lesson4 第三方原生组件、原生相关知识

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
8. 高德地图
9. 高德地图详细配置
10. 学习案例，注意版本很老，学习思路就行，叶枫推荐
11. vue vdom diff
12. react vdom diff
13. https://github.com/bubucuo/vdom

课堂目标

1. 掌握高德地图使用
2. 掌握依赖定制（打补丁）
3. 掌握RN周边常用库
4. 掌握RN调试

知识要点

开始

```
npx react-native init lesson4
cd lesson4
yarn ios
yarn android
```

```
yarn add @react-navigation/bottom-tabs @react-navigation/native @react-navigation/native-stack react-native-elements react-native-safe-area-context react-native-screens react-native-vector-icons react-native-webview redux react-redux redux-saga redux-thunk
```

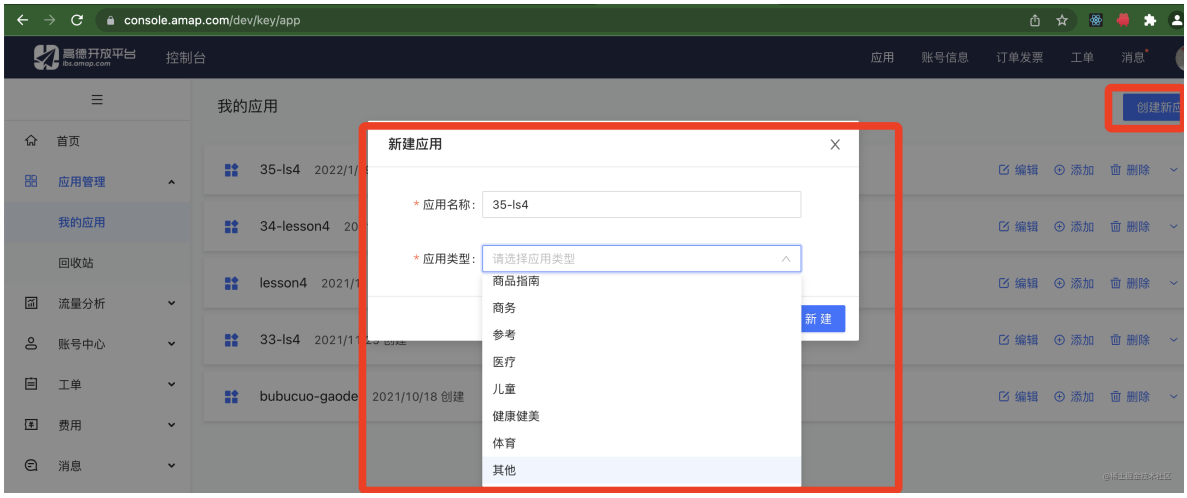
```
cd ios && pod install && cd..
```

高德地图

```
yarn add react-native-amap3d
cd ios && pod install && cd ..
# 配置key完成之后，打完补丁之后
yarn ios
```

配置key

首先创建新应用:

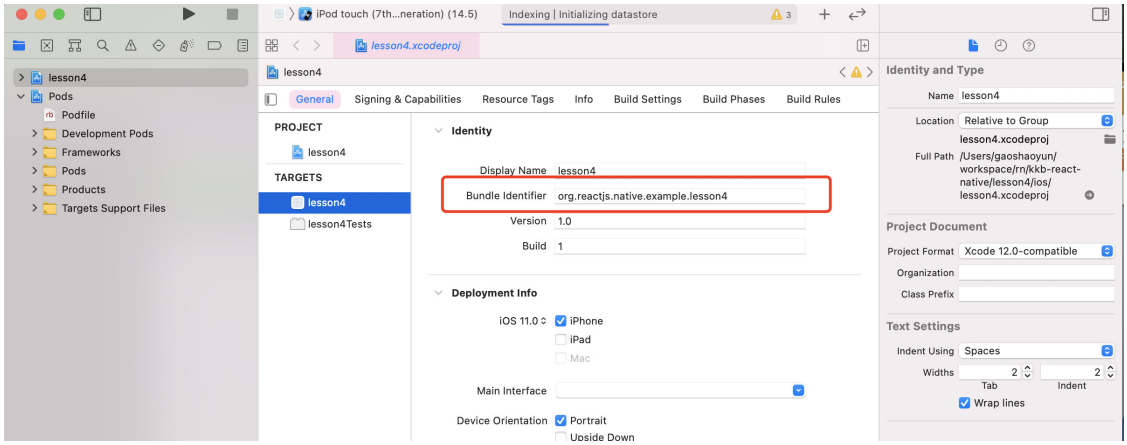


然后需要再配置key，这个ios和android不同，需要分别配置:

iOS配置

1. 获取高德 Key。

获取Bundle ID:



创建“附近”页面

```
// <Screen name="map" component={MapScreen} options={{title: '附近'}} />

import React from 'react';
import {View, Text} from 'react-native';
import Section from '@/components/Section';
import {MapView, MapType} from 'react-native-amap3d';

export default function MapScreen() {
  return (
    <MapView
      mapType={MapType.Satellite}
      initialCameraPosition={{
        target: {
          latitude: 39.91095,
          longitude: 116.37296,
        },
        zoom: 8,
      }}
    />
  );
  //   return (
  //     <MapView
  //       style={{flex: 1}}
  //       center={{
  //         latitude: 39.91095,
  //         longitude: 116.37296,
  //       }}
  //     />
  //   );
}
```

定制依赖（打补丁）

1. 安装

```
yarn add patch-package
```

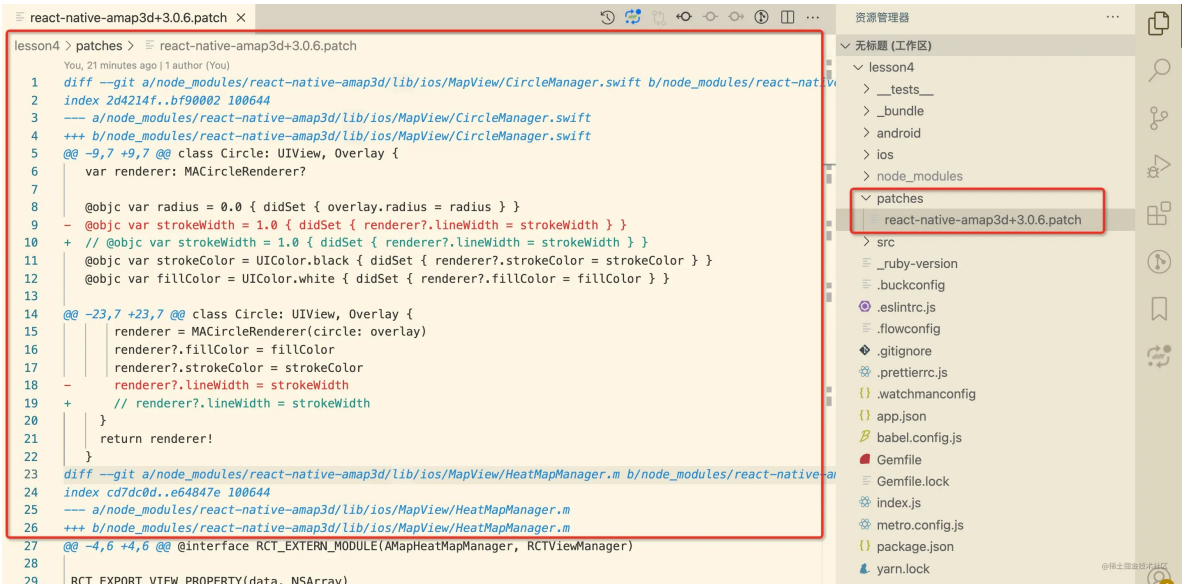
2. 修改package.json，scripts 中加入

```
 "postinstall": "patch-package"
```

