一、Flutter介绍

Flutter是谷歌公司开发的一款开源、免费的UI框架，可以让我们快速的在Android和iOS上构建高质量App。它最大的特点就是跨平台、以及高性能。 目前 Flutter 已经支持 iOS、Android、Web、Windows、macOS、Linux等。

Flutter基于谷歌的dart语言，如果没有任何Dart语言的基础，不建议直接学习Flutter。建议先学习Dart语言的基本语法。然后再进入Flutter的学习。

Dart基础教程：https://www.itying.com/goods-1101.html

市面上已经有很多的混合App开发框架了，但是有些混合APP开发框架主要是针对前端开发者的：比如ReactNative（基于React）、Ionic（基于Angular、Vue、React）。有些则是针对.Net平台针对.Net开发者的比如：Xamarin

Flutter是谷歌基于Dart语言开发的一款跨平台的App开发框架。它针对的开发者是全部开发者。它的性能相比RN、Ionic这样的框架要更好一些。

Flutter在2019年的时候就拥有了非常高的关注度。 我们录制的《Flutter仿京东商城项目实战第一版》已经有100多万的学习者了。Flutter目前已经非常稳定，并且社区也非常完善了，应用市场中新发布的新应用有进一半是Flutter开发的应用。Flutter3.x以后不仅支持了Android Ios App的跨平台开发，还支持了Web、Windows、MacOs、Linux桌面应用的跨平台开发。全球很多公司都已经在商业项目中使用Flutter，比如Google、微软、阿里、字节、百度、京东等。Flutter 在 Github Star 数已经有150万了，在跨端框架中排名第一。据官方统计截止到2022年6月，已经有超过50万个商业应用程序是用Flutter建立的 。Flutter是一个非常值得学习的框架，Flutter不仅具有跨平台、高性能等特点，还具有稳定性的特点，从2018年12月5日发布的Flutter1.0到后面的所有版本用法都是一样的。

Flutter 官网：https://flutter.dev/

Flutter Packages官网：https://pub.dev/

二、Windows上面搭建Flutter Android运行环境（适用于Flutter3.7.3之后的版本）

提示：Flutter3.7.x之前的版本配置请参考教程，或者参考文档《Flutter Android开发环境搭建，适用于Flutter3.3.10之前的版本》

Flutter Android环境搭建：

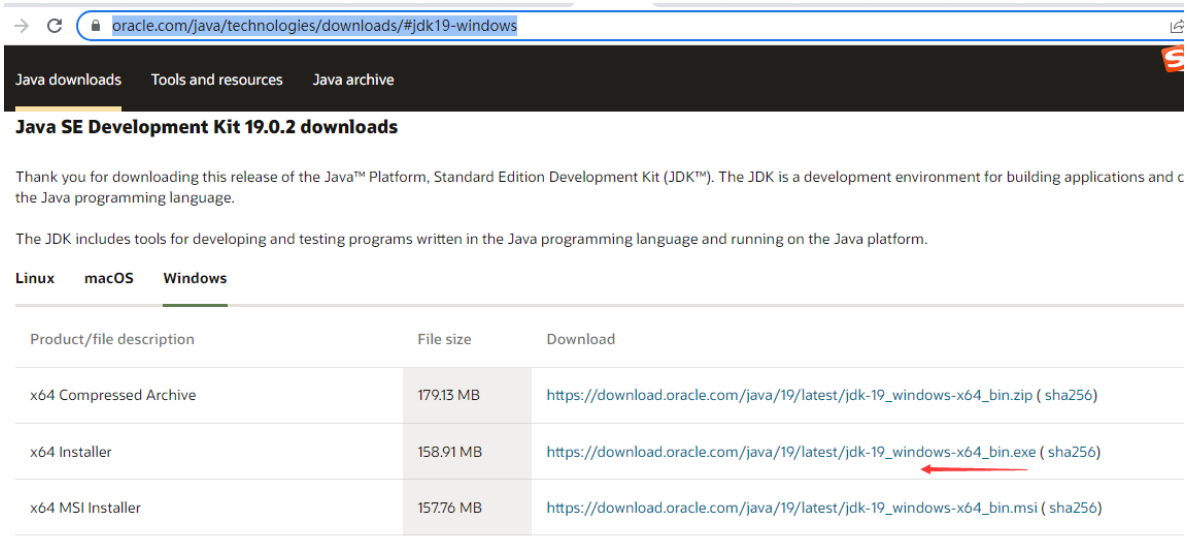
- 电脑上面安装配置JDK （**版本** java version "19.0.2" 2023-01-17）
- 电脑上下载安装Android Studio （**版本** 2022.1.1 Patch 1）

- 电脑上面下载配置Flutter Sdk （**版本** Flutter3.7.3以及Flutter之后的版本）
- 电脑上配置Flutter国内镜像
- 运行 flutter doctor命令检测环境是否配置成功，根据提示配置安装对应软件
- 打开Android Studio 安装Flutter插件
- 创建运行Flutter项目

