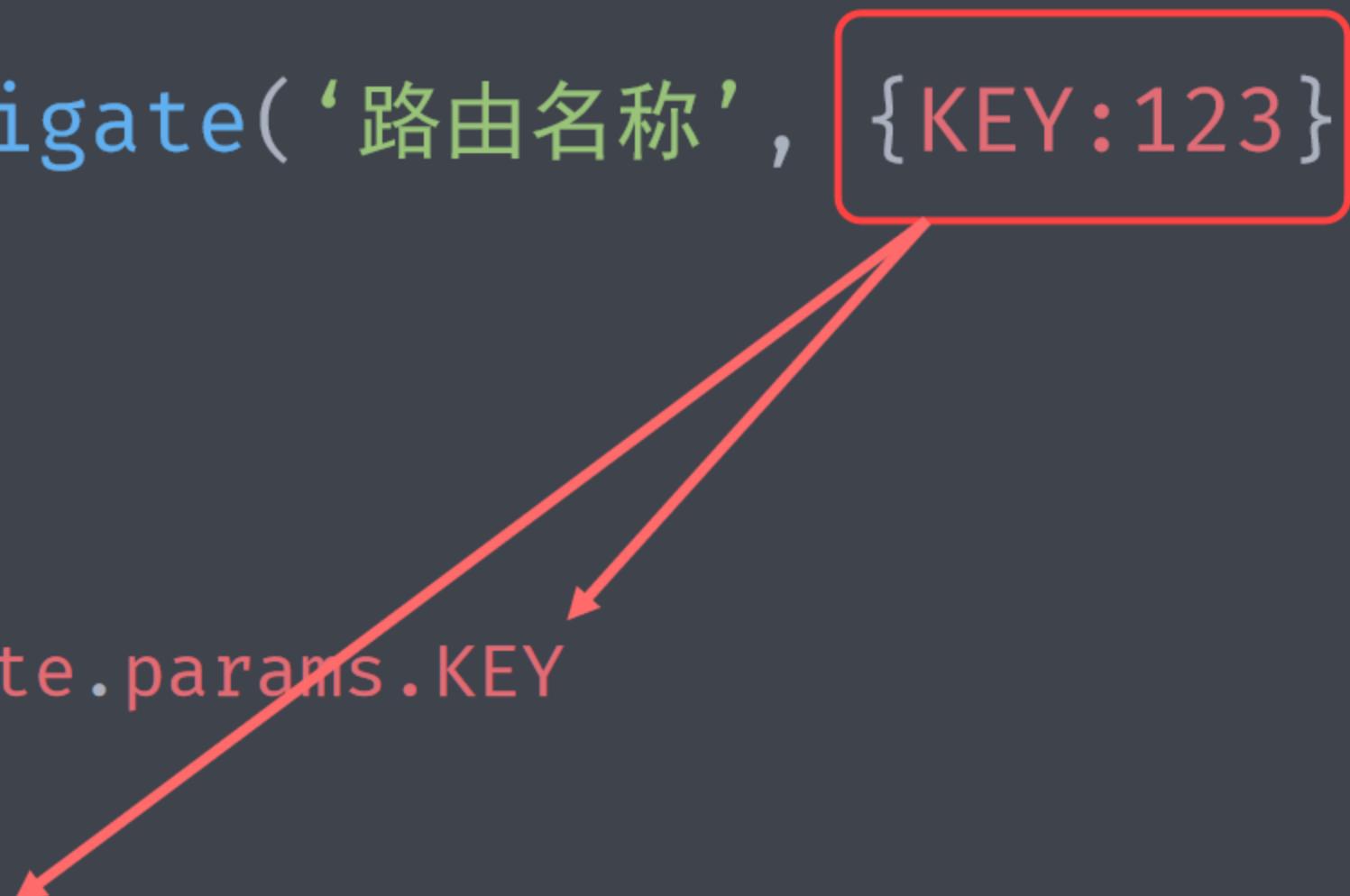
- 接收参数

- 类组件

- `this.props.route.params.KEY`

- 函数组件

- `route.params.KEY`



架构原理

架构原理

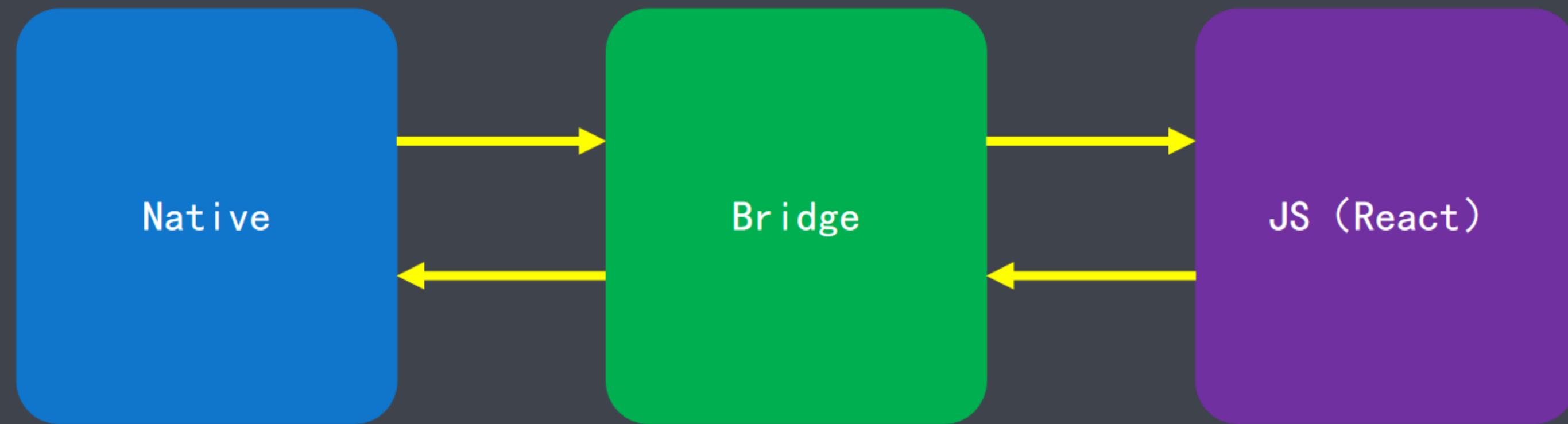
STEPS

1. 现有架构
2. 新架构

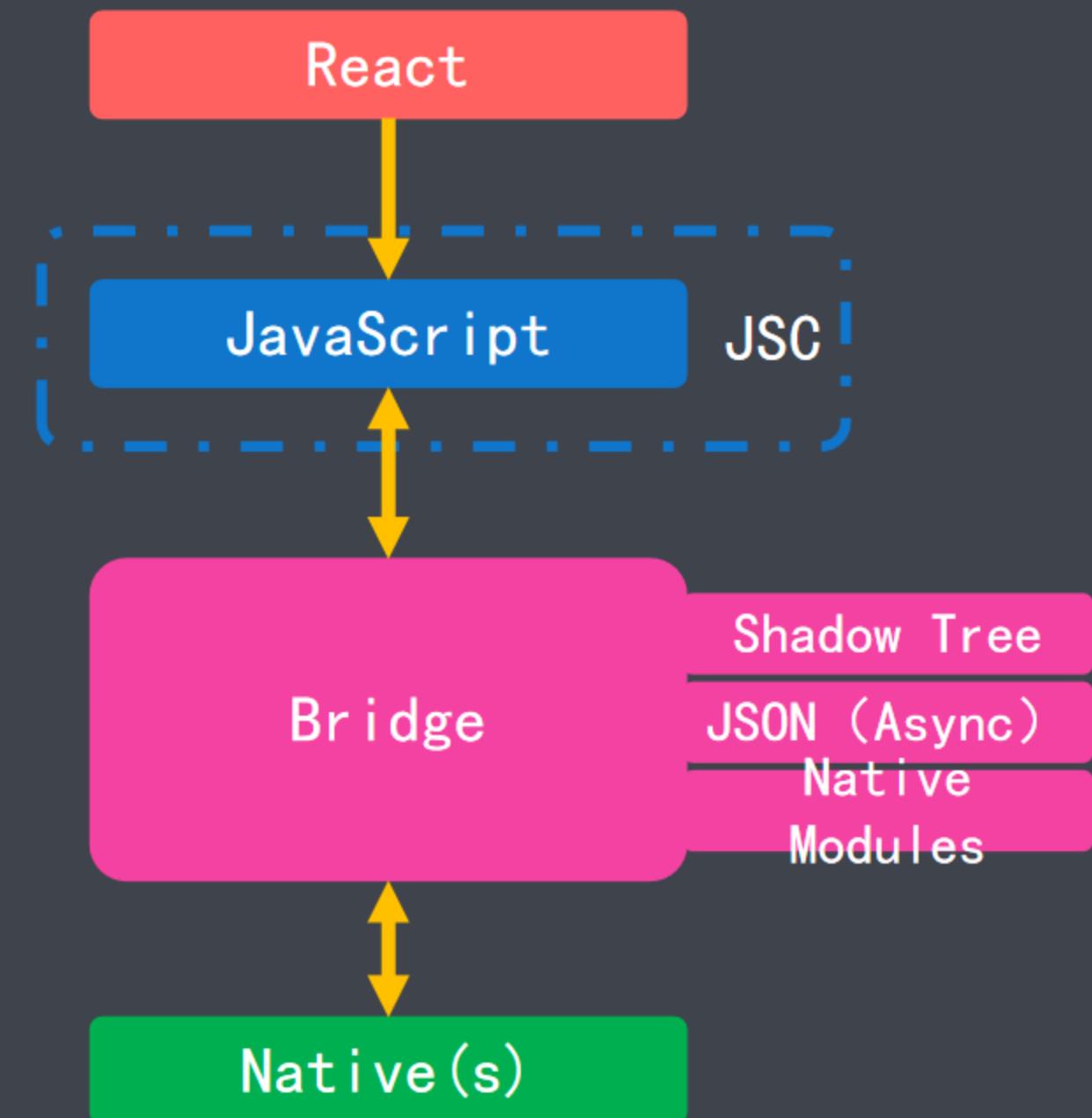
架构原理

现有架构

架构设计



架构设计



架构原理

线程模型

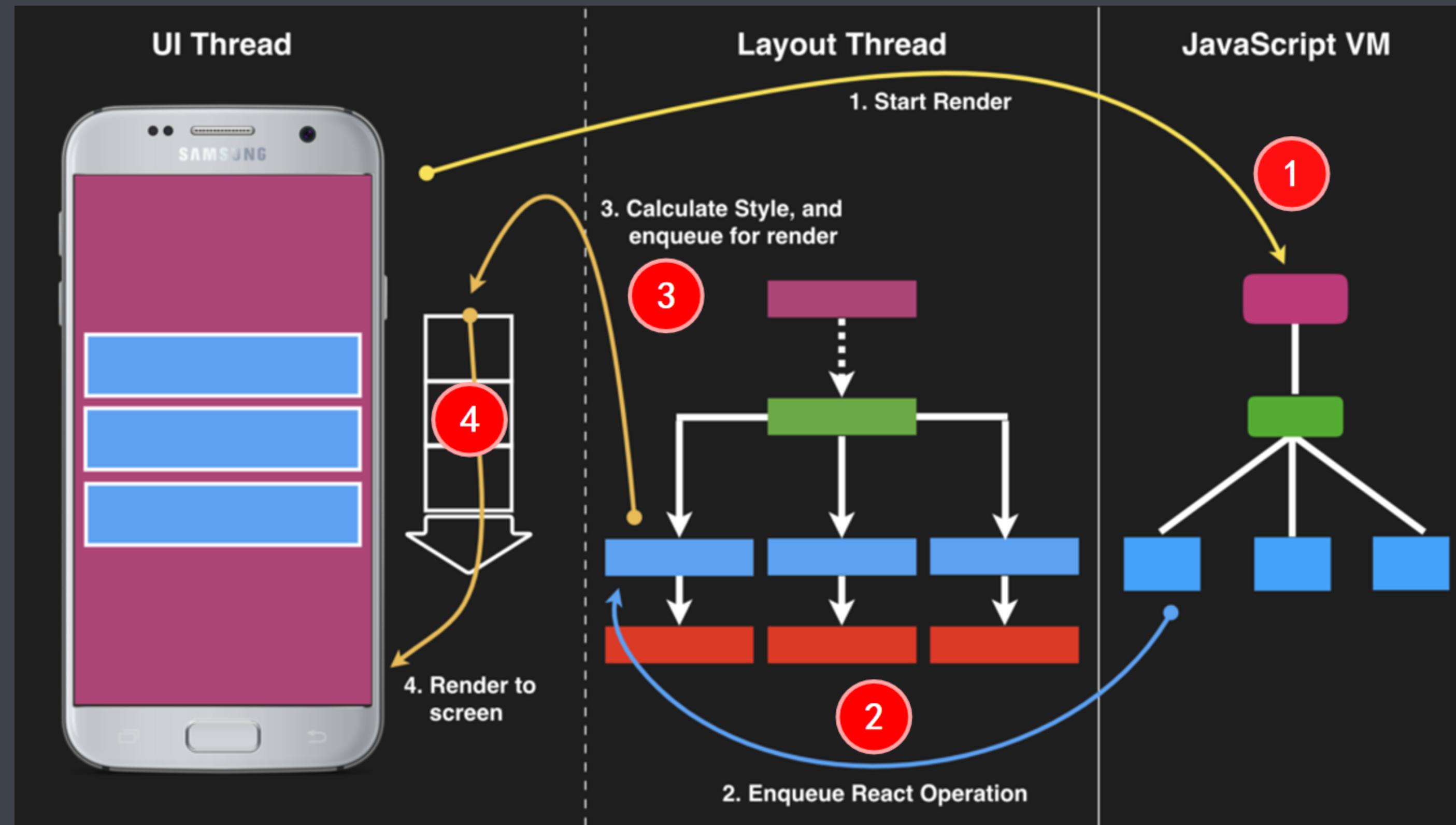
线程模型

- JS 线程
 - JS 代码的执行线程，将源码通过 Metro 打包后，传给 JS 引擎进行解析
- Main 线程 (UI 线程 或 原生线程)
 - 主要负责原生渲染 (Native UI) 和调用原生模块 (Native Modules)
- Shadow 线程 (Layout 线程)
 - 创建 Shadow Tree 来模拟 React 结构树 (类似虚拟 DOM)
 - 再由 Yoga 引擎将 Flexbox^A 等样式，解析成原生平台的布局方式