3. 修改 node_modules 中的源码
4. 执行

```
yarn patch-package react-native-amap3d
```

，此时会生成一个patches文件。



建立网络请求

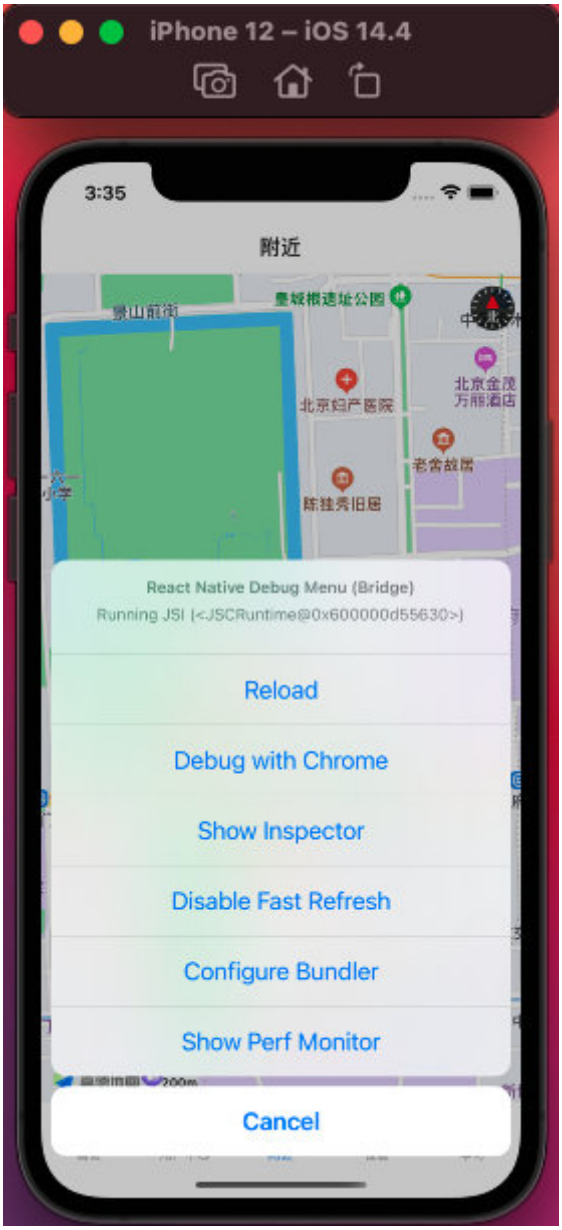
```
export default function UserScreen({navigation}) {
  const user = useSelector(({user}) => user);
  const dispatch = useDispatch();
  const {isLogin, userInfo} = user;

  console.log('UserScreen', isLogin); //sy-log

  const route = useRoute();

  const [random, setRandom] = useState({name: {}});

  const getUser = () =>
    fetch('https://randomuser.me/api')
      .then(x => x.json())
      .then(x => {
        setRandom(x.results[0]);
      });

  useEffect(() => {
    getUser();
    // debugger;
```

```
  }, []);

  return (
    <View>
      <Section>UserScreen</Section>

      <Link to={{screen: 'vip'}}>vip</Link>

      <Text style={{margin: 10}}>id:{userInfo.id}</Text>
      <Text style={{margin: 10}}>name:{userInfo.name}</Text>
      <Text style={{margin: 10}}>score:{userInfo.score}</Text>

      {isLogin ? (
        <Button
          title={userInfo.name + 'logout'}
          buttonStyle={{marginVertical: 20}}
          onPress={() => {
            dispatch(logout());
            // dispatch({type: 'LOGOUT_SUCCESS'});
          }}
        />
      ) : (
        <Button
          title="login"
          buttonStyle={{marginVertical: 20}}
          onPress={() =>
            dispatch({type: 'LOGIN_SUCCESS', payload: {name: '小米'}})
          }
        />
      )}

      <View style={{margin: 20}}>
        <Button
          title="get random"
          onPress={() => {
            setRandom(getUser());
          }}
        />
        <Text> {random.name?.first} </Text>
        {/* <Text>{JSON.stringify(random)}</Text> */}
      </View>
    </View>
  );
}
```

调试

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

开发环境下，我们可以通过摇晃设备或是选择 iOS 模拟器的"Device"菜单中的"Shake"选项来打开开发菜单。另外，如果是在 iOS 模拟器中运行，还可以按下 `Command` + `Control` + `Z` 快捷键，Android 模拟器对应的则是 `Command` ⌘ + `M`（windows 上可能是 F1 或者 F2），或是直接在命令行中运行 `adb shell input keyevent 82` 来发送菜单键命令。



Reload

很多情况下自动刷新并不能顺利实施，如使用导航的时候。如果碰到任何界面刷新上的问题，请尝试手动刷新。具体的操作就是在开发菜单中点击“r”按键。也可以在 iOS 模拟器中按下点击Reload。