2.1、 电脑上面安装配置JDK

1、下载安装JDK（19.0.2版本）

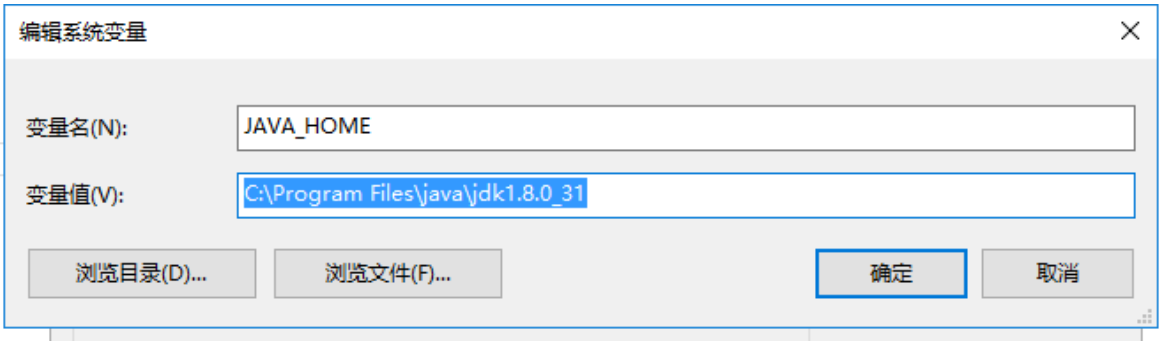
https://www.oracle.com/java/technologies/downloads/#jdk19-windows



2、配置JDK（19.0.2版本）

提示：jdk19安装完成后输入java、javac就有提示信息，但是也需要配置环境变量

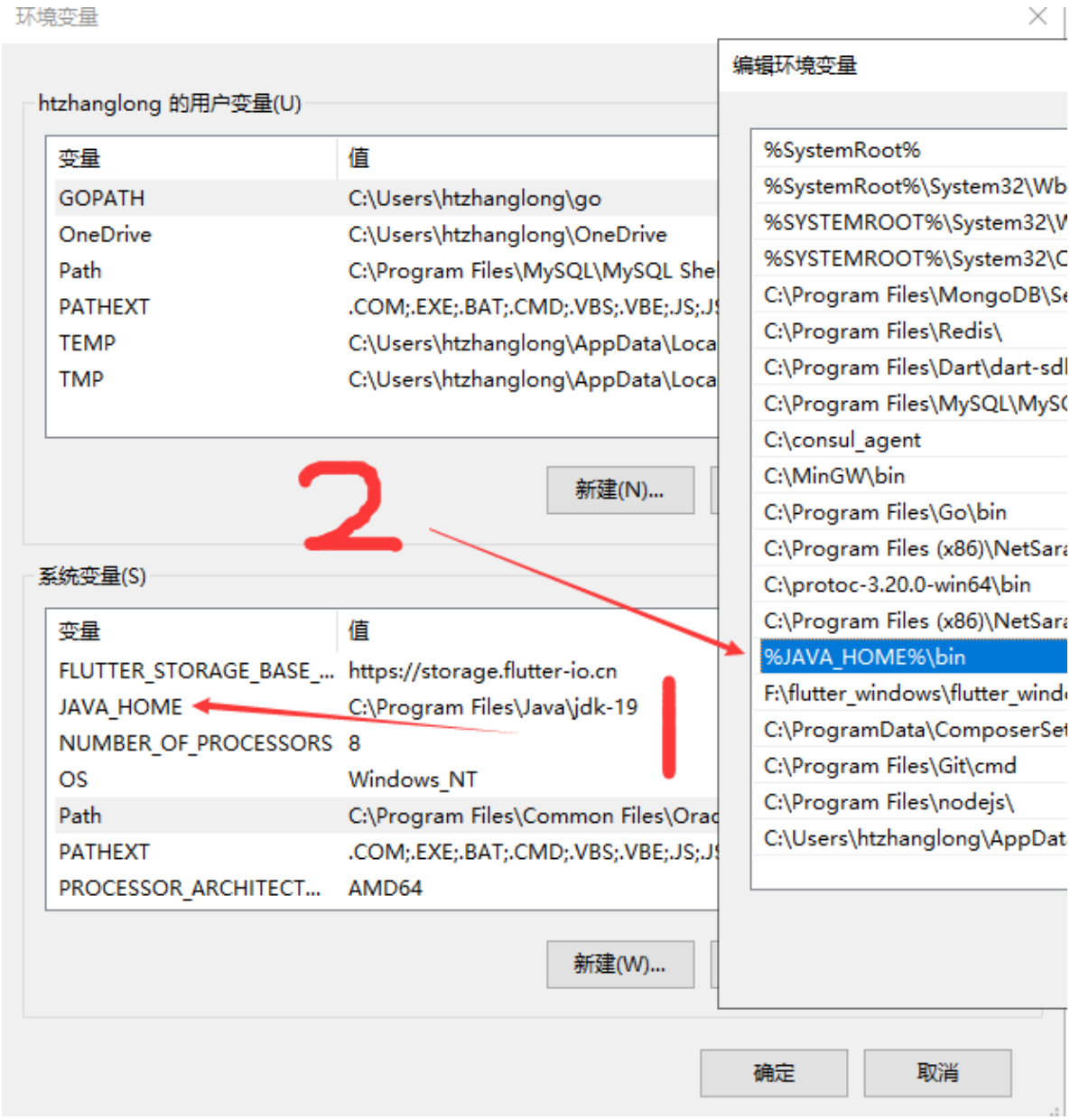
1、**系统变量** 里面新增JAVA_HOME，设置值为java sdk 根目录：