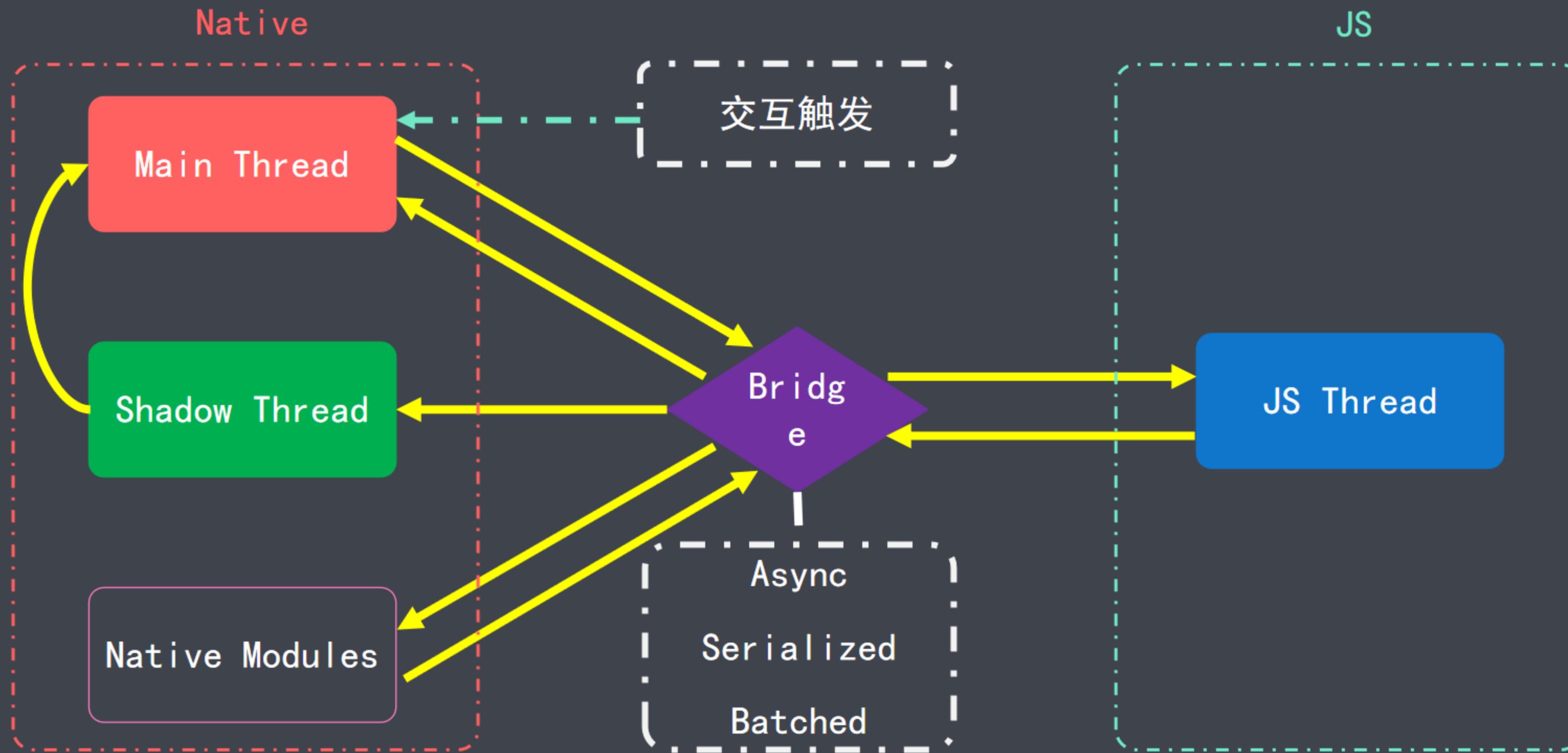
Android 或 iOS 中的布局，不支持 Flexbox

渲染机制 - 视图渲染 - 项目启动



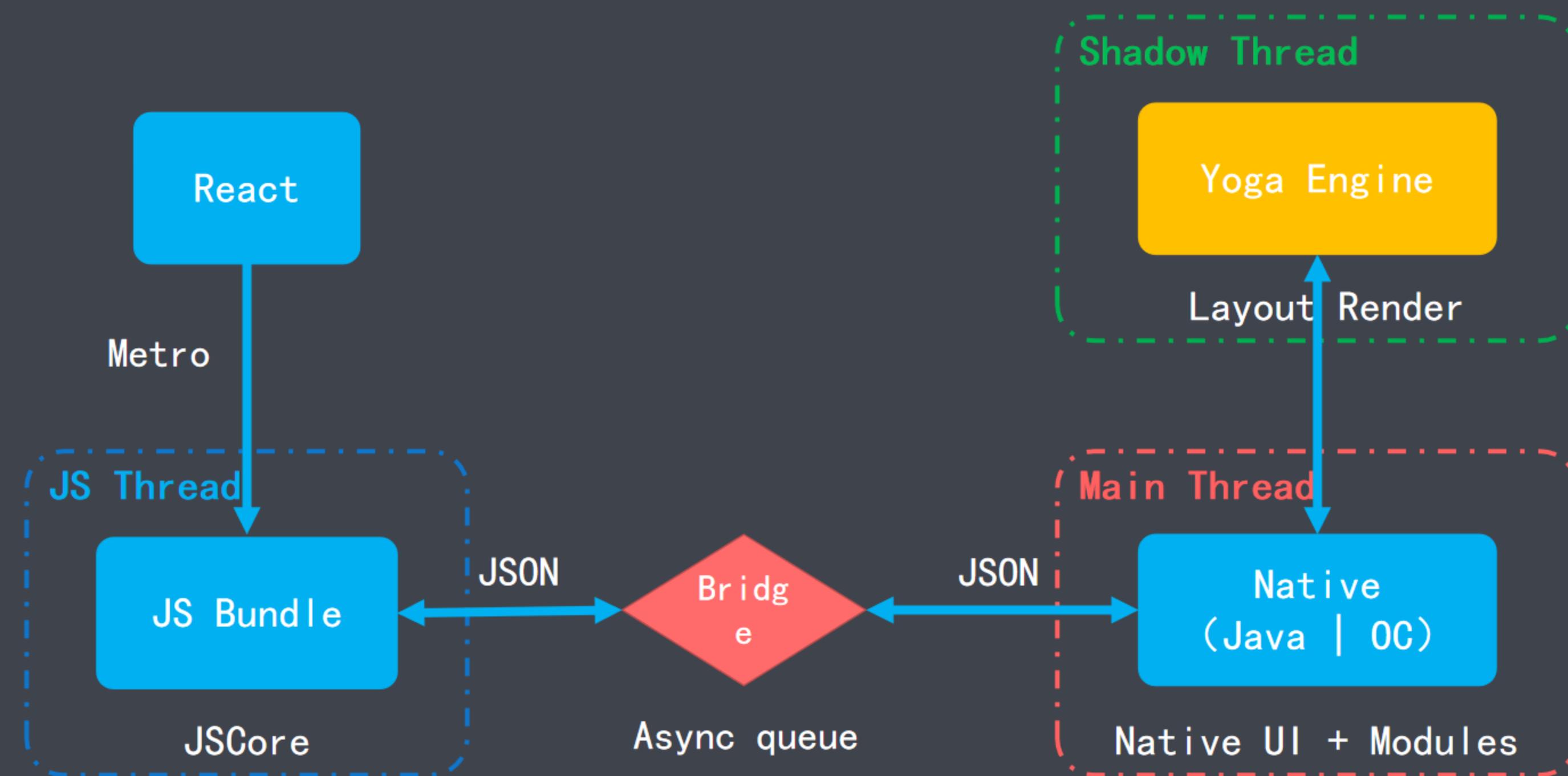


线程模型 - 线程间交互



从启动过程，来理解架构设计

架构设计 - 启动过程



从应用过程，来理解架构设计

Native

1 Event
(touch, timer, networks, etc.)

2 Collect data and notify JS

3 Serialized payload

4 Process event

8 Update UI
(if needed)

7 Process commands

6 Serialized response

5 Call 0 – ∞
native methods

Bridge

JavaScript



Native

```
[_bridge enqueueJSCall:@"RCTEventEmitter.receiveTouches"  
    args:@[@["end",  
            @{@"x": @42, @"y": @106}]];
```

1

Bridge

```
[  
    'EventEmitter', 'receiveTouches',  
    ['end', {'x': 42, 'y': 106}]  
]
```

