lesson1 项目结构与初始配置

- lesson1 项目结构与初始配置
  - 课堂目标
  - 资源
  - 知识要点
    - 移动应用开发
    - 什么是React Native
    - 谁在用RN
    - 为什么使用React Native
    - 大环境配置
      - 安装node
    - 选择cli
      - expo cli
      - React Native cli
        - 安装HomeBrew
        - 安装watchman
      - ios环境配置（Mac）
        - 安装Xcode
          - Command Line Tools
          - 安装模拟器
        - 安装CocoaPods
      - android环境配置(mac)
        - 安装jdk
        - 安装Android Studio
      - 创建项目
      - 启动ios端
      - 启动android端
    - 开始coding
      - 掌握了react的你，学习rn你需要注意的事项
      - 核心组件与原生组件

原生组件
核心组件
建立常见开发目录
UI库react native elements
常见组件
View
SafeAreaView
Image
ImageBackground
Button
TextInput
Text
容器
关于继承
行溢出
滚动视图
使用长列表
导航管理
导航方式
Stack navigation
基本使用
Link跳转
状态管理
回顾
作业
下节

课堂目标

1. 掌握什么是react-native
2. 掌握如何初始化一个react-native项目
3. 掌握react-native项目结构与初始配置

资源

1. 课堂代码地址(不要忘记切分支)
2. react native知识图谱
3. react-native
4. react-native中文
5. expo
6. ignite
7. react native elements
8. React Navigation
9. WebView
10. Flex 青蛙
11. flex defense

知识要点

完全没有接触过移动应用的扣0，开发过移动应用的扣1，对接过移动应用的扣2。

移动应用开发

什么是React Native

用react开发ios和android应用的js库。

React Native 将原生开发的最佳部分与 React 相结合， 致力于成为构建用户界面的顶尖 JavaScript 框架。

谁在用RN

手机百度、美团、京东、小米、shopee等

为什么使用React Native

酌量添加，多少随意。随时都可以把 React Native 无缝集成到你已有的 Android 或 iOS 项目，当然也可以完全从头焕然一新地重写。

流水的多平台，铁打的 React。绝大多数情况下，使用 React Native 的团队可以在多个平台间共享一份基础代码，以及通用的技术 —— React。

简单易开发，人人有功练。React Native 使你可以创建真正原生的应用，用户体验绝不拉胯。它提供了一些平台无关的抽象核心组件，像是`View`, `Text` 以及 `Image`等，可直接映射渲染为 对应平台的原生UI组件。

无缝跨平台。通过 React 的声明式组件机制和 JavaScript 代码，现有的原生代码和api可以完美地封装嵌合到 React 组件中。这样既为更多新的开发团队赋予原生应用的开发能力，也能极大地提升现有原生团队的开发效率。

保存即刷新。 借助 JavaScript 的动态特性， React Native 能够让你光速迭代。不要再傻等编译了，改、存、刷新！

大环境配置

文档：https://reactnative.cn/docs/environment-setup

开始一个rn项目前，可以使用expo cli或者rn cli，不过在这之前我们都需要先配置下环境。

安装node

如果你node装过了，这一步跳过。注意检查下版本号，node最好需要12或者12+，你可以使用`node -v`先检查下版本号。如果低于这个版本，建议升级或者重装。

```
brew install node
```

Yarn是 Facebook 提供的替代 npm 的工具，可以加速 node 模块的下载。

```
npm install -g yarn
```

安装完 yarn 之后就可以用 yarn 代替 npm 了，例如用`yarn`代替`npm install`命令，用`yarn add 某第三方库名`代替`npm install 某第三方库名`。

选择cli

如果你是个新手，最简单的方式就是用expo cli。expo是个rn的构建工具，它可以让我们快速开始一个rn项目，你可以在web端查看运行效果，也可以用真机或者模拟器查看。

如果你想用react native cli开启一个项目，则需要按照后面的步骤继续安装。

当然，因为网络原因，相较于expo cli，国内使用更多的还是react native cli。

expo cli

因为网络原因，这个小结可以跳过~

Expo是一套沙盒开发环境，还带有一个已上架的空应用容器。这样你可以在没有原生开发平台（Xcode 或是 Android Studio）的情况下直接编写 React Native 应用（当然这样你只能写 js 部分代码而没法写原生代码）。