2、**系统变量** 找到Path 在Path环境变量里面增加如下代码（提示jdk19无需配置jre)

```
;%JAVA_HOME%\bin;
```

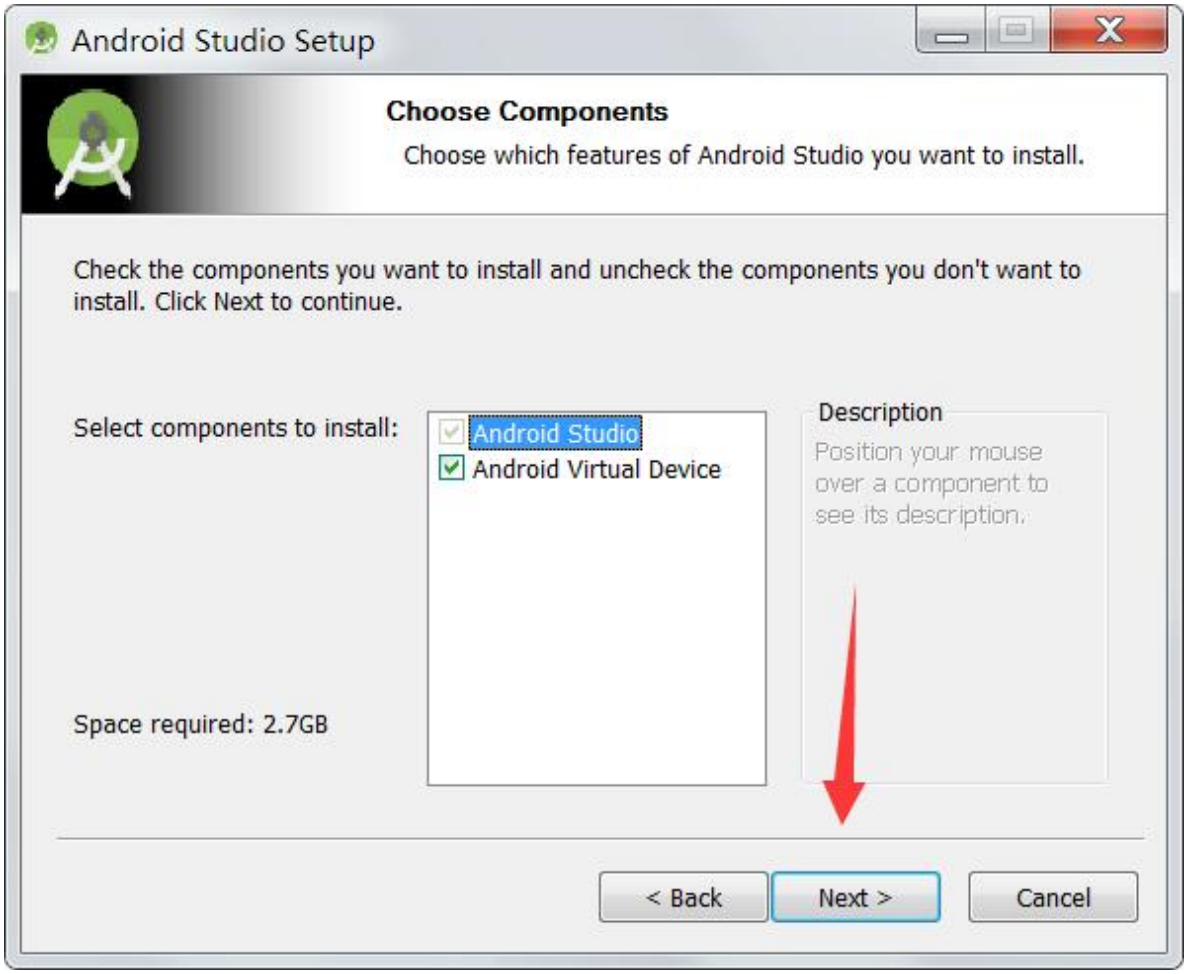
win10、win11中的配置