2

JavaScript

```
call('EventEmitter', 'receiveTouches', [{x: 42, y: ...}])
```

3

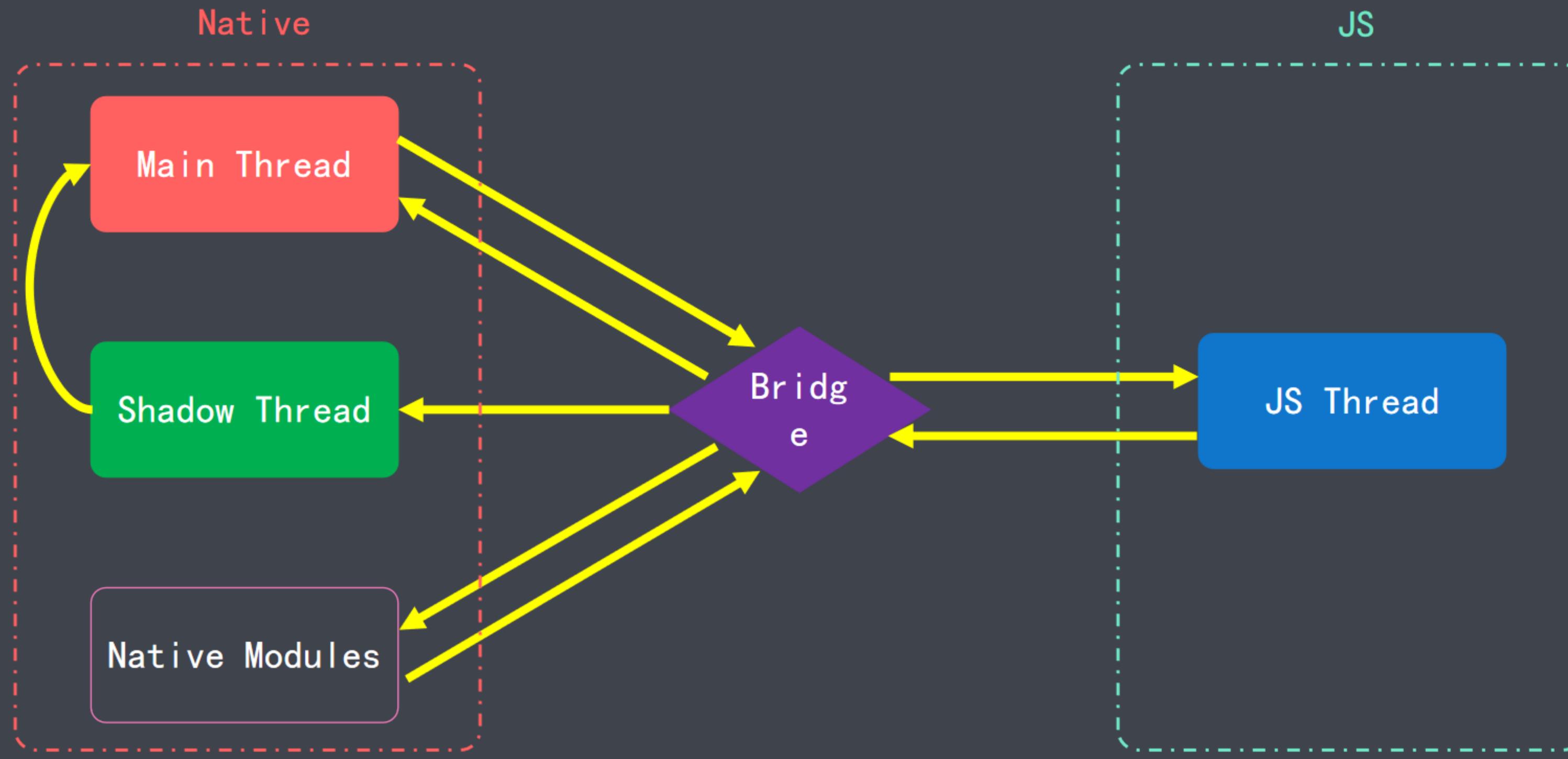
架构原理

新架构

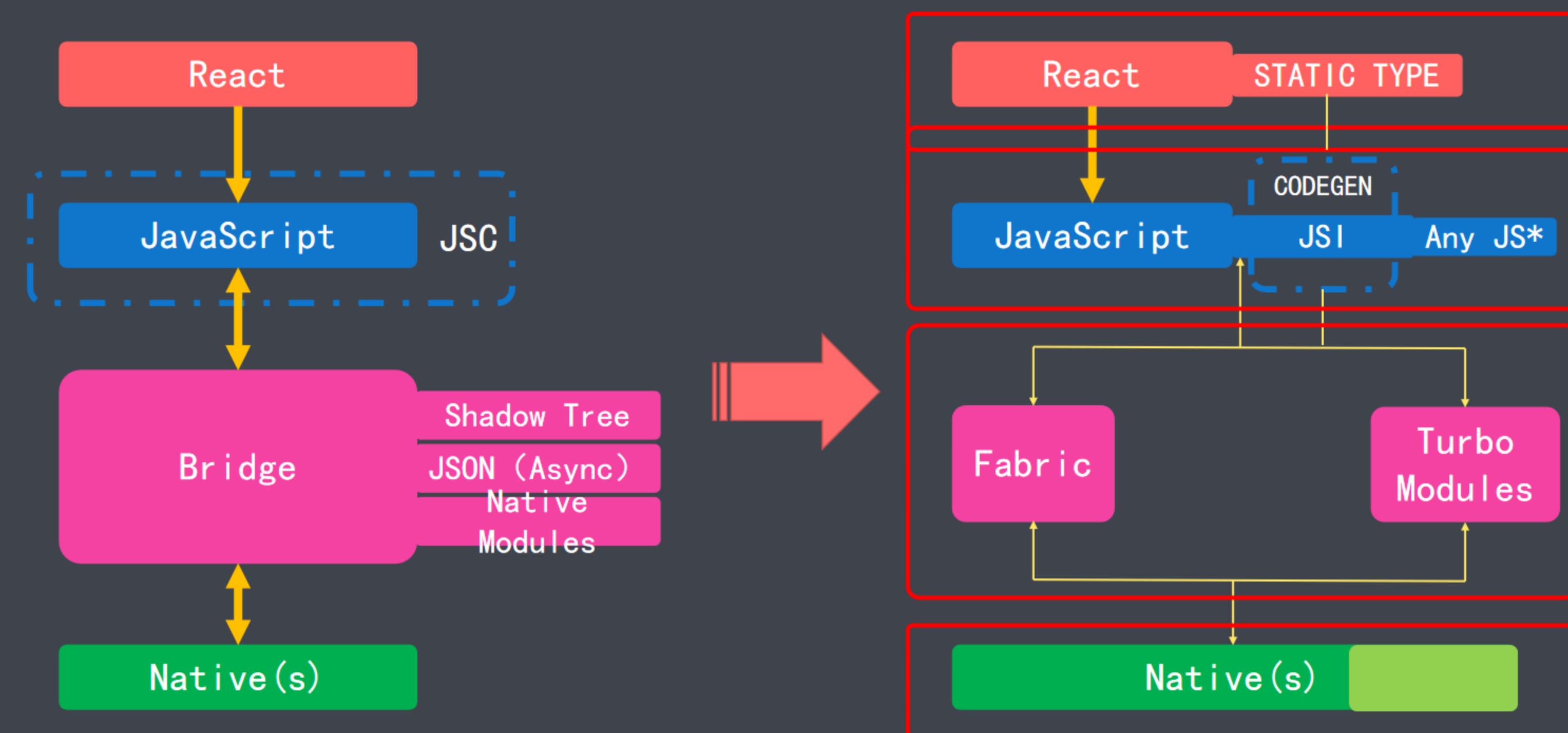
RN 重构背景

- 之前的版本，存在诸多性能问题
- 受到 Flutter 等后起之秀的压力
- 2018 年 6 月，提出重构计划

线程模型 - 线程间交互



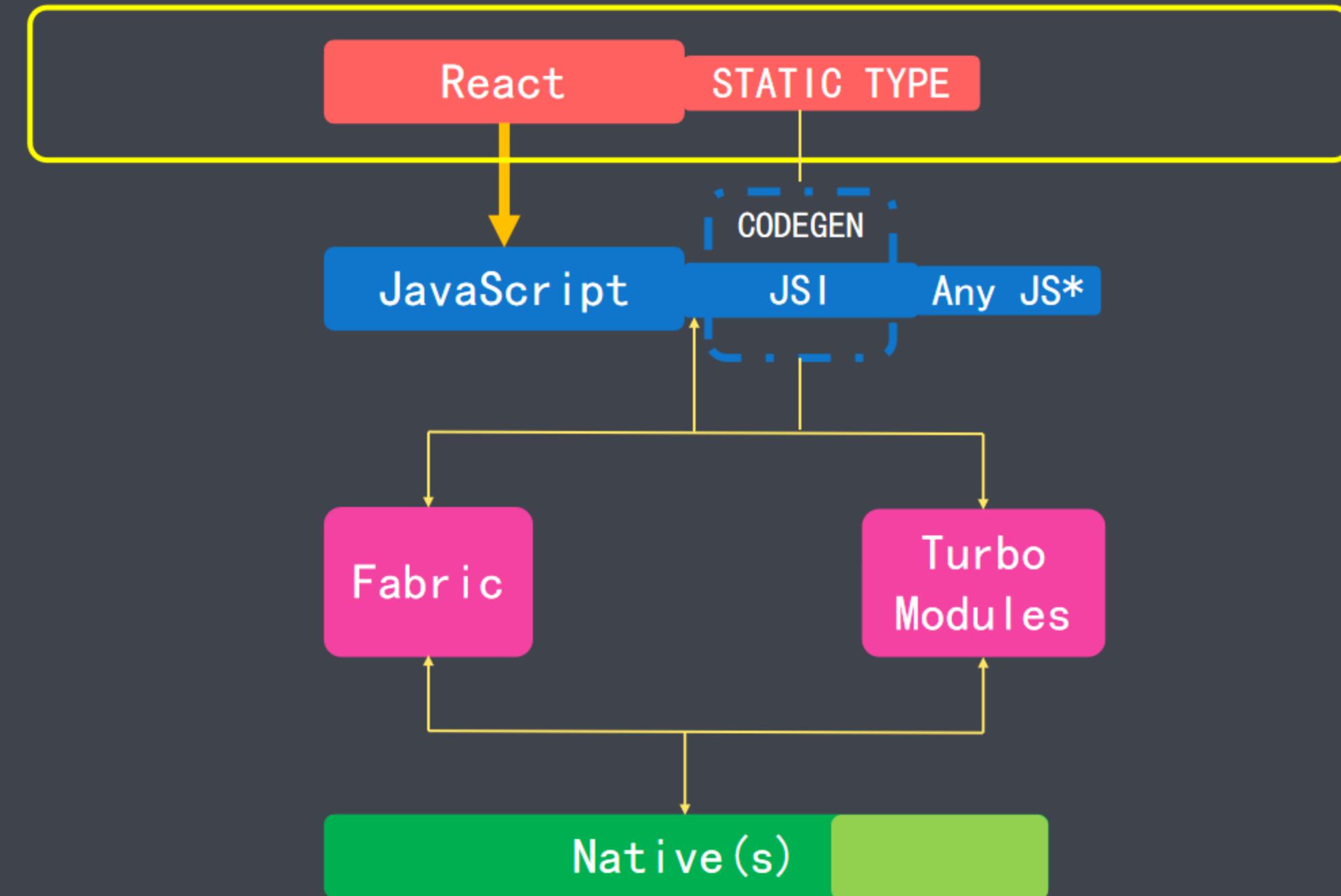
架构设计 - 新旧架构对比



三大改动

- JavaScript 层：
 - 支持 React 16+ 的新特征
 - 增强 JS 静态类型检查（CodeGen）
 - 引入 JSI，允许替换不同的 JavaScript 引擎
- Bridge 层：
 - 划分成 Fabric 和 TurboModules 两部分，分别负责 UI 管理与 Native 模块
- Native 层：
 - 精简核心模块，将非核心部分拆分出去，作为社区模块，独立更新维护