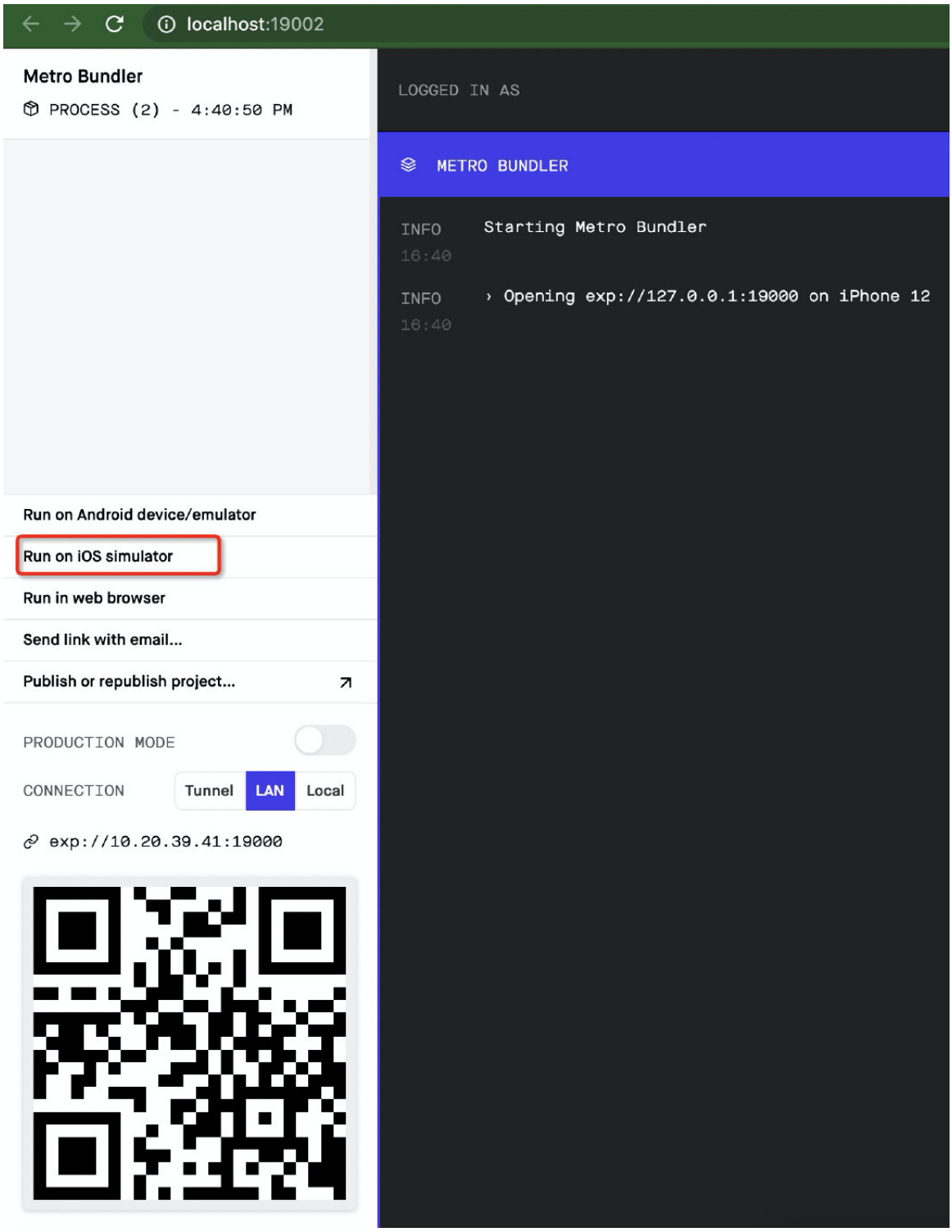
首先安装expo cli工具：

```
yarn global add expo-cli
```