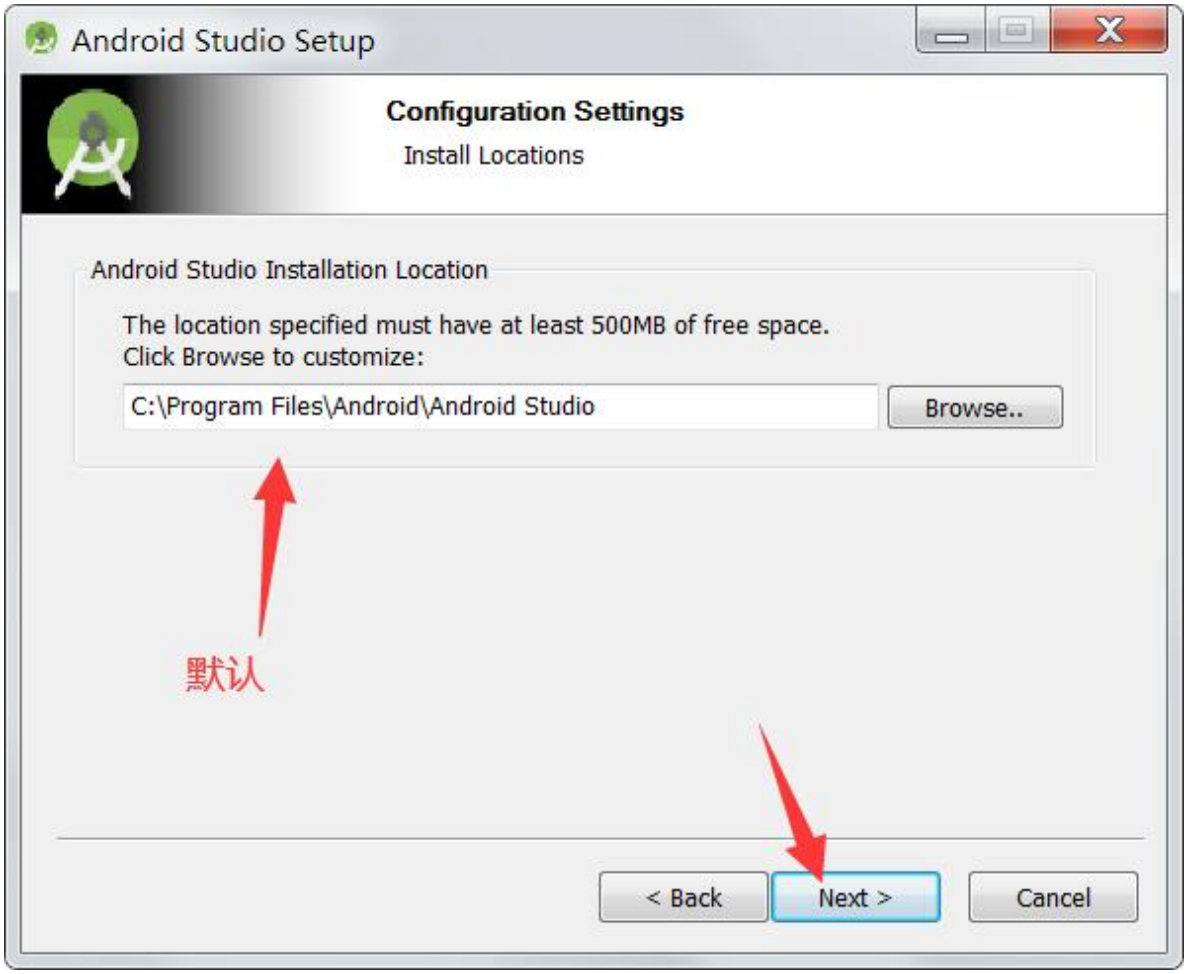


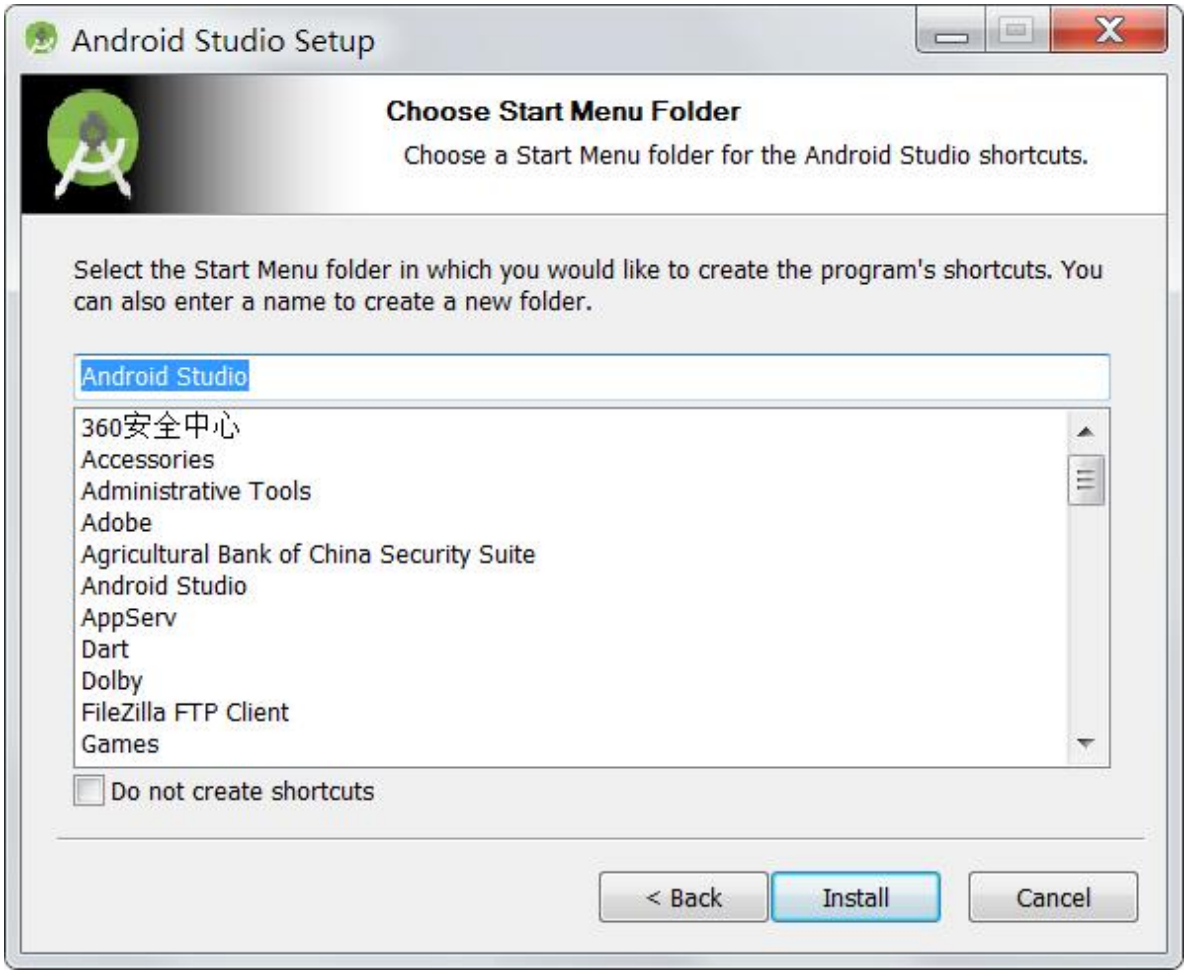
2.2 、电脑上下载安装Android Studio（2022.1.1 Patch 1）