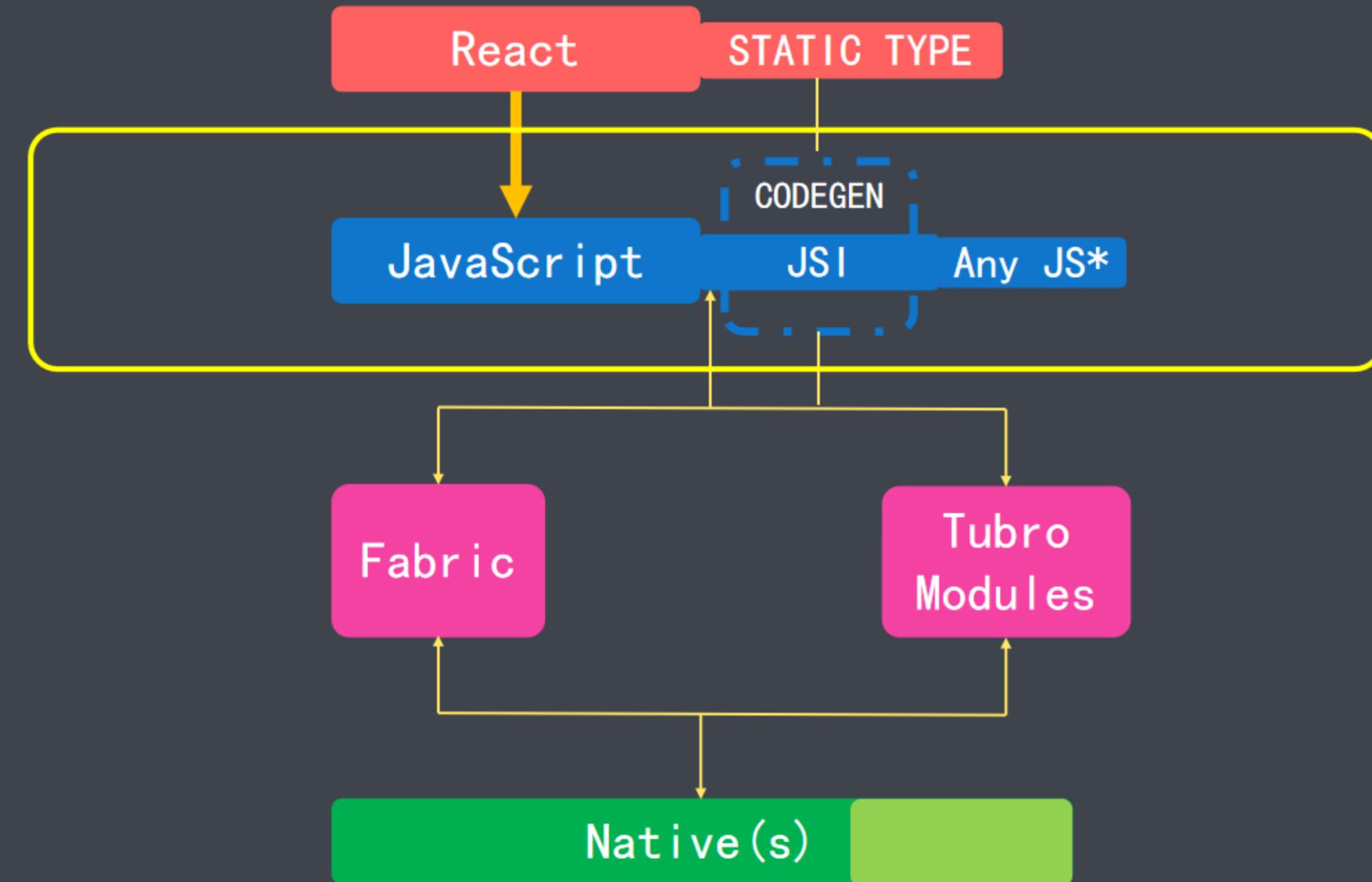
架构设计 - 增强 JavaScript 类型安全



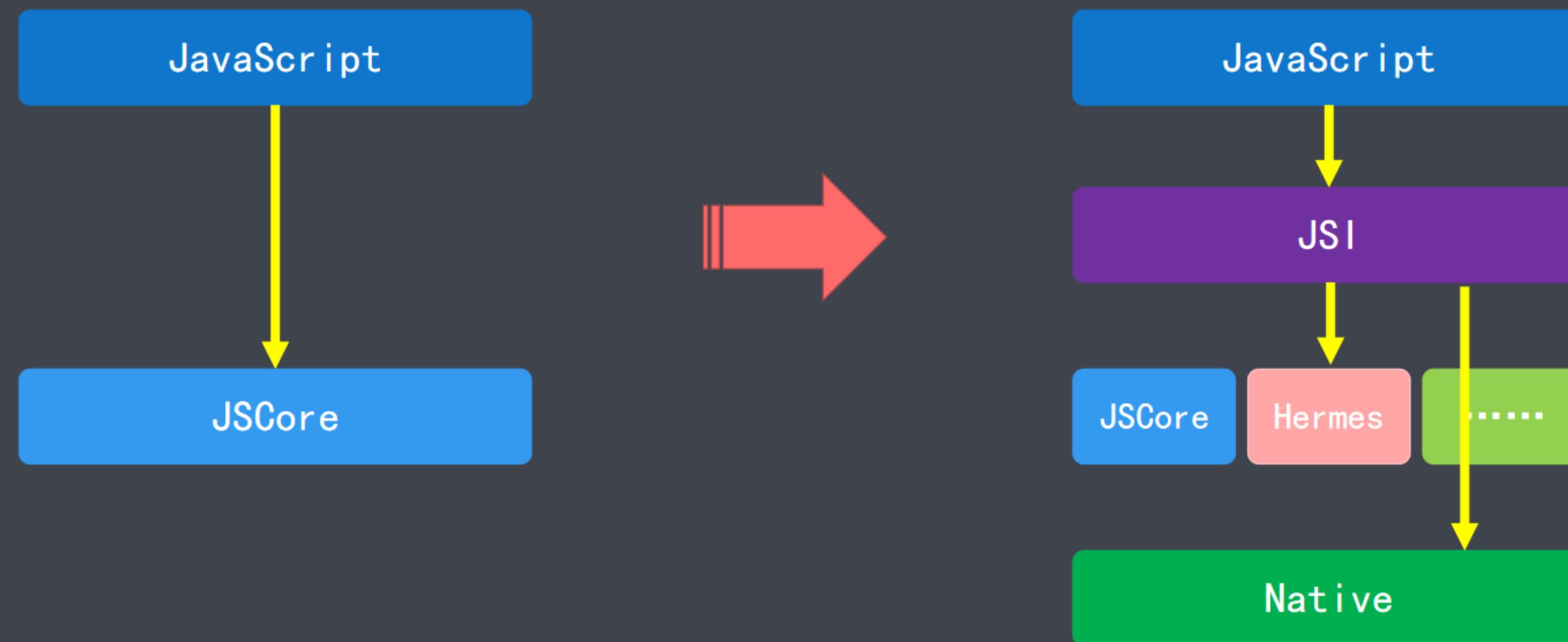
CodeGen

- CodeGen 是 FaceBook 推出的代码生成工具
- 通过 CodeGen，自动将 Flow 或者 TypeScript 等有静态类型的 JS 代码翻译成 Fabric 和 TurboModules 使用的接口文件。
- 加入类型约束后的作用
 - 减少了数据类型错误
 - 减少了数据验证的次数，提高了通信性能

架构设计 - 引入 JSI (JavaScript Interface)



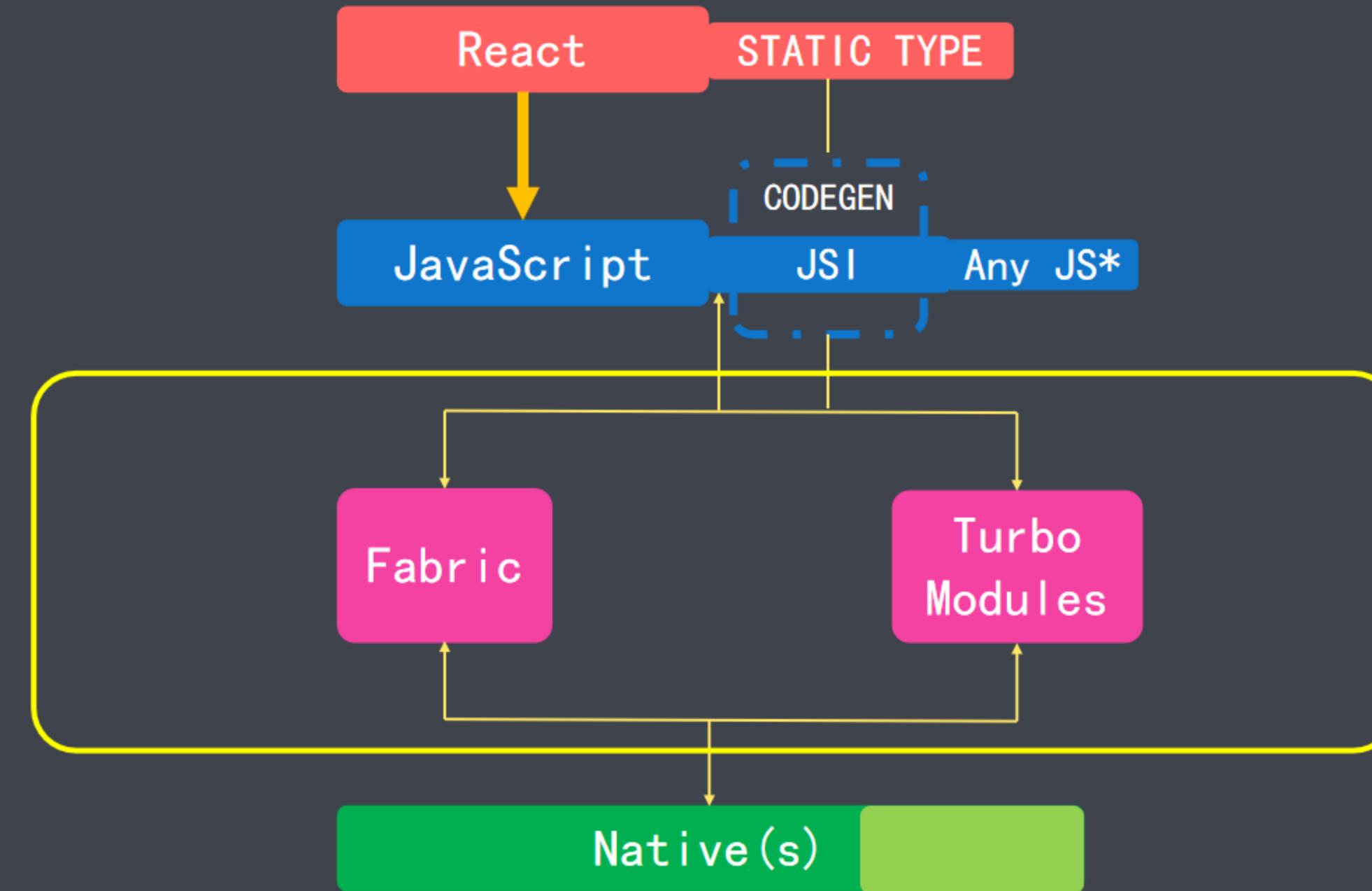
架构设计 - 引入 JSI (JavaScript Interface)



JSI (JavaScript Interface)

- JSI 是一个用C++写成的轻量级框架。其作用主要有两个：
 - 通过 JSI，可以实现 JS 引擎的更换
 - 通过 JSI，可以通过 JS 直接调用 Native
 - JS 对象可以直接获得 C++ 对象(Host Objects)引用，从而允许 JS 与 Native 的直接调用
 - 减少不必要的线程通信
 - 省去了序列化和反序列化的成本
 - 减轻了通信压力，提高了通信性能