Chrome 开发者工具

在开发者菜单中选择"Debug With Chrome"选项，即可以开始在 Chrome 中调试 JavaScript 代码。点击这个选项的同时会自动打开调试页面 http://localhost:8081/debugger-ui.(如果地址栏打开的是 ip 地址，则请自行改为 localhost)。

这里可以打印console或者是断点调试 js 脚本，但是无法监控网络请求，这时候可以选择React Native Debugger或者是Flipper。

React Native Debugger

文档：https://github.com/jhen0409/react-native-debugger

开始文档：https://github.com/jhen0409/react-native-debugger/blob/master/docs/getting-started.md

安装

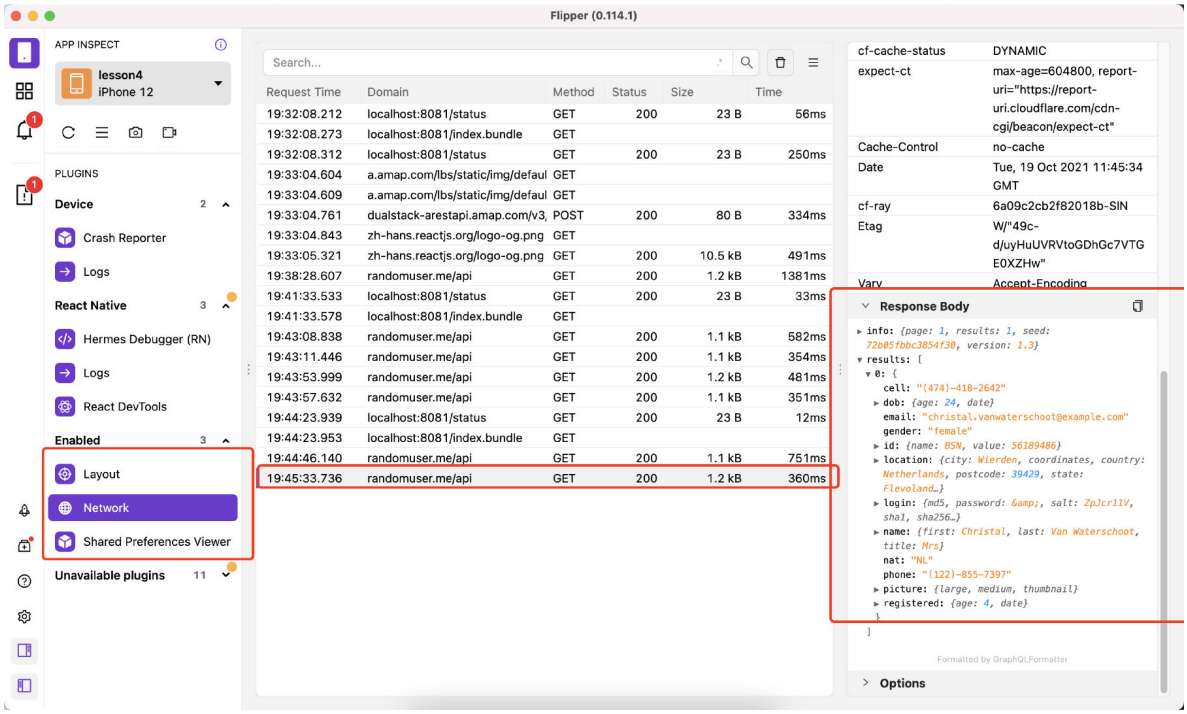
```
brew install --cask react-native-debugger
```

启动：

```
open "rndebugger://set-debugger-loc?host=localhost&port=8081"
```

Flipper

文档：https://fbflipper.com/



React Developer Tools

调试 React 组件层次结构。

> 注意：react-devtools v4 需要 react-native 0.62 或更高版本才能正常工作。

```
npm install -g react-devtools
```