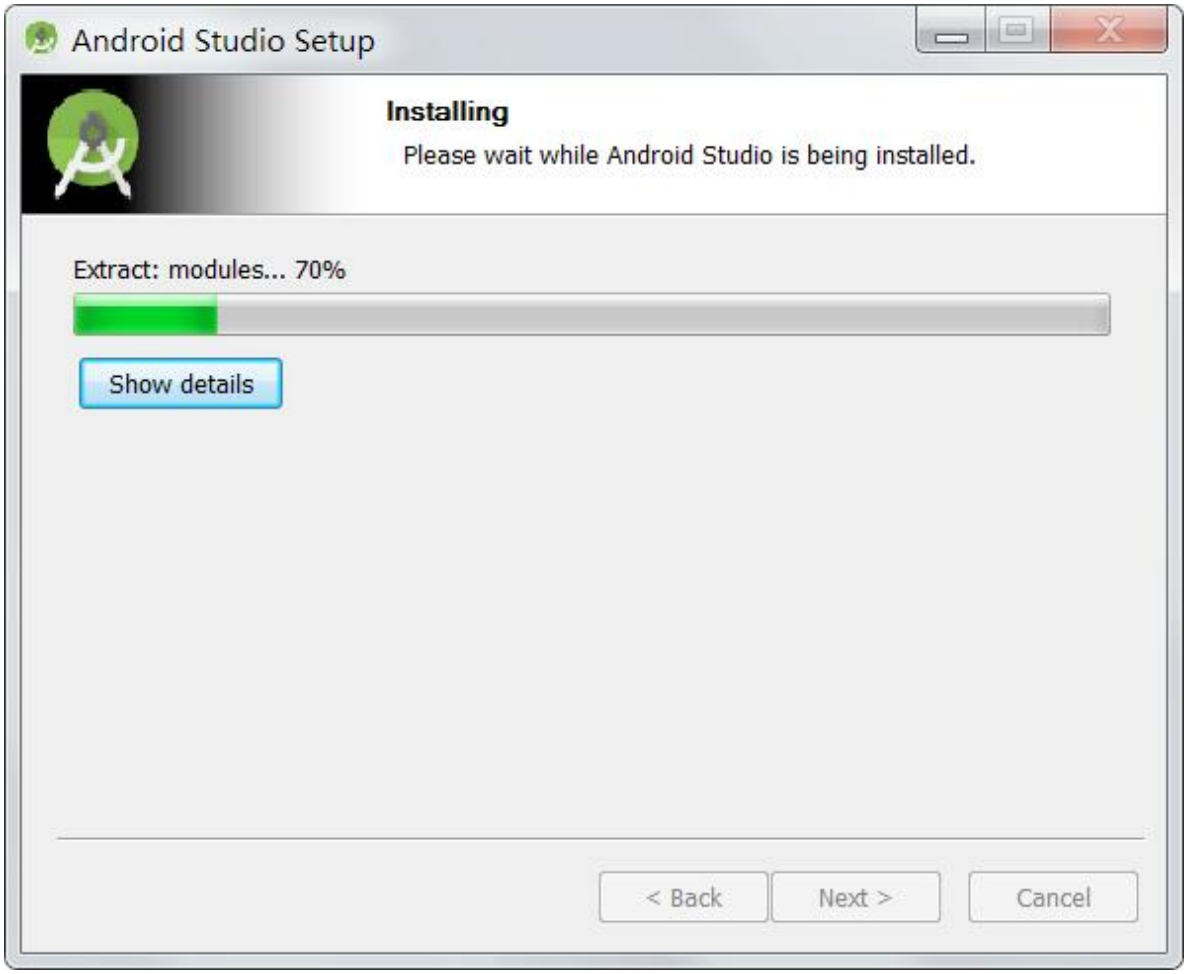
https://developer.android.google.cn/studio