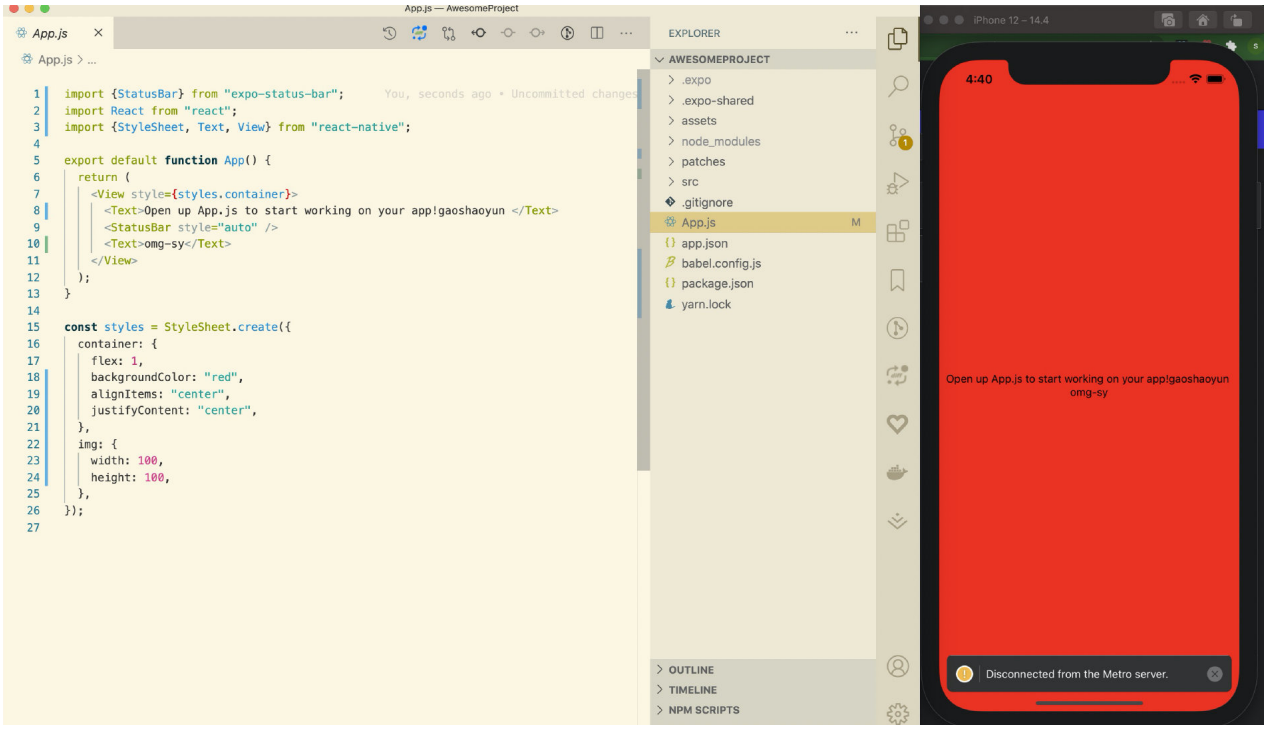
接下来就可以创建一个项目了：

```
expo init AwesomeProject
cd AwesomeProject
yarn start # 也可以用 expo start
```

启动之后，看到如下页面，可以选择打开ios模拟器：



这个时候可以再用vscode打开代码，vscode和模拟器截图如下：



React Native cli

~~*下面有些包也许你电脑上已经装过了，那直接跳过就行。当然最好记住你跳过了哪些包。因为如果有些配置你是很久以前装的，那么后续运行如果出问题了，则可以先检查是不是有些包太老了，先重装下试试，也许就能解决你的问题。*~~

安装[HomeBrew]

（粘贴下面代码带终端运行）

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
```

这一步你很可能就卡住了，那换下面镜像安装试试：

```
/bin/bash -c "$(curl -fsSL https://cdn.jsdelivr.net/gh/ineo6/homebrew-install/install.sh)"
```

安装watchman

```
brew install watchman
```

watchman是Facebook的一个开源项目，它可以用来监视文件并记录文件的变更，当文件变更的时候它可以触发一些操作，比如执行一些命令等。

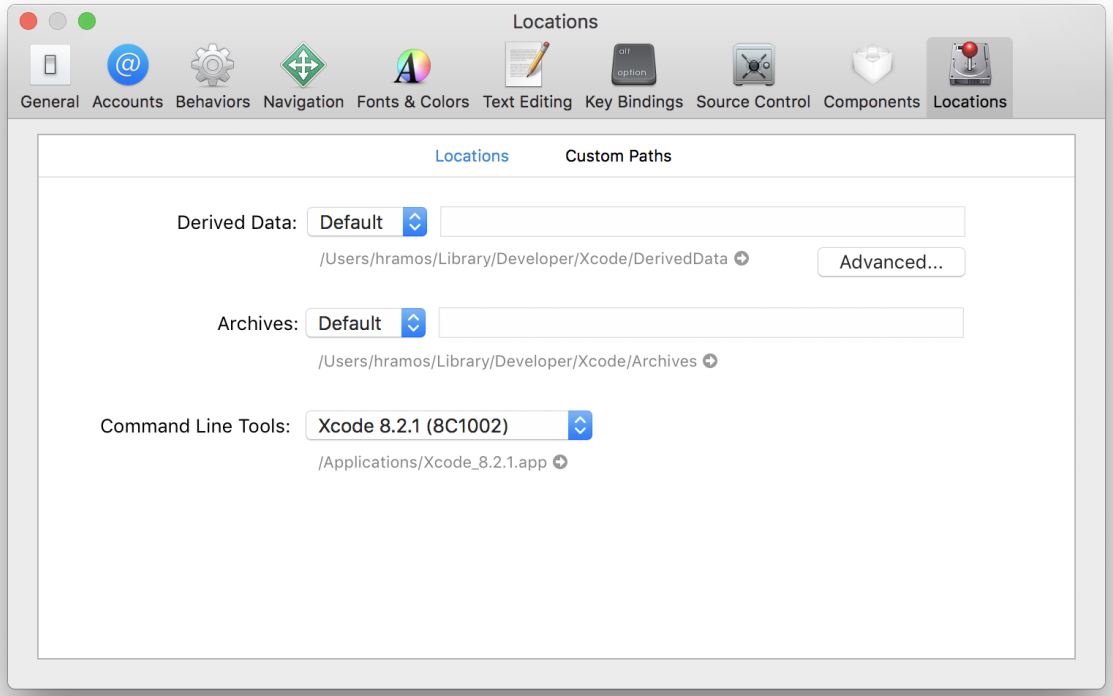
ios环境配置（Mac）

安装Xcode

mac上直接去App store安装即可，注意安装Xcode10或者10+。安装Xcode的同时也可以装上ios模拟器以及其他相关的工具。

Command Line Tools

打开Xcode，选择Preferences，如下图，选择Locations，点击最下面的Command Line Tools，选择最新的Xcode就可以了。



安装模拟器

和上图一样，选择**Components**，然后安装你选择的模拟器即可。

可以用`open -a simulator`启动ios虚拟机查看虚拟机安装是否成功，开发时候不需要这个命令，直接使用XCode启动就行。

安装CocoaPods