启动：

```
react-devtools
```

WebView调试

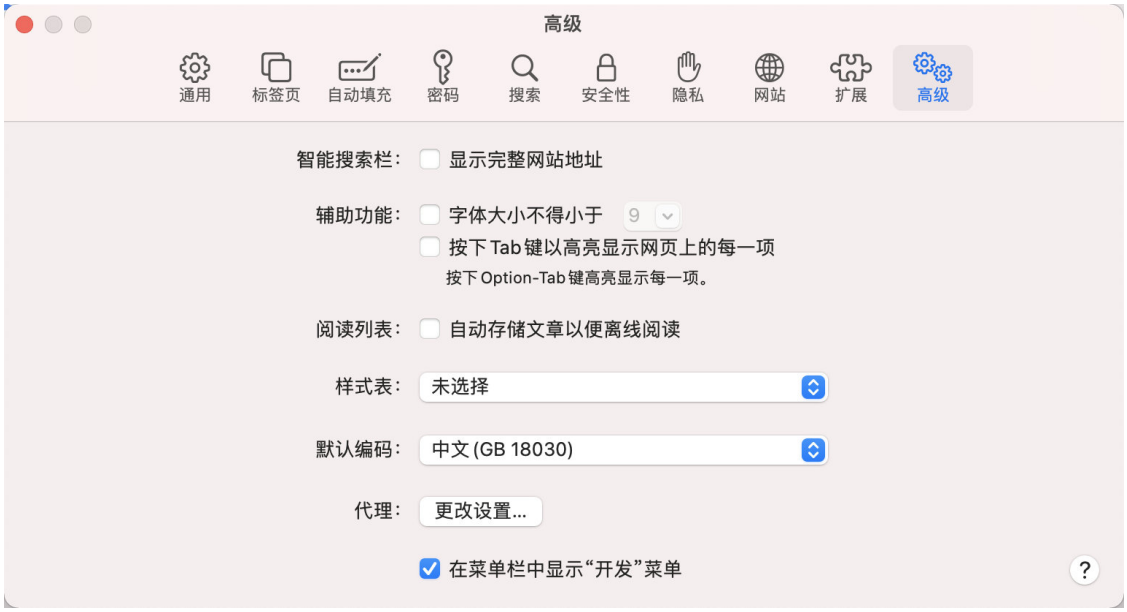
文档：https://github.com/react-native-webview/react-native-webview/blob/master/docs/Debugging.md

iOS & Safari

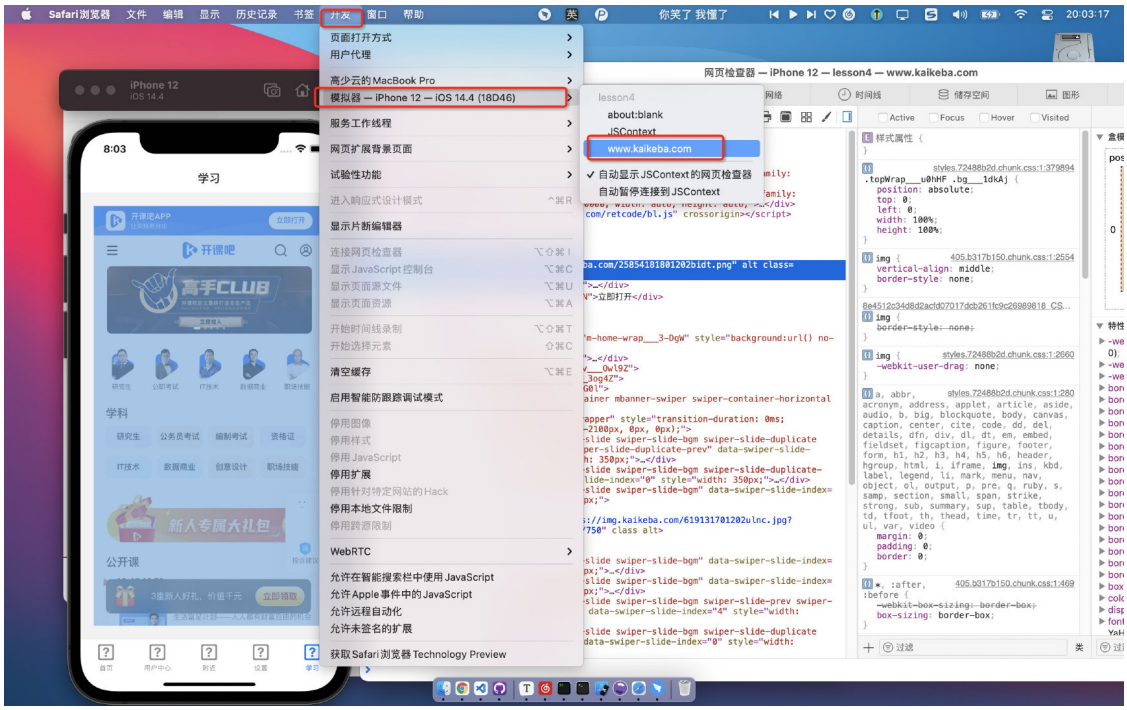
ios模拟器或者设备可以使用Safari开发者来调试WebView内容。

步骤:

1. Open Safari 偏好设置 -> 高级 -> 选中最下面的 '在菜单中显示"开发"菜单'"



2. 启动ios模拟器或者ios设备上的程序
3. 现在你可以像Web上一样的调试WebView内容了



注意:

当你在设备上调试的时候，你必须开启ios设备上的开发者模式:

设置 -> Safari -> 高级 -> 网页检查器。

如果你在开发者菜单上没看到你的设备，并且你启动Safari后再启动的模拟器，那尝试重启下Safari。

Android & Chrome

可以用Chrome开发者模式来调试安卓模拟器或者设备。

1. 把下面的代码加入MainApplication.java。

```
  import android.webkit.WebView;

  @Override
  public void onCreate() {
    super.onCreate();
      ...
    WebView.setWebContentsDebuggingEnabled(true);
  }
```

1. 启动安卓模拟器或者设备上的app的WebView
2. 打开Chrome的 `chrome://inspect/#devices` (参考: Remote debug Android devices)
3. 选择设备上Select your device on the left and select "Inspect" on the WebView contents you'd like to inspect
4. 现在你可以像Web上一样的调试WebView内容了

注意:

在设备上调试的时候，你必须在设备的设置里允许USB调试:

When debugging on device you must enable USB debugging in your device settings:

参考：设置 -> 系统 -> 关于手机 -> 开发者选项 -> 允许USB模式调试

FlatList

```
import React from 'react';
import {View, Text, FlatList, StyleSheet, Image} from 'react-native';
import Section from '@/components/Section';
import {movieOnInfoList} from '@/utils/service';
import {useNavigation} from '@react-navigation/core';
import {Button} from 'react-native-elements';
import {SafeAreaView} from 'react-native-safe-area-context';

export default function CinemaScreen() {
  return (
    <FlatList
      style={styles.list}
      data={movieOnInfoList.movieList}
      renderItem={({item}) => <Movie {...item} />}
      keyExtractor={item => item.id}
    />
  );
}

const styles = StyleSheet.create({
  list: {margin: 14, paddingBottom: 100},
```

```
  box: {
    position: 'relative',
    width: '100%',
    height: 100,
  },
  img: {position: 'absolute', width: 64, height: 90, margin: 4},
  middle: {
    marginLeft: 80,
  },
  nm: {width: 200, marginBottom: 4, fontSize: 16, fontWeight: 'bold'},
  txt: {lineHeight: 20, fontSize: 13, color: '#666'},
  sc: {color: 'orange', fontSize: 18, fontWeight: 'bold'},
  buy: {
    position: 'absolute',
    top: 20,
    right: 20,
    backgroundColor: '#f03d37',
    borderRadius: 8,
  },
});

const Movie = movie => {
  const navigation = useNavigation();
  return (
    <View style={styles.box}>
      <Image
        source={{
          uri: movie.img + '@1l_1e_1c_128w_180h',
        }}
        style={styles.img}
      />
      <View style={styles.middle}>
        <Text numberOfLines={1} style={styles.nm}>
          {movie.nm}
        </Text>
        <Text style={styles.txt}>
          观众评 <Text style={styles.sc}>{movie.sc}</Text>
        </Text>
        <Text style={styles.txt}>主演: {movie.star}</Text>
        <Text style={styles.txt}>{movie.showInfo}</Text>
      </View>
      <View style={styles.buy}>
        <Button
          title="购票"
          accessibilityLabel="购票"
          onPress={() => {
            navigation.navigate('cinemaList', {...movie});
          }}
        />
      </View>
    </View>
  );
};
```

lesson4 第三方原生组件、原生相关知识

资源

课堂目标

作业

搭建一下框架，丰富内容，复习。