lesson3 导航与页面布局

课堂目标

1. 掌握RN页面布局
2. 掌握导航嵌套

3. 掌握自定义导航
4. 掌握RN长列表渲染
5. 掌握React Native + React Navigation + Redux + React-Redux + Redux-Thunk / Redux-Saga使用方案

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. React Navigation
6. WebView的安装与配置
7. WebView调试

知识要点

开始

```
npx react-native init lesson3
cd lesson3
yarn ios
yarn android
```

定制自己的Navigator

对应文档：https://reactnavigation.org/docs/custom-navigators#useenavigationbuilder

React Navigation支持用户自定义Navigator

`useNavigationBuilder`

```
import * as React from 'react';
import {StyleSheet, Text, Pressable, View} from 'react-native';
import {
  useNavigationBuilder,
  TabRouter,
  TabActions,
  createNavigatorFactory,
} from '@react-navigation/native';

import {Header, Screen} from '@react-navigation/elements';
import {Button} from 'react-native-elements';

function TabNavigator({
  initialRouteName,
  children,
  screenOptions,
  tabBarStyle,
  contentStyle,
}) {
  const {state, navigation, descriptors, NavigationContent} =
    useNavigationBuilder(TabRouter, {
      children,
      screenOptions,
      initialRouteName,
    });
```

```
  return (
    <NavigationContent>
      <View style={[{flexDirection: 'row', paddingTop:
50}, tabBarStyle]}>
        {state.routes.map((route, index) => (
          <Pressable
            title={descriptors[route.key].options.title
|| route.name}
            key={route.key}
            onPress={() => {
              const event = navigation.emit({
                type: 'tabPress',
                target: route.key,
                canPreventDefault: true,
              });

              if (!event.defaultPrevented) {
                navigation.dispatch({
                  ...TabActions.jumpTo(route.name),
                  target: state.key,
                });
              }
            }}
            style={{
              flex: 1,
              justifyContent: 'space-between',
              alignItems: 'center',
            }}>
            <Text>{descriptors[route.key].options.title
|| route.name}</Text>
          </Pressable>
        ))}
      </View>
      <View style={[{flex: 1}, contentStyle]}>
```

```
        {state.routes.map((route, index) => {
          const descriptor = descriptors[route.key];
          const isFocused = state.index === index;
          return (
            <Screen
              key={route.key}
              focused={isFocused}
              route={descriptor.route}
              navigation={descriptor.navigation}
              header={<Header title=
{descriptor.options.title || route.name} />}
              headerShown=
{descriptor.options.headerShown}
              style={[
                StyleSheet.absoluteFill,
                {display: index === state.index ?
'flex' : 'none'},
              ]}>
              {descriptor.render()}
              {/* <Text>{JSON.stringify(state)}</Text>
*/}
            </Screen>
          );
        })}
      </View>
    </NavigationContent>
  );
}

const createMyNavigator =
createNavigatorFactory(TabNavigator);

export default createMyNavigator;
```

可以在RootRouter中使用：

```
const {Navigator, Screen, Group} = createMyNavigator();
```