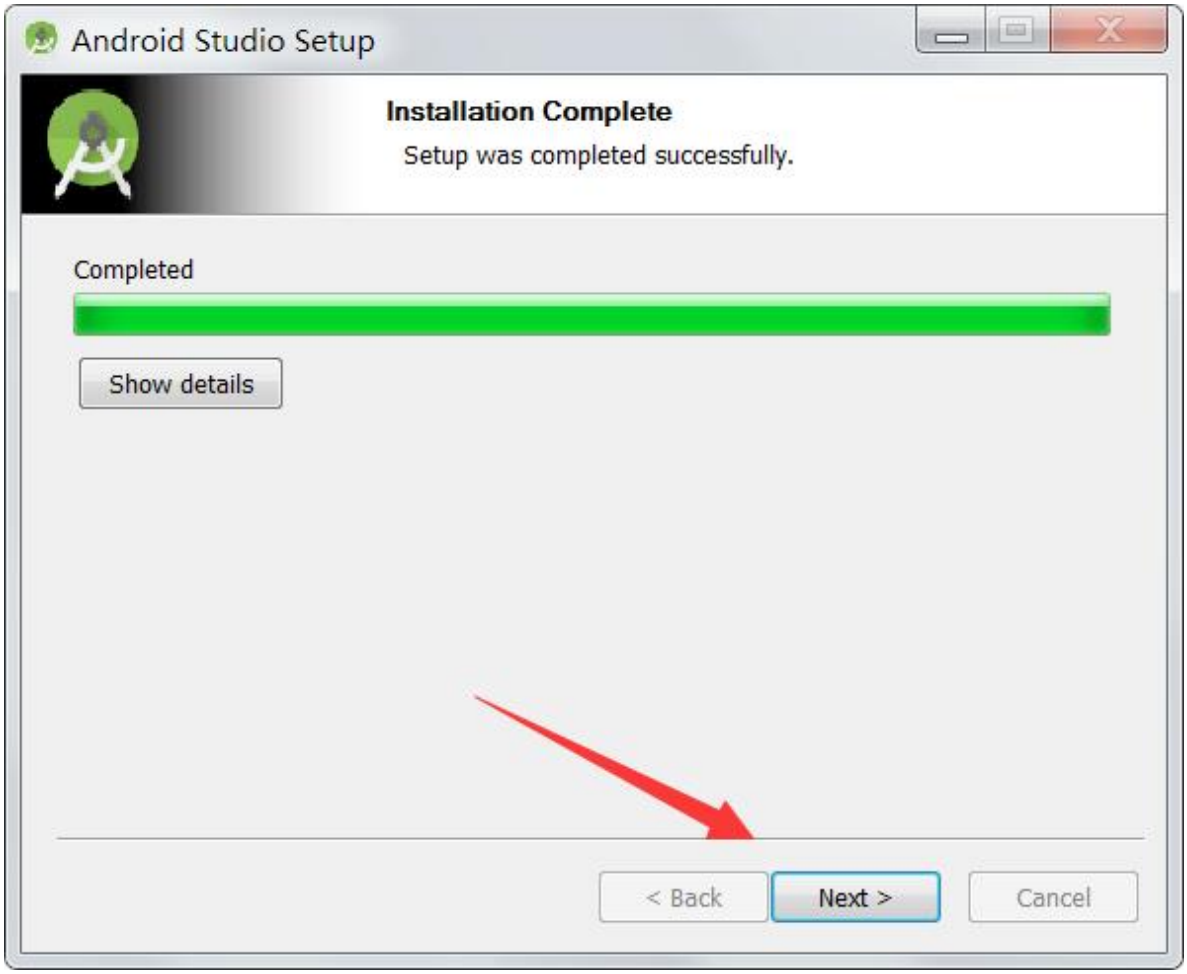

Android Studio Setup
Welcome to Android Studio Setup
Setup will guide you through the installation of Android Studio.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Android Studio
< Back
Next >
Cancel

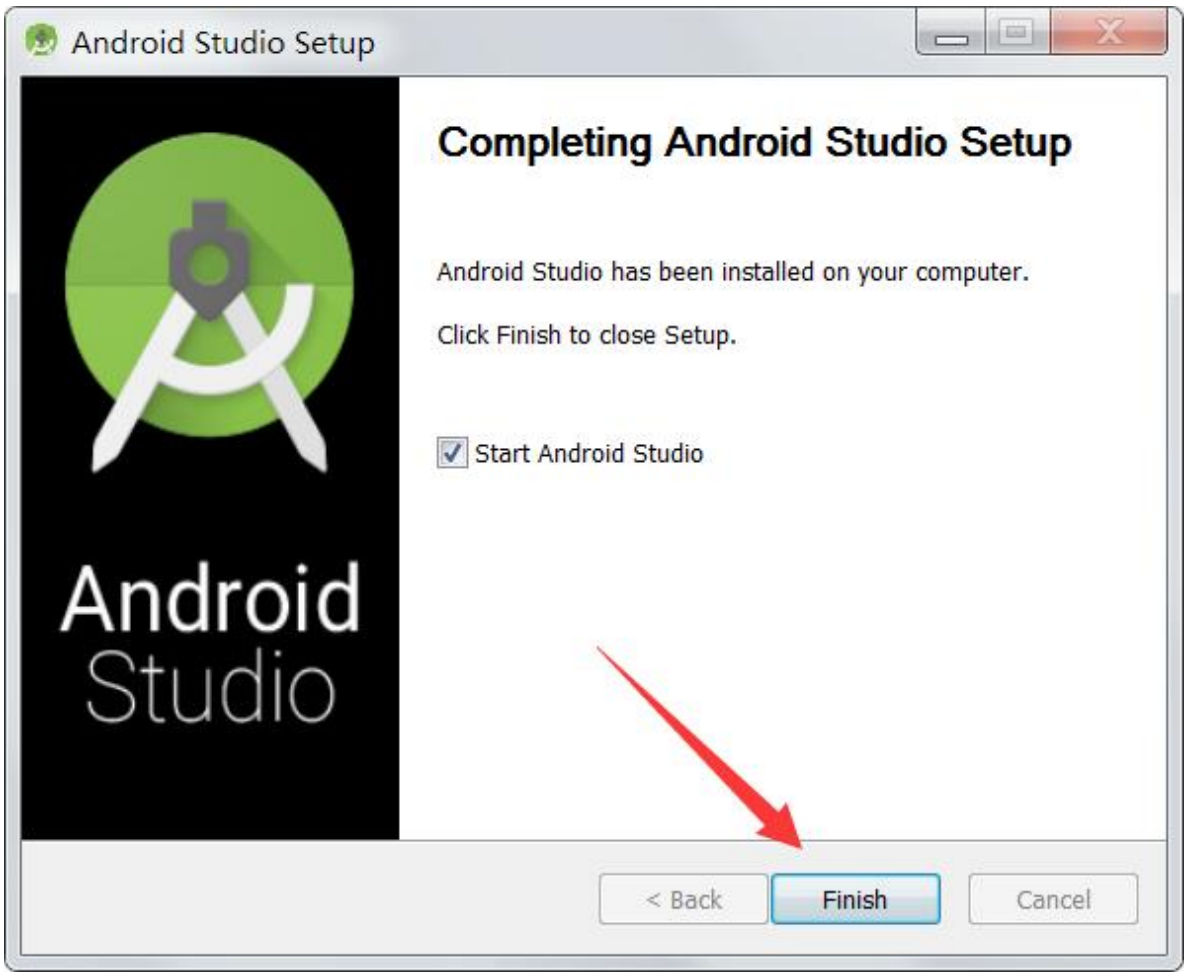

Android Studio Setup
Choose Components
Choose which features of Android Studio you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Android Studio
Android Virtual Device
Description
Position your mouse over a component to see its description.
Space required: 2.7GB
< Back
Next >
Cancel