```
brew install cocoapods
```

或者

```
sudo gem install cocoapods
```

CocoaPods是用Ruby编写的包管理器，它可以帮助优雅地帮助我们扩展项目。

android环境配置(mac)

安装jdk

```
brew install adoptopenjdk/openjdk/adoptopenjdk8
```

安装Android Studio

首先下载和安装 Android Studio，国内用户可能无法打开官方链接，请自行使用搜索引擎搜索可用的下载链接。安装界面中选择"Custom"选项，确保选中了以下几项：

- `Android SDK`
- `Android SDK Platform`
- `Android Virtual Device`

具体参考https://reactnative.cn/docs/environment-setup

创建项目

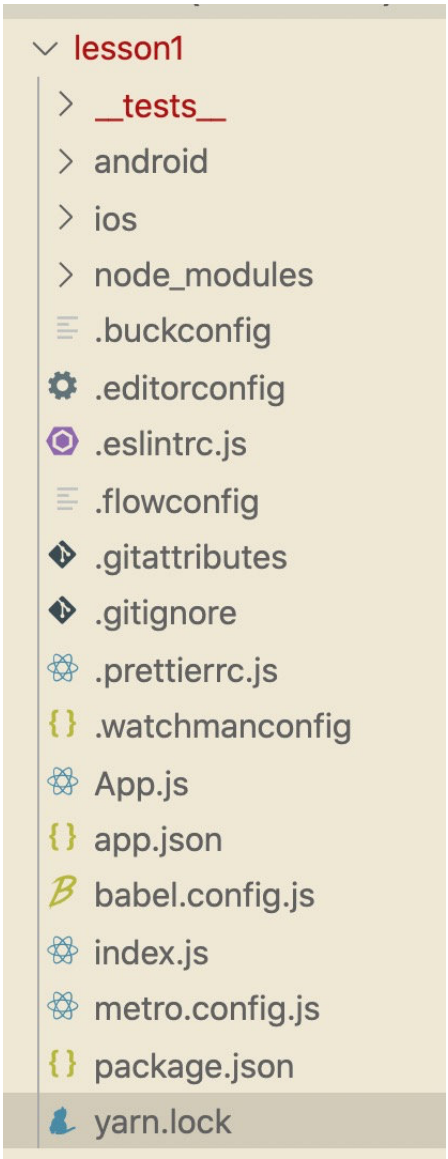
如果你之前全局安装过旧的`react-native-cli`命令行工具，请使用`npm uninstall -g react-native-cli`卸载掉它以避免一些冲突。（我以前就装过，所以我先卸载了~）