导航嵌套

如何"隐藏"MovieScreen的TabBar

文档地址：https://reactnavigation.org/docs/hiding-tabbar-in-screens/

修改导航结构即可。让stack在外层，内层使用tab。

根路由使用RootRouter，如下：

```
import React from 'react';
import {View, Text} from 'react-native';
import Section from '@/components/Section';
import {createNativeStackNavigator} from '@react-
navigation/native-stack';
import {createBottomTabNavigator} from '@react-
navigation/bottom-tabs';
import HomeScreen from '../screens/HomeScreen';
import {useSelector} from 'react-redux';
import LoginScreen from '@/screens/LoginScreen';
import HomeRouterScreen from './HomeRouterScreen';
import MovieScreen from '@/screens/MovieScreen';
import CinemaScreen from '../screens/CinemaScreen';
import VIPScreen from '../screens/VIPScreen';
import createMyNavigator from
'../components/createMyNavigator';

const {Navigator, Screen, Group} =
createNativeStackNavigator();
// const {Navigator, Screen, Group} =
createBottomTabNavigator();
```

```
// const {Navigator, Screen, Group} =
createMyNavigator();

export default function RootRouter() {
  const user = useSelector(({user}) => user);
  const {isLogin} = user;

  return (
    <Navigator
      initialRouteName="home"
      screenOptions={{
        headerStyle: {backgroundColor: 'orange'},
        headerBackTitle: '返回',
        // title: '开课吧',
      }}>
      {isLogin ? (
        <Group>
          <Screen
            name="home1"
            // component={HomeScreen}
            component={HomeRouterScreen}
            options={{headerShown: false, title: '首
页'}}
          />
          <Screen
            name="movie"
            // component={HomeScreen}
            component={MovieScreen}
            options={{title: '电影'}}
          />
          <Screen
            name="cinema"
            // component={HomeScreen}
            component={CinemaScreen}
            options={{title: '影院'}}
```

```
          />
          <Screen
            name="vip"
            // component={HomeScreen}
            component={VIPScreen}
            options={{title: '会员中心'}}
          />
        </Group>
      ) : (
        <Group>
          <Screen name="login" component={LoginScreen}
/>
        </Group>
      )}
    </Navigator>
  );
}
```

Home页的路由HomeRouterScreen，如下：

```
import {createBottomTabNavigator} from '@react-
navigation/bottom-tabs';

const {Navigator, Screen, Group} =
createBottomTabNavigator();

export default function HomeRouterScreen() {
  return (
    <Navigator>
      <Screen name="home" component={HomeScreen}
options={{title: '首页'}} />
      <Screen
        name="user"
        component={UserScreen}
        options={{title: '用户中心'}}
```

```
      />
      <Screen
        name="setting"
        component={SettingScreen}
        options={{title: '设置'}}
      />
    </Navigator>
  );
}
```

异步

如下，实现用户名登录：

```
import React, {useState} from 'react';
import {View, Text, TextInput} from 'react-native';
import Section from '@/components/Section';
import {useDispatch, useSelector} from 'react-redux';
import {Button} from 'react-native-elements';
import {login} from '@/action/user';

export default function LoginScreen() {
  const [text, setText] = useState('');
  const user = useSelector(({user}) => user);
  const dispatch = useDispatch();
  const {isLogin, userInfo} = user;

  return (
    <View>
      <Section>LoginScreen</Section>
      <TextInput
        style={{borderWidth: 1, margin: 10, padding:
10}}
```

```
        value={text}
        onChangeText={txt => setText(txt)}
      />
      <Text>{JSON.stringify(user)}</Text>
      <Text style={{color: 'red'}}>{user.err.msg}
</Text>
      <Button
        title={user.loading ? 'loading' : 'login'}
        buttonStyle={{marginVertical: 20}}
        onPress={() => {
          login(dispatch, {name: text});
          // dispatch({type: 'LOGIN_SUCCESS', payload:
{name: '小米'}});
        }}
      />
    </View>
  );
}
```

函数请求

登录与取消登录：

```
import {
  LOGIN_SUCCESS,
  LOGOUT_SUCCESS,
  REQUEST,
  LOGIN_FAILURE,
  LOGIN_SAGA,
} from '../store/const';
import LoginService from '@/service/login';

// export const login = userInfo => dispatch =>
```

```
//   dispatch({type: LOGIN_SUCCESS, payload:
userInfo});

export const logout = () => ({type: LOGOUT_SUCCESS});

export const getMoreUserInfo = (dispatch, userInfo) =>
{
  LoginService.getMoreUserInfo(userInfo).then(
    res => {
      dispatch({type: LOGIN_SUCCESS, payload: res});
    },
    err => {
      dispatch({type: LOGIN_FAILURE, payload: err});
    },
  );
};

export const loginPromise = (dispatch, userInfo) => {
  return LoginService.login(userInfo).then(
    res => {
      return res;
    },
    err => {
      dispatch({type: LOGIN_FAILURE, payload: err});
    },
  );
};

export const login = userInfo => dispatch => {
  dispatch({type: REQUEST});
  LoginService.login(userInfo).then(
    res => {
      getMoreUserInfo(dispatch, res);
    },
    err => {
```

```
      dispatch({type: LOGIN_FAILURE, payload: err});
    },
  );
};

// export const login = userInfo => ({type: LOGIN_SAGA,
payload: userInfo});
```