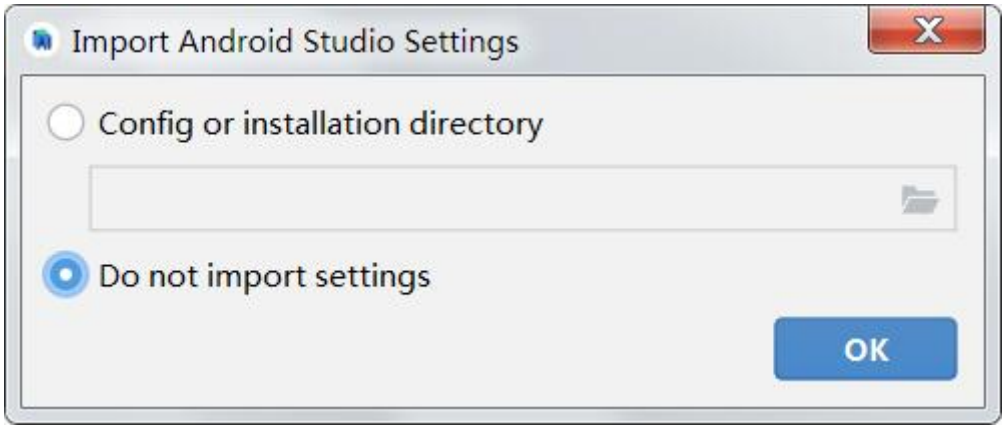

Android Studio Setup
Configuration Settings
Install Locations
Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio
Browse..
默认
< Back
Next >
Cancel

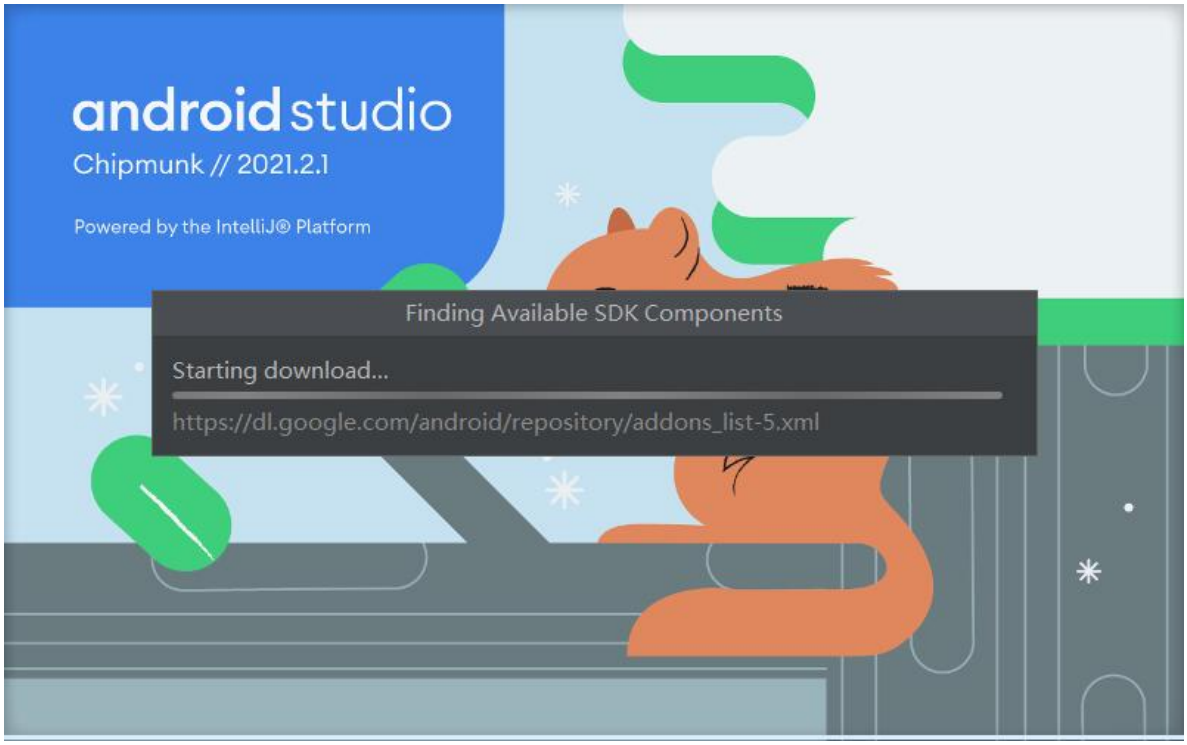

Android Studio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
Android Studio
360安全中心
Accessories
Administrative Tools
Adobe
Agricultural Bank of China Security Suite
Android Studio
AppServ
Dart
Dolby
FileZilla FTP Client
Games
Do not create shortcuts
< Back
Install
Cancel