```
npx react-native init lesson1
```

> **注意一**：请`不要`在目录、文件名中使用中文、空格等特殊符号。请`不要`单独使用常见的关键字作为项目名（如 class, native, new, package 等等）。请`不要`使用与核心模块同名的项目名（如 react, react-native 等）。

> **注意二**：0.60 及以上版本的原生依赖是通过 CocoaPods 集成安装的。CocoaPods 的仓库在国内也很难访问。如果在 CocoaPods 的依赖安装步骤卡很久（命令行停在 Installing CocoaPods dependencies），请务必使用稳定的代理软件并确定其配置对命令行有效。

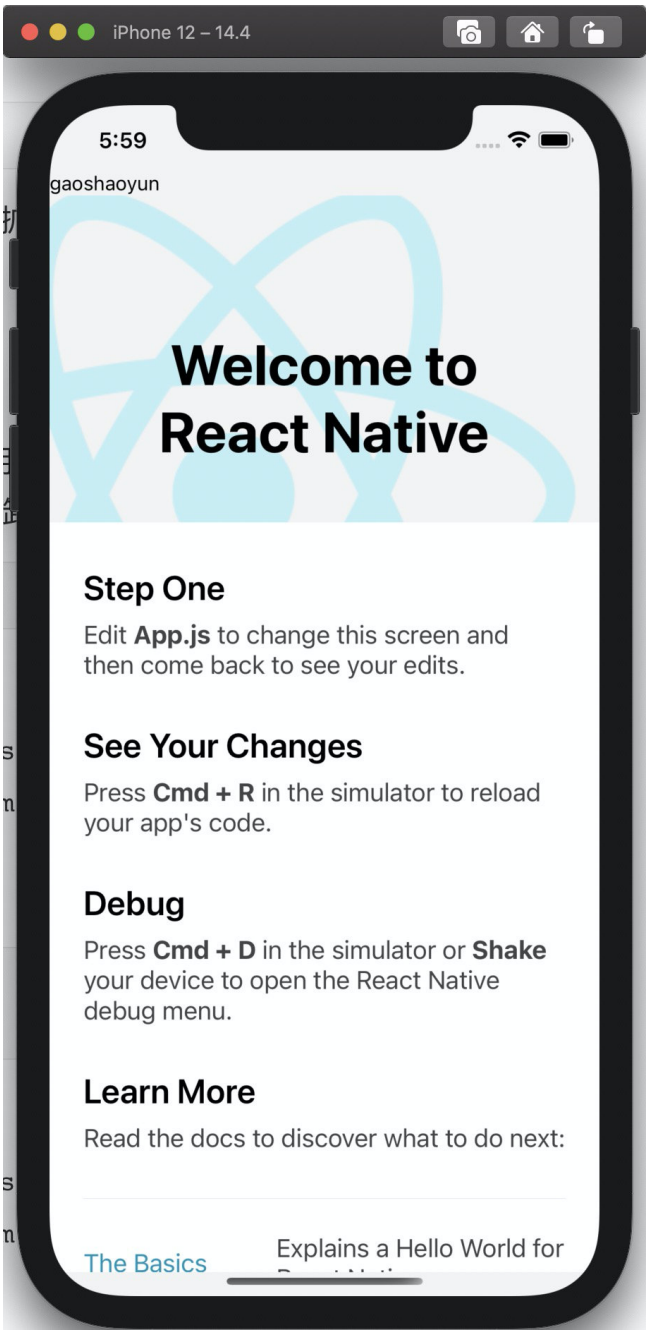
初始项目结构如下：



启动ios端

```
cd lesson1
npx react-native run-ios 或者 yarn ios
```

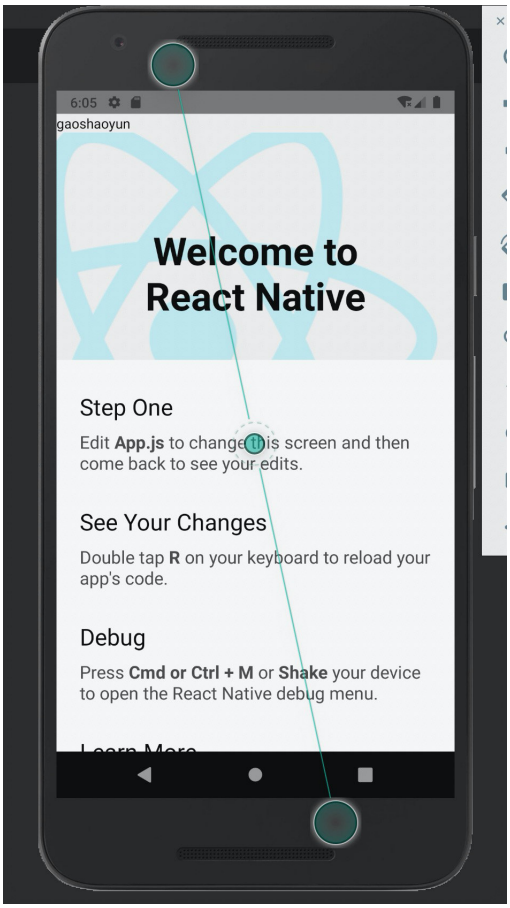
修改代码之后，效果如下：



启动android端