架构设计 - 重构 Bridge 层



优化 Bridge 层

- Fabric
 - Fabric 是整个架构中的新 UI 层
 - 简化了之前渲染流程
- Turbo Modules

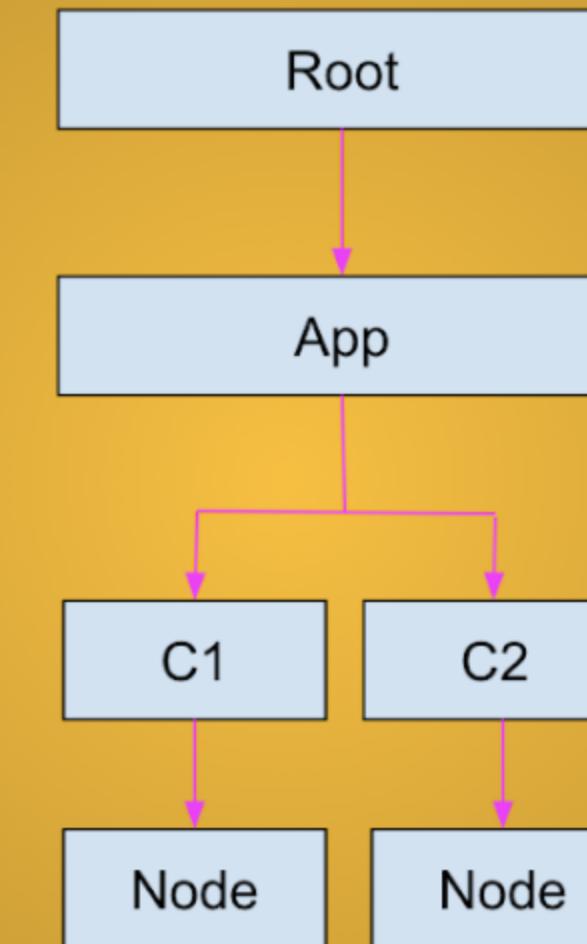
Native Thread or Main Thread

- Synchronous Thread



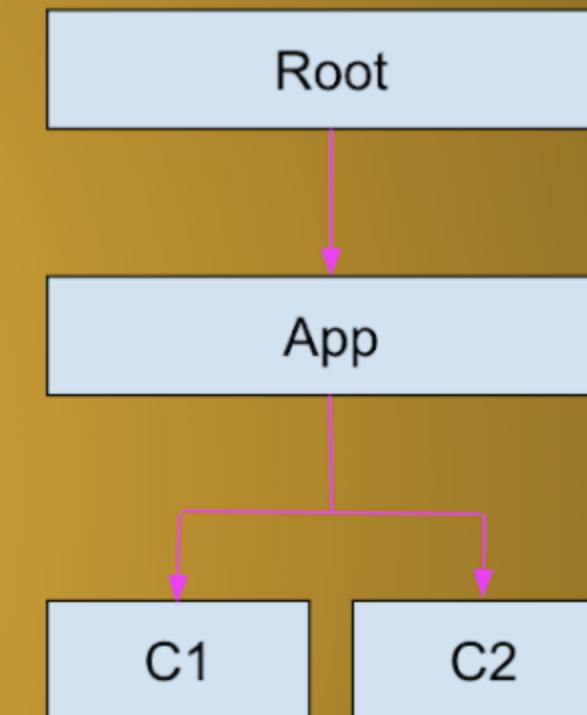
Shadow Thread

- Asynchronous Thread

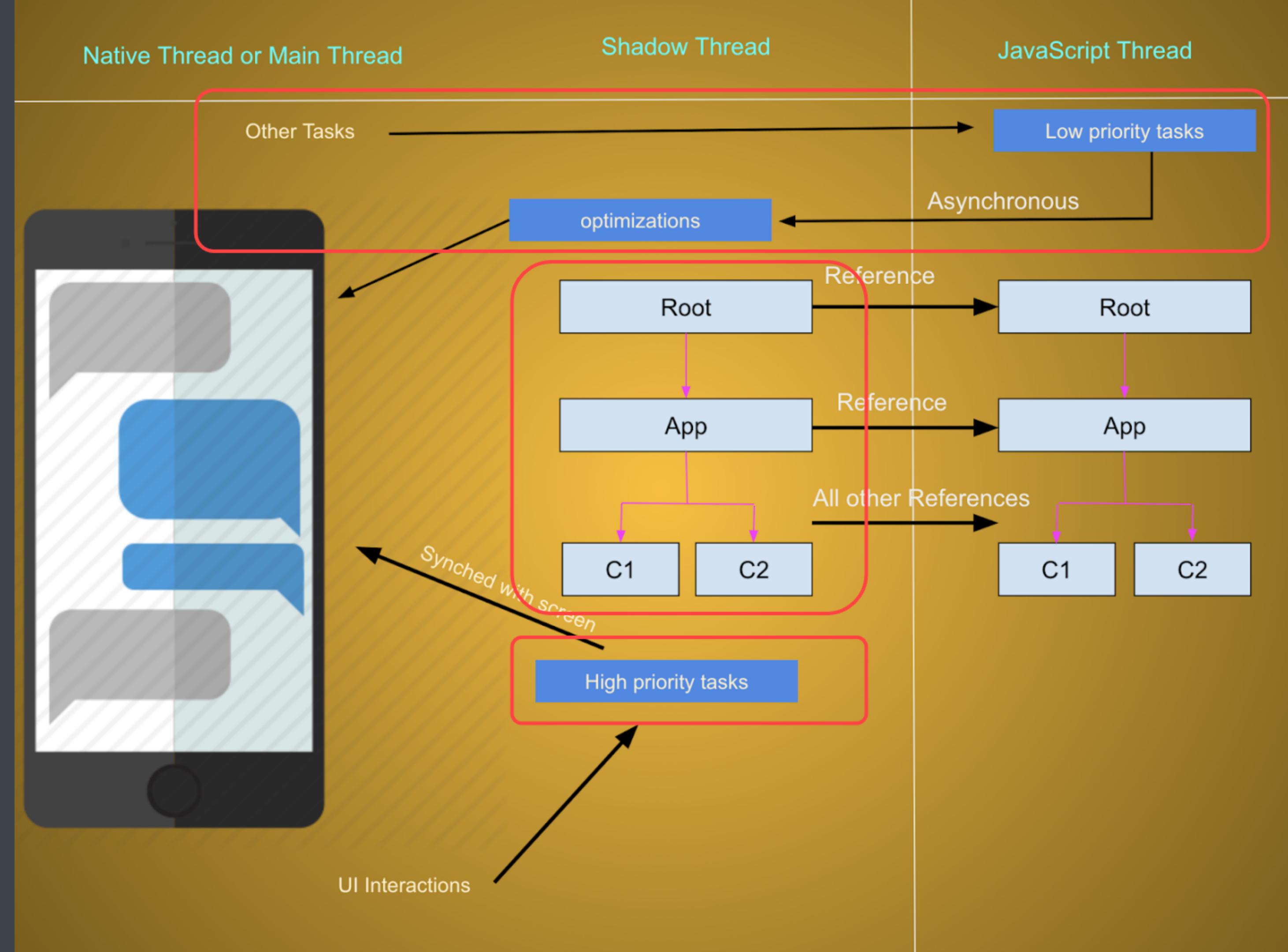


JavaScript Thread

- Asynchronous Thread



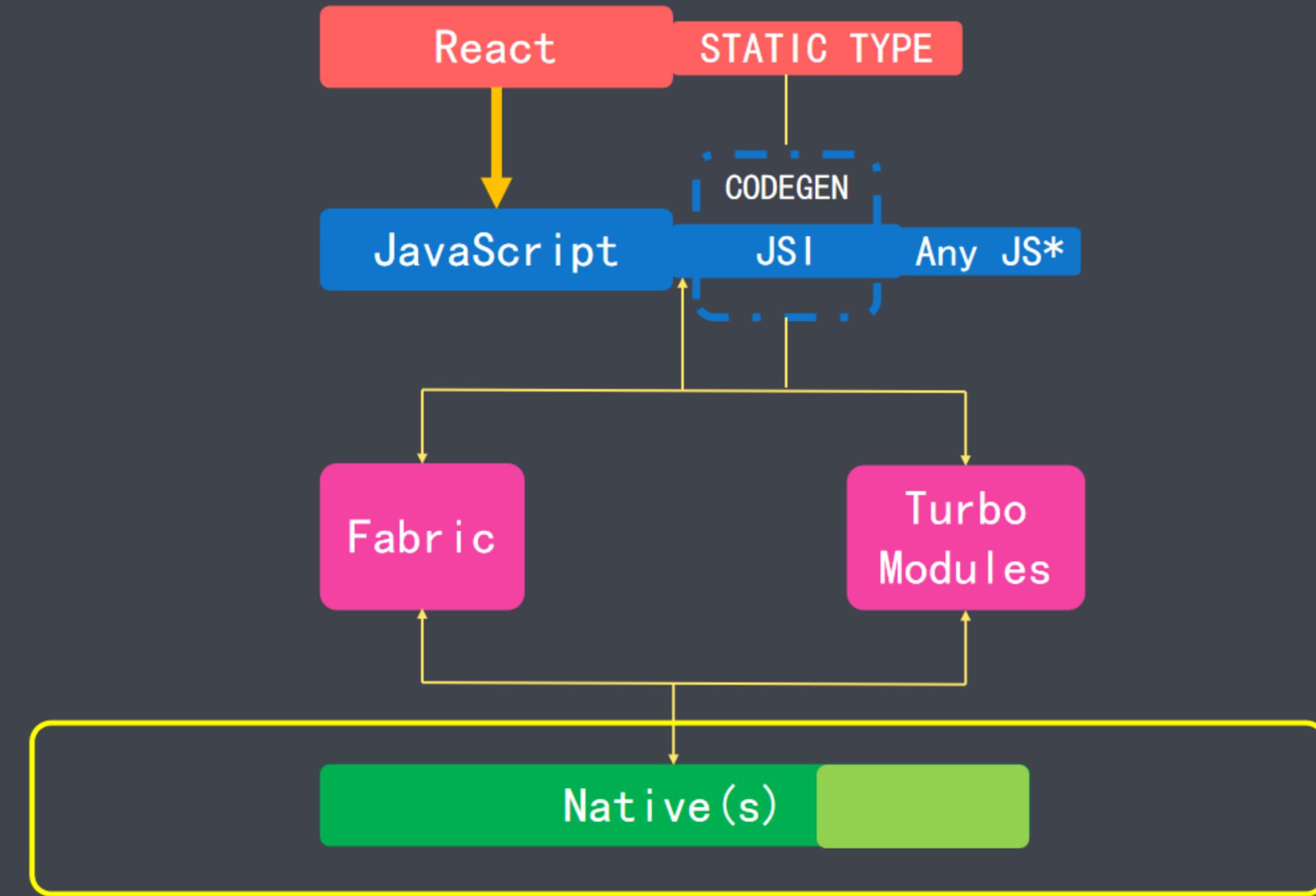
UI Interactions



优化 Bridge 层

- Fabric