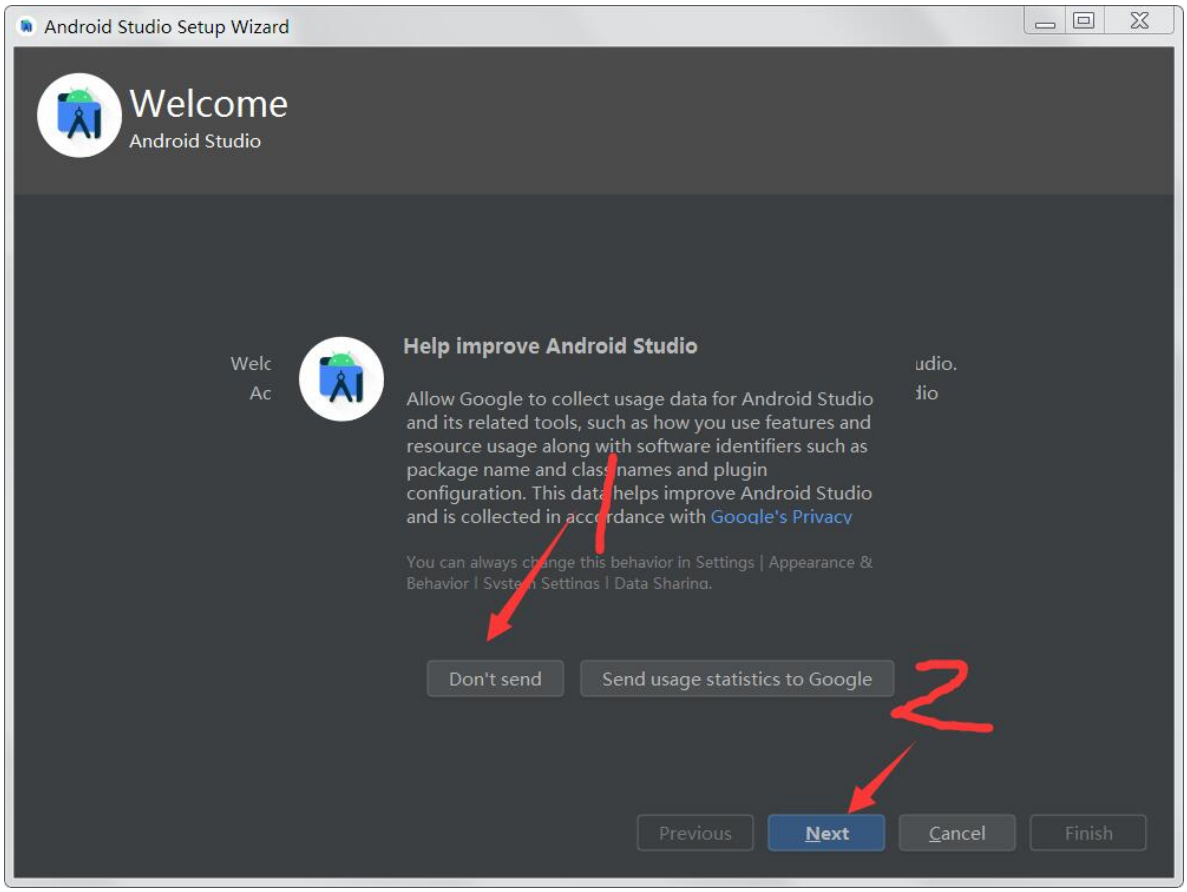

Android Studio Setup
Installing
Please wait while Android Studio is being installed.
Extract: modules... 70%
Show details
< Back
Next >
Cancel

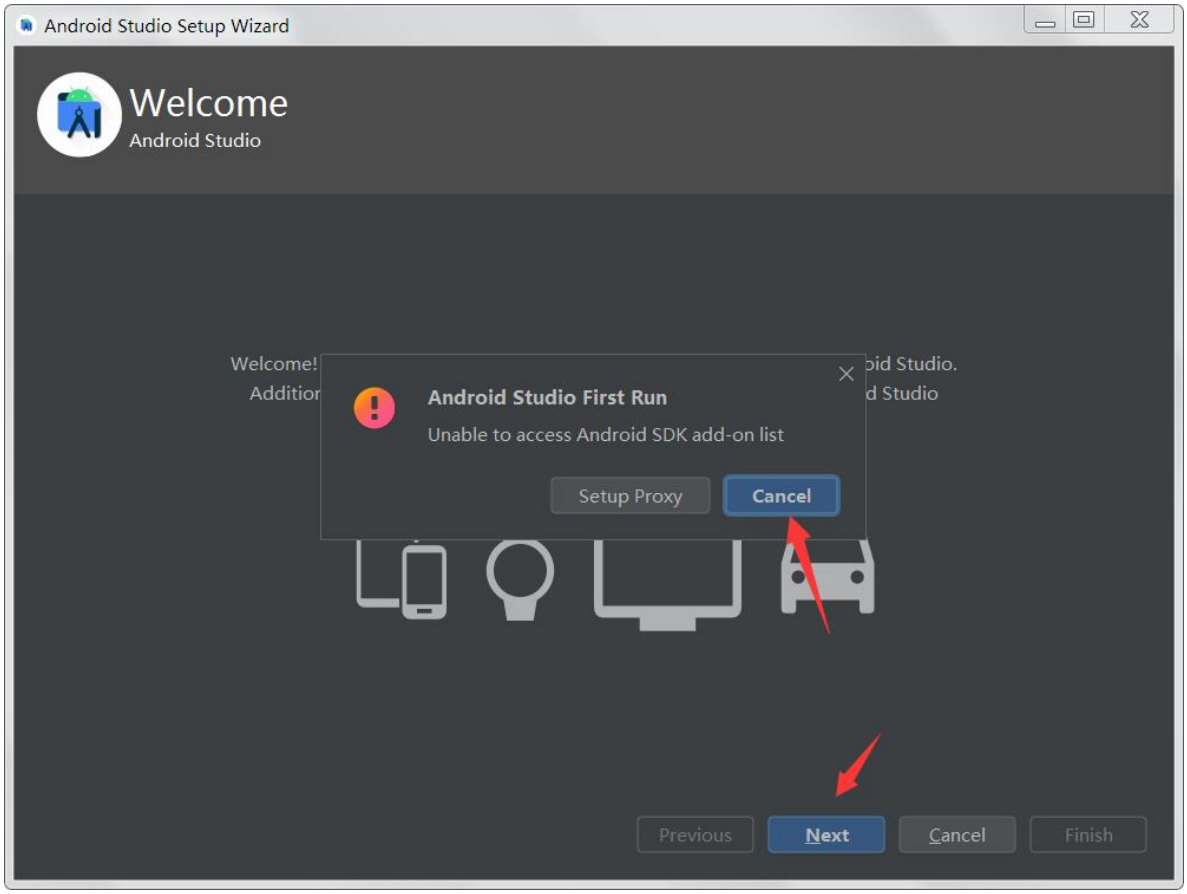

Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Next >
Cancel