```
cd lesson1
npx react-native run-android 或者 yarn android
```

修改代码之后，效果如下：

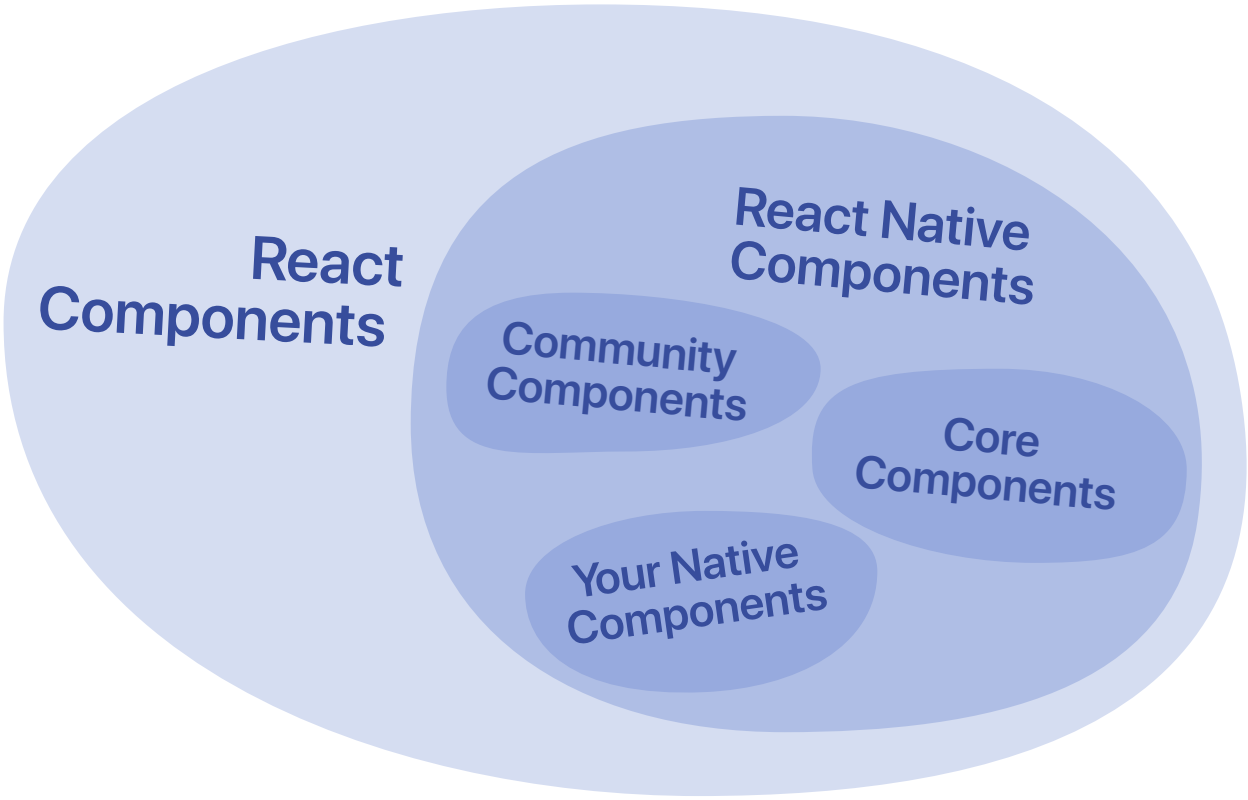


开始coding

掌握了react的你，学习rn你需要注意的事项

- Hooks API 是 React Native 0.59 提供的新特性。
- 因为 JSX 语法糖的实质是调用`React.createElement`方法，所以你必须在文件头部引用`import React from 'react'`。

核心组件与原生组件



原生组件

在 Android 开发中是使用 Kotlin 或 Java 来编写视图；在 iOS 开发中是使用 Swift 或 Objective-C 来编写视图。在 React Native 中，则使用 React 组件通过 JavaScript 来调用这些视图。在运行时，React Native 为这些组件创建相应的 Android 和 iOS 视图。由于 React Native 组件就是对原生视图的封装，因此使用 React Native 编写的应用外观、感觉和性能与其他任何原生应用一样。我们将这些平台支持的组件称为**原生组件**。

React Native 还包括一组基本的，随时可用的原生组件，您可以使用它们来构建您的应用程序。这些是 React Native 的**核心组件**。

核心组件

React Native 具有许多核心组件，从表单控件到活动指示器，应有尽有。你可以在API 章节找到它们。

REACT NATIVE UI COMPONENT	ANDROID VIEW	IOS VIEW	WEB ANALOG	说明
`<View>`	`<ViewGroup>`	`<UIView>`	A non-scrollling `<div>`	A container that supports layout with flexbox, style, some touch handling, and accessibility controls
`<Text>`	`<TextView>`	`<UITextView>`	`<p>`	Displays, styles, and nests strings of text and even handles touch events
`<Image>`	`<ImageView>`	`<UIImageView>`	`<img>`	Displays different types of images
`<ScrollView>`	`<ScrollView>`	`<UIScrollView>`	`<div>`	A generic scrolling container that can contain multiple components and views
`<TextInput>`	`<EditText>`	`<UITextField>`	`<input type="text">`	Allows the user to enter text

建立常见开发目录

1. 在跟目录下建立src，在src下建立components、routes、screens、utils等目录。
2. 把App.js移入routes下。
3. 让@代指src，在src下建立package.json：