 - Fabric 是整个架构中的新 UI 层
 - 简化了之前渲染
- Turbo Modules
 - 通过 JSI，可以让 JS 直接调用 Native 模块，实现同步操作
 - 实现 Native 模块按需加载，减少启动时间，提高性能

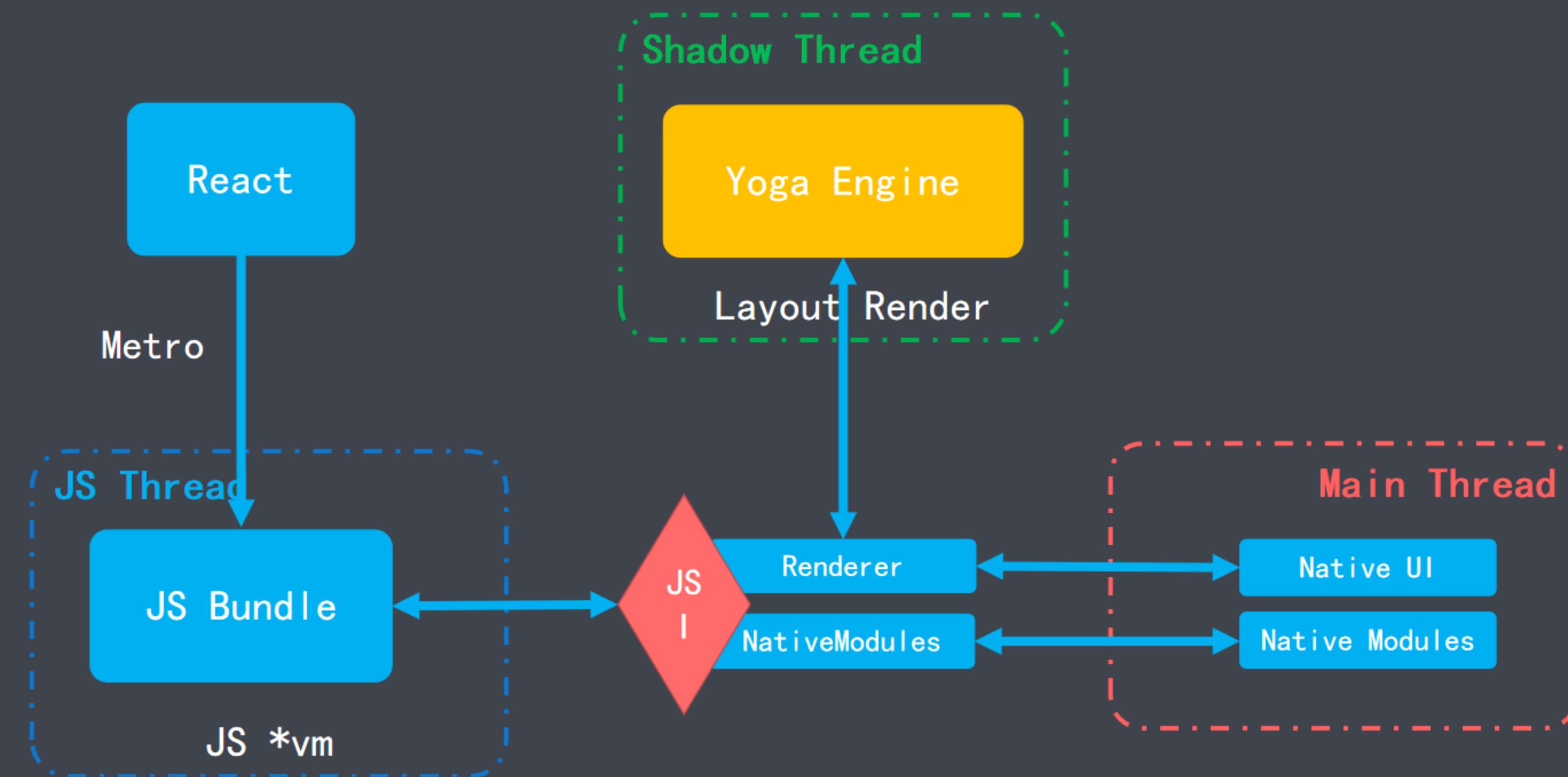
架构设计 - 精简核心模块



精简核心模块 (Lean Core)

- 将 react-native 核心包进行瘦身
 - RN 推出多年，其核心包太过臃肿
 - 有些包在项目中用不到，每次也要引入，造成资源浪费
- 非必要的包，移到社区模块，单独维护
 - 例如：AsyncStorage、WebView 等

新架构 - 启动过程



原计划2020年第四季度重构完成，现在看来时间会延后

项目实践

项目实践

STEPS

1. 项目展示
2. 数据接口
3. UI 界面
4. 状态管理
5. 项目优化



看更大的世界!