结合redux-thunk

同步：

```
export const login = userInfo => dispatch =>
dispatch({type: LOGIN_SUCCESS, payload: userInfo});
```

异步:

```
export const login = userInfo => dispatch => {
  dispatch({type: REQUEST});
  LoginService.login(userInfo).then(
    res => {
      getMoreUserInfo(dispatch, res);
    },
    err => {
      dispatch({type: LOGIN_FAILURE, payload: err});
    },
  );
};
```

结合redux-saga

```
export const login = userInfo => ({type: LOGIN_SAGA,
payload: userInfo});
```

结合saga实现loginSaga

```
// 调用异步操作 call、fork
//  状态更新put dispatch
// 做监听 take takeEvery

import {call, put, takeEvery, take, fork} from 'redux-
saga/effects';
import LoginService from '../service/login';
import {LOGIN_FAILURE, LOGIN_SAGA, LOGIN_SUCCESS,
REQUEST} from '@/store/const';

// 做异步
function* loginHandle(action) {
  yield put({type: REQUEST});
  try {
    const res1 = yield call(LoginService.login,
action.payload);
    const res2 = yield
call(LoginService.getMoreUserInfo, res1);
    yield put({type: LOGIN_SUCCESS, payload: res2});
  } catch (err) {
    yield put({type: LOGIN_FAILURE, payload: err});
  }
}

export function* loginSaga() {
  yield takeEvery(LOGIN_SAGA, loginHandle);
  //   while (true) {
  //     const action = yield take(LOGIN_SAGA);
```

```
  //     yield fork(loginHandle, action);
  //     console.log('action', action); //sy-log
  //   }
}
```

修改store，引入中间件

```
import {applyMiddleware, combineReducers, createStore}
from 'redux';
import {loginReducer} from './loginReducer';
import createSagaMiddleware from 'redux-saga';
import thunk from 'redux-thunk';
import {loginSaga} from '@/action/loginSaga';

const sagaMiddleware = createSagaMiddleware();

const store = createStore(
  combineReducers({user: loginReducer}),
  applyMiddleware(thunk, sagaMiddleware),
);

sagaMiddleware.run(loginSaga);

export default store;
```

WebView

安装

`WebView` 创建一个原生的 WebView，可以用于访问一个网页。yarn之后不要忘记`cd ios && pod install`。

```
yarn add react-native-webview
```

使用

首先创建组件：

```
import React from 'react';
import {WebView} from 'react-native-webview';
import Section from '@/components/Section';

export default function WebScreen({route}) {
  const {uri = 'https://main.m.taobao.com'} =
route.params || {};

  return (
    <WebView
      source={{
        uri,
      }}
      // style={{marginTop: 20}}
      originWhitelist={['*']}
      // source={{html: '<h1>Hello world</h1>'}}
    />
  );
}
```

在RootRouter或者HomeRouterScreen中配置后，访问即可。

```
<Screen
  name="webview"
  component={WebScreen}
  options={{title: 'webview'}}
  initialParams={{uri: 'http://www.baidu.com/'}}
/>
```

长列表

ScrollView

简单粗暴地把所有子元素一次性全部渲染出来。

FlatList

`FlatList`会惰性渲染子元素，只在它们将要出现在屏幕中时开始渲染。这种惰性渲染逻辑要复杂很多，因而 API 在使用上也更为繁琐。除非你要渲染的数据特别少，否则你都应该尽量使用`FlatList`，哪怕它们用起来更麻烦。

SectionList

文档：https://reactnative.cn/docs/sectionlist

高性能的分组(section)列表组件，支持下面这些常用的功能：

- 完全跨平台。
- 行组件显示或隐藏时可配置回调事件。
- 支持单独的头部组件。
- 支持单独的尾部组件。
- 支持自定义行间分隔线。
- 支持分组的头部组件。
- 支持分组的分隔线。
- 支持多种数据源结构
- 支持下拉刷新。
- 支持上拉加载。

如果你的列表不需要分组，那么可以使用结构更简单的FlatList即可。

回顾

作业

复习今天内容，实现功能。

下节课预告

地图、调试。