```
{
  "name": "@"
}
```

UI库react native elements

安装：

```
yarn add react-native-elements react-native-vector-icons react-native-safe-area-context
```

因为react-native-elements中使用到了`react-native-vector-icons`和`react-native-safe-area-context`，在`react-native init`初始化的项目里，这三者需要都安装。

常见组件

View

创建 UI 时最基础的组件，是一个支持 Flexbox 布局、样式、触摸响应、和一些无障碍功能的容器。

SafeAreaView

`react-native-safe-area-context`是处理安全区域的库。提供SafeAreaView，虽然这个组件在rn中也有，但是rn中的只支持ios10+，不支持ios和android的老版本。项目中建议选择react-native-safe-area-contex的SafeAreaView。

Image

```
const logo = 'https://zh-hans.reactjs.org/logo-og.png';
```

```
<Image source={{uri: logo}} style={styles.logo} />
```

ImageBackground

```
<ImageBackground source={{uri: logo}} style=
{styles.logo}>
  <Text style={{color: 'white'}}>Inside</Text>
</ImageBackground>
```

Button

样式固定，所以如果你想修改样式，需要使用`TouchableOpacity`或是`TouchableNativeFeedback`。或者自己定制、去github使用别人的定制组件。

TextInput

TextInput 是一个允许用户在应用中通过键盘输入文本的基本组件。本组件的属性提供了多种特性的配置，譬如自动完成、自动大小写、占位文字，以及多种不同的键盘类型（如纯数字键盘）等等。

Text

一个用于显示文本的 React 组件，并且它也支持嵌套、样式，以及触摸处理。

容器

`<Text>`元素在布局上不同于其它组件：在 Text 内部的元素不再使用 flexbox 布局，而是采用文本布局。这意味着`<Text>`内部的元素不再是一个个矩形，而可能会在行末进行折叠。见下面的例子。

关于继承

在 React Native 中：**必须把你的文本节点放在`<Text>`组件内**。也不能直接设置一整颗子树的默认样式。使用一个一致的文本和尺寸的推荐方式是创建一个包含相关样式的组件`MyAppText`，然后在你的 App 中反复使用它。你还可以创建更多特殊的组件譬如`MyAppHeaderText`来表达不同样式的文本。

```
<View>
  <MyAppText>
    这个组件包含了一个默认的字体样式，用于整个应用的文本
  </MyAppText>
  <MyAppHeaderText>这个组件包含了用于标题的样
式</MyAppHeaderText>
</View>
```

React Native 实际上还是有一部分样式继承的实现，不过仅限于文本标签的子树。在下面的代码里，第二部分会在加粗的同时又显示为红色：

```
<Text style={{fontWeight: 'bold'}}>
  I am bold <Text style={{color: 'red'}}>and red</Text>
</Text>
```

行溢出

```
<Text numberOfLines={1}>{'书名'.repeat(100)}</Text>
```

但是注意，下面的嵌套Text不支持溢出隐藏，应该避开下面这种写法：

```
<Text>
  <Text  numberOfLines={1}>{'sometext'.repeat(1000)}
</Text>
</Tex>
```

示例：

```
import React, {useState} from 'react';
import {
  View,
  Text,
  StyleSheet,
  SafeAreaView,
  Image,
  ImageBackground,
  TextInput,
  ScrollView,
} from 'react-native';
import {Button} from 'react-native-elements';

const logo = 'https://zh-hans.reactjs.org/logo-og.png';

export default function App() {
  const [count, setCount] = useState(0);
  const [text, setText] = useState('default');
  return (
    <SafeAreaView>
      <ScrollView>
        <Text>App</Text>

        <Image source={{uri: logo}} style={styles.logo}
/>
        <ImageBackground source={{uri: logo}} style=
{styles.logo}>
          <Text style={{color: 'white'}}>Inside</Text>
        </ImageBackground>

        <Button
          title={count + ''}
          onPress={() => setCount(count + 1)}
          buttonStyle={styles.btn}
```

```
        />
        <TextInput
          value={text}
          onChangeText={txt => setText(txt)}
          style={styles.txt}
        />

        <Text style={[styles.txt, {borderColor:
'green'}]}>
          <Text style={styles.bold}>文本: </Text>
          <Text>{text}</Text>
        </Text>
        <View style={[styles.txt, {borderColor:
'green'}]}>
          <Text style={styles.bold}>文本: </Text>
          <Text>{text}</Text>
        </View>

        {/* //! 溢出 */}
        <Text>omg:</Text>
        <Text numberOfLines={1}>{'书名'.repeat(100)}
</Text>

        {/* <Text>{'书名'.repeat(1000)}</Text> */}
      </ScrollView>
    </SafeAreaView>
  );
}

const styles = StyleSheet.create({
  logo: {
    width: 100,
    height: 100,
    marginHorizontal: 10,
    marginVertical: 10,
```