走起

Welcome!

用户名

密码

忘记密码?

登录

注册

©2020 版权所有 lagou.com

用户注册

用户名

密码

确认密码

注册用户默认同意 服务条款 and 保密协议

注册

登录

©2020 版权所有 lagou.com

首页

拍照



扫一扫

付款码



出行



卡包



美国 加利福尼亚（加州） 山景城

紫外线指数
强

洗车指数
较适宜

钓鱼指数
较适宜

2020-12-02



晴 15°C

5°C 晴间多云



2020-12-03



晴 16°C

5°C 晴



2020-12-04



晴 17°C

4°C 晴



©2021 版权所有 lagou.com



Home



News



RNStack



Profile

今日新闻

头条 社会 国内 国际 娱乐 体育

就在3天前，嘎子哥终究还是“翻车”了，潘长江彻底死心！



2020-12-02 19:54 雷Sir娱乐圈



吴刚走红毯秀恩爱，和发妻相识34年是珍珠婚，俩人却常忽视儿子



2020-12-02 19:47 会火



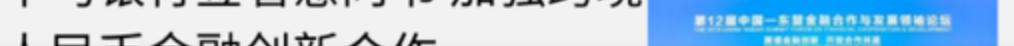
每到冬季，我家必做这碗汤，5块钱煮1锅，温暖滋润，清淡养胃！



2020-12-02 19:45 小面姨美食日记



中马银行签署意向书 加强跨境



← 新闻详情



每到冬季，我家必做这碗汤，5块钱煮1锅，温暖滋润，清淡养胃！

小面姨美食日记 2020-12-02 19:45



莫愁厨路无知己，谁人不识小面姨。大家好，我是小面姨。今天小面姨给大家分享一道“香菜豆腐汤”的做法。最近这段时间，家里吃香菜的频率又多了起来，大家知道这是为什么吗？原因其实很简单，主要是家里菜地里的香菜最近成熟了，到了该吃它的时候了。在我们家里，香菜是全家人最喜欢吃的一种蔬菜，它独特的香味，往往能提升美食的口感。所以为了家人能及时吃到鲜嫩的香菜，我就在自家菜地里专门开辟了一小块地方，种植了香菜，现



Home



News



RNStack



Profile

项目实践

数据接口

数据接口

STEPS

1. 申请接口
2. 调试接口
3. 使用接口

数据接口

申请接口

申请数据接口

后端工程师

例如：Express、Koa 开发的

模拟接口（Mock API）

例如：Rap2

第三方接口

例如：和风天气

申请第三方接口

- 注册账号
 - <https://id.qweather.com/#/register> 和风天气
 - <https://www.juhe.cn/register> 聚合数据
- 创建应用并申请密钥（key）
 - key 是调用接口的凭证
- 开发集成（开发文档）
 - 请求接口的语法
 - 返回数据的示例

申请数据接口

- 和风天气
 - 城市信息搜索: <https://dev.qweather.com/docs/api/geo/>
 - 3天预报: <https://dev.qweather.com/docs/api/weather/>
 - 生活指数: <https://dev.qweather.com/docs/api/indices/>
- 聚合新闻
 - 新闻头条: <https://www.juhe.cn/docs/api/id/235>

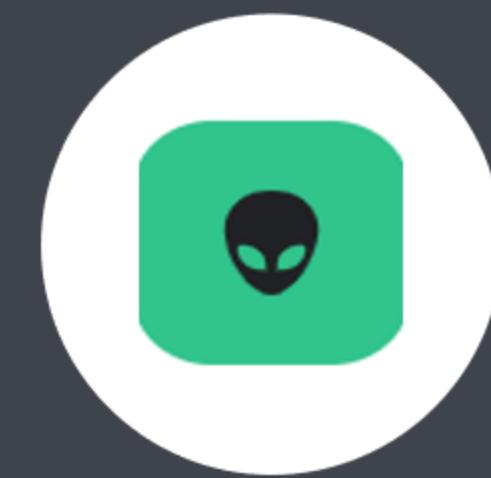
数据接口

调试接口

接口调试工具



postman



hopscotch



insomnia

insomnia

- 下载（Insomnia Core）
 - <https://insomnia.rest/>
- 配置
 - 将接口调用中使用的公共内容，提升为变量，例如：key
 - 不同的项目，具有不同的环境变量

接口调试中的公共内容

`https://devapi.qweather.com/v7/weather/3d?key=xxxx1234xxxx&location=116.29,39.95`

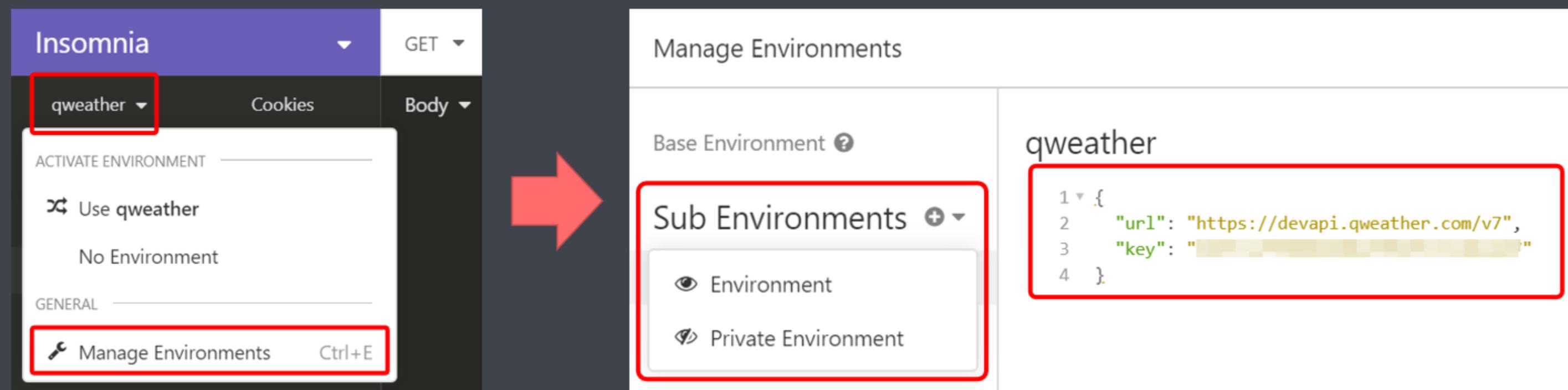
`https://devapi.qweather.com/v7/air/now?key=xxxx1234xxxx&location=104.29,36.21`

`https://devapi.qweather.com/v7/indices/1d?key=xxxx1234xxxx&location=98.56,34.67`

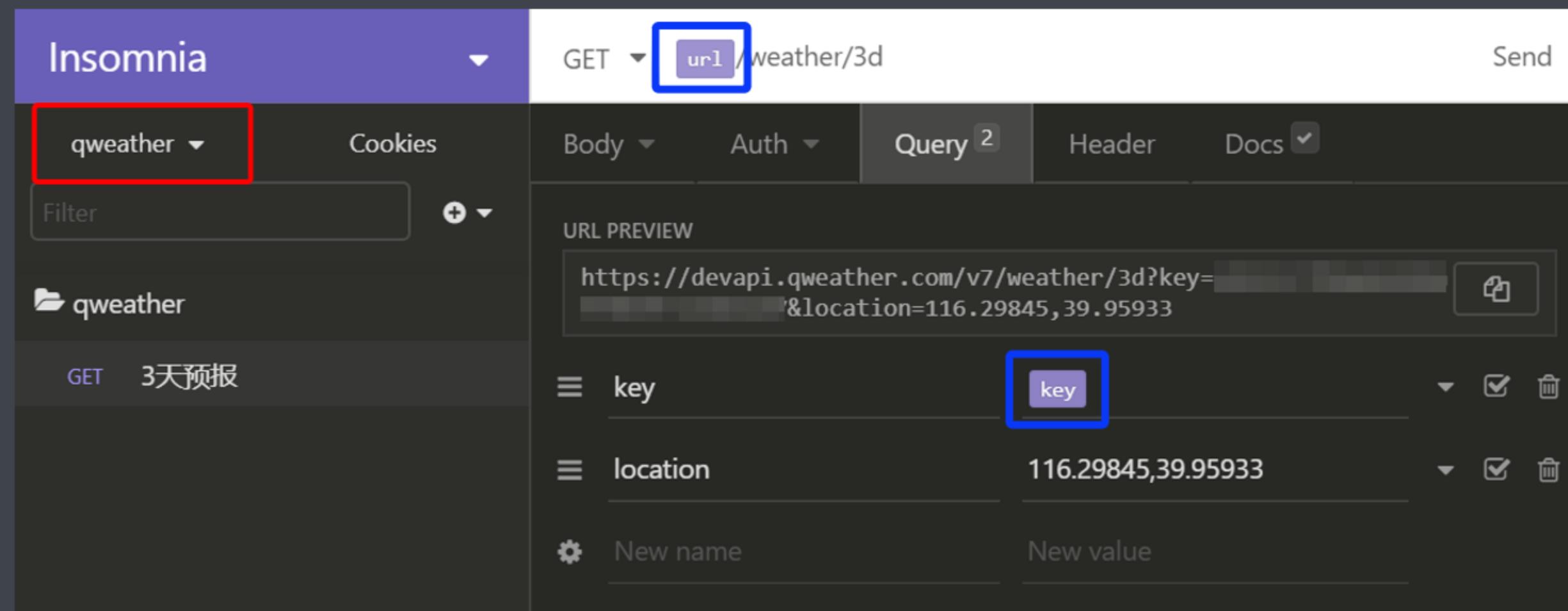
```
{  
    "url": "https://devapi.qweather.com/v7"  
    "key": "xxxx1234xxxx"  
}
```

url /weather/3d?key= key &location=116.29,39.95
url /air/now?key= key &location=104.29,36.21
url /indices/1d?key= key &location=98.56,34.67