```
  },

  btn: {backgroundColor: 'red'},
  txt: {
    margin: 10,
    padding: 10,
    borderColor: 'red',
    borderWidth: 1,
    color: 'grey',
  },
  bold: {color: 'orange', fontWeight: 'bold'},
});
```

滚动视图

`ScrollView`是一个通用的可滚动的容器，你可以在其中放入多个组件和视图，而且这些组件并不需要是同类型的。ScrollView 不仅可以垂直滚动，还能水平滚动（通过`horizontal`属性来设置）

`ScrollView`适合用来显示数量不多的滚动元素。放置在`ScrollView`中的所有组件都会被渲染，哪怕有些组件因为内容太长被挤出了屏幕外。如果你需要显示较长的滚动列表，那么应该使用功能差不多但性能更好的`FlatList`组件。

使用长列表

React Native 提供了几个适用于展示长列表数据的组件，一般而言我们会选用FlatList或是SectionList。

和`ScrollView`不同的是，`FlatList`并不立即渲染所有元素，而是优先渲染屏幕上可见的元素。

如果要渲染的是一组需要分组的数据，也许还带有分组标签的，那么`SectionList`将是个不错的选择。

导航管理

在web端，我们可以使用a标签导向不同的页面。当用户点击一个link，对应的URL就被push到浏览器的历史栈中，当用户点击后退按钮，浏览器再把历史栈中的这个URL弹出(pop)。这个时候，当前活跃的页面就是上次访问的页面了。不过React Native并没有浏览器那样的历史栈，所以React Navigation就来做这件事了。

- `react-native` >= 0.63.0

```
yarn add @react-navigation/native react-native-screens
react-native-safe-area-context
```

导航方式

React Navigation包含以下功能来帮助你创建导航器:

- StackNavigator
- TabNavigator
- DrawerNavigator

Stack navigation

```
yarn add @react-navigation/native-stack
```

基本使用

`createNativeStackNavigator`是一个函数，返回一个对象，这个对象里有两个都是React组件的属性：`Screen` and `Navigator`。注意它们的包裹关系：NavigationContainer->Navigator->Screen。

Screen属性：

- `name`：string，标记页面，此值须唯一。
- `component`：要渲染的组件。(同react-router中Route的path以及component)
- `options`: object。默认情况下header显示的是name，也可以通过options的title修改。
- `initialParams`给组件参数

App.js

```
import React from 'react';
import {View, Text} from 'react-native';
import {NavigationContainer} from '@react-
navigation/native';
import RootRouter from './RootRouter';

export default function App() {
  return (
    <NavigationContainer>
      <RootRouter />
    </NavigationContainer>
  );
}
```

RootRouter

```
import React from 'react';
```

```
import {View, Text} from 'react-native';
import {createNativeStackNavigator} from '@react-
navigation/native-stack';
import HomeScreen from '@/screens/HomeScreen';
import UserScreen from '@/screens/UserScreen';

const {Screen, Navigator} =
createNativeStackNavigator();

export default function RootRouter() {
  return (
    //   ! SafeAreaView不能写这里
    // <SafeAreaView>
    <Navigator>
      <Screen name="home" component={HomeScreen} />
      <Screen
        name="user"
        component={UserScreen}
        options={{title: '用户中心'}}
        initialParams={{name: '我是小明'}}
      />
    </Navigator>
    // </SafeAreaView>
  );
}
```

> 注意: Screen的 `component` 是组件。不要传一个内联函数 (e.g. `component={() => <HomeScreen />}`), 不然会导致组件无法复用。

Link跳转

```
<Link to={{screen: 'user', params: {id: 'jane'}}}>
  Go to Jane's profile
</Link>
```

状态管理

redux + react-redux。

回顾

lesson1 项目结构与初始配置

- 课堂目标
- 资源
- 知识要点
  - 移动应用开发
  - 什么是React Native
  - 谁在用RN
  - 为什么使用React Native
  - 大环境配置
    - 安装node
  - 选择cli
    - expo cli
    - React Native cli
      - 安装HomeBrew
      - 安装watchman
    - ios环境配置（Mac）
      - 安装Xcode
        - Command Line Tools
        - 安装模拟器
      - 安装CocoaPods

- android环境配置(mac)
  - 安装jdk
  - 安装Android Studio
- 创建项目
- 启动ios端
- 启动android端

开始coding

- 掌握了react的你，学习rn你需要注意的事项
- 核心组件与原生组件
  - 原生组件
  - 核心组件
- 建立常见开发目录
- UI库react native elements
- 常见组件
  - View
  - SafeAreaView
  - Image
  - ImageBackground
  - Button
  - TextInput
  - Text
    - 容器
    - 关于继承
    - 行溢出
- 滚动视图
- 使用长列表
- 导航管理
  - 导航方式
  - Stack navigation
    - 基本使用
    - Link跳转
- 状态管理

回顾

作业

下节

作业

搭建环境、跑起来项目。尝试写下今天的页面。权限。

下节

react navigation