Insomnia 配置环境变量



Insomnia 使用环境变量



insomnia

- 下载（Insomnia Core）
 - <https://insomnia.rest/>
- 配置
 - 接口调用中使用的公共内容，应该提升为变量，例如：key
 - 不同的项目，具有不同的环境变量
- 使用
 - 请求方法 + 请求地址 + 请求参数

Insomnia

qweather ▾

Cookies

Filter

qweather

GET 3天预报

GET 生活指数-当天

GET 空气质量

GET 城市信息搜索

GET 热门城市查询

GET POI 范围搜索

GET POI 信息查询

GET 24小时预报

GET 7天预报

接口列表

GET ▾ url /weather/3d

Send

Body ▾

Auth ▾

Query 2

Header

Docs ✓

URL PREVIEW

https://devapi.qweather.com/v7/weather/3d?key=████████████████&location=116.29845,39.95933

key

key

✓



location

116.29845,39.95933

✓



New name

New value

接口详情

Import from URL

Bulk Edit

200 OK

254 ms

496 B

3 Hours Ago ▾

Preview ▾

Header 7

Cookie

Timeline

```
1 {  
2   "code": "200",  
3   "updateTime": "2020-11-02T12:35+08:00",  
4   "fxLink": "http://hfx.link/2ay1",  
5   "daily": [  
6     {  
7       "fxDate": "2020-11-02",  
8       "sunrise": "06:46",  
9       "sunset": "17:09",  
10      "moonrise": "18:15",  
11      "moonset": "07:59",  
12      "moonPhase": "亏凸月",  
13      "tempMax": "15",  
14      "tempMin": "-1",  
15      "iconDay": "100",  
16      "textDay": "晴",  
17      "iconNight": "150",  
18      "textNight": "晴",  
19      "wind360Day": "315",  
20      "windDirDay": "西北风",  
21      "windScaleDay": "3-4",  
22      "windSpeedDay": "24",  
23      "wind360Night": "315",  
24      "windDirNight": "西北风",  
25      "windScal...  
26      "windSpeedNight": "5",  
27      "humidity": "20",  
28      "precip": "0.0",  
29      "pressure": "1019",  
30    }  
31  ]  
32}  
33  
34 $.store.books[*].author
```

接口返回数据

数据接口

使用接口

使用接口



```
fetch(url, {  
    method: 'GET' // Get 请求  
}).then(function(response) {  
    // 获取数据, 数据处理  
}).catch(function(err) {  
    // 错误处理  
});
```



```
let param = {user:'xxx',phone:'xxxxxx'};  
fetch(url, {  
    method: 'post', // Post 请求  
    body: JSON.stringify(param)  
}).then(function(response) {  
    // 获取数据, 数据处理  
});
```

<https://devapi.qweather.com/v7/weather/3d?key=xxxx1234xxxx&location=116.29,39.95>

项目实践

UI 界面

UI 界面

STEPS

1. 首页
2. 新闻页
3. 用户页

UI 界面

首页

首页

拍照



扫一扫



付款码



出行



卡包



美国 加利福尼亚（加州） 山景城

紫外线指数

强

洗车指数

较适宜

钓鱼指数

较适宜

2020-12-02



晴 15°C



5°C 晴间多云

2020-12-03



晴 16°C



5°C 晴

2020-12-04



晴 17°C



4°C 晴

©2021 版权所有 lagou.com



Home



News



RNStack



Profile

react-native-linear-gradient

- 安装
 - `yarn add react-native-linear-gradient`
- 配置
 - <https://github.com/react-native-linear-gradient/react-native-linear-gradient>
- 使用
 - `import LinearGradient from 'react-native-linear-gradient'`
 - `<LinearGradient start={{x: 0, y: 0}} end={{x: 1, y: 0}} colors={['#ddd', '#333']}> ... </LinearGradient>`

UI 界面

新闻页

今日新闻

[头条](#) [社会](#) [国内](#) [国际](#) [娱乐](#) [体育](#)

就在3天前，嘎子哥终究还是“翻车”了，潘长江彻底死心！



2020-12-02 19:54 雷Sir娱乐圈



吴刚走红毯秀恩爱，和发妻相识34年是珍珠婚，俩人却常忽视儿子



2020-12-02 19:47 会火



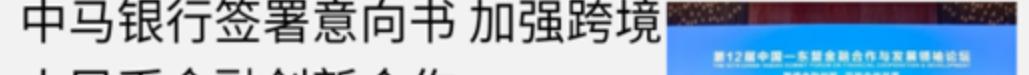
每到冬季，我家必做这碗汤，5块钱煮1锅，温暖滋润，清淡养胃！



2020-12-02 19:45 小面姨美食日记



中马银行签署意向书 加强跨境



Home



News



RNStack



Profile

包含三张图片的项目布局

新闻标题

图片1

图片2

图片3

发表时间 发表机构

其他

包含一张图片的项目布局

新闻标题

发表时间 发表机构

其他

图片

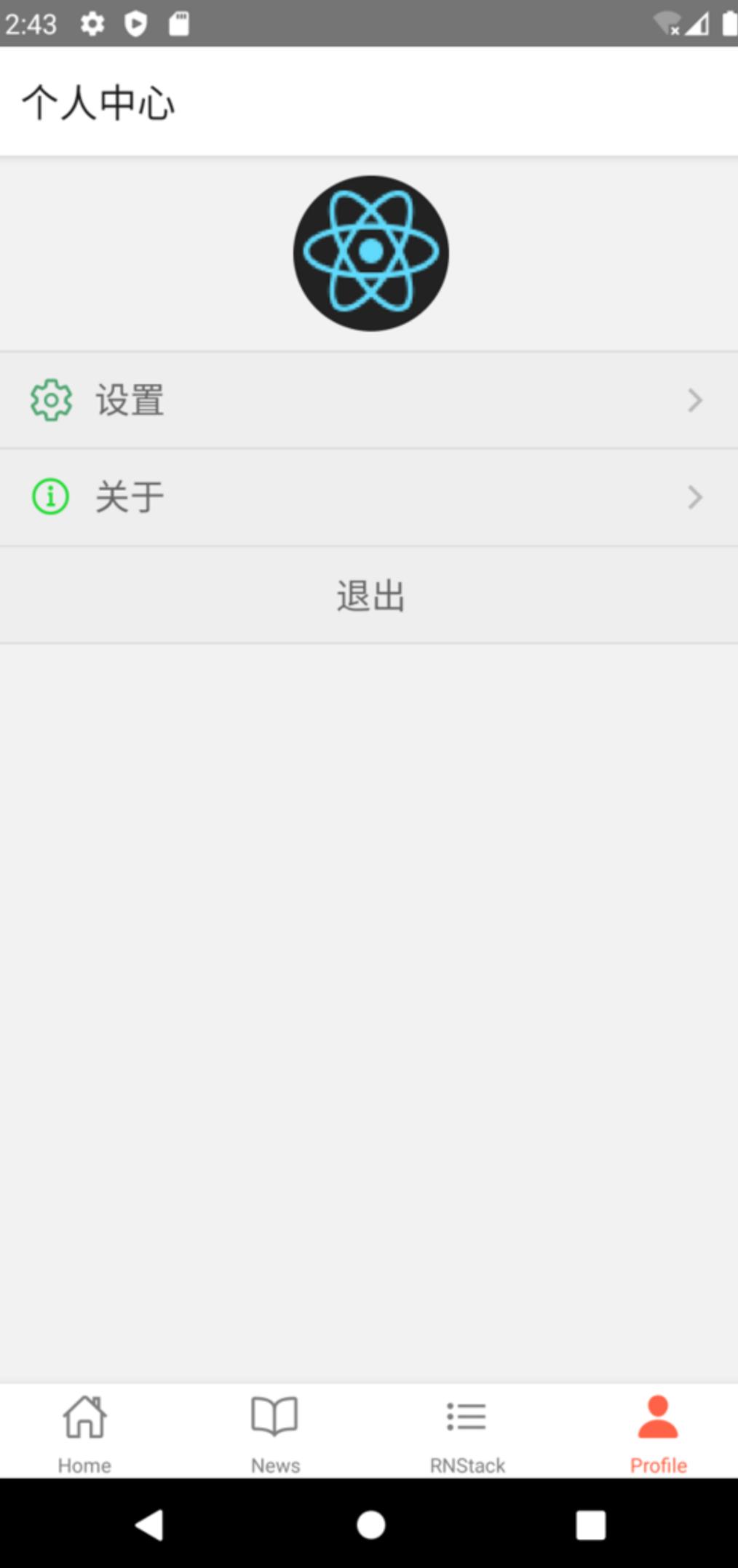
UI 界面

用户页

用户页

STEPS

1. 个人中心
2. 登录页
3. 注册页



UI 界面

用户页



react-native-animated

- 安装
 - `yarn add react-native-animated`
- 使用
 - `import * as Animatable from 'react-native-animated' ;`
 - `<Animatable.View animation="fadeInUpBig">`

react-native-paper

- 安装
- 配置
- 使用

项目实践

状态管理

UI 界面

STEPS

1. Redux
2. 路由鉴权

项目实践

项目优化

项目优化

- 添加项目 logo
- 修改项目名称

拉勾教育

—互联网人实战大学—



下载「拉勾教育App」
获取更多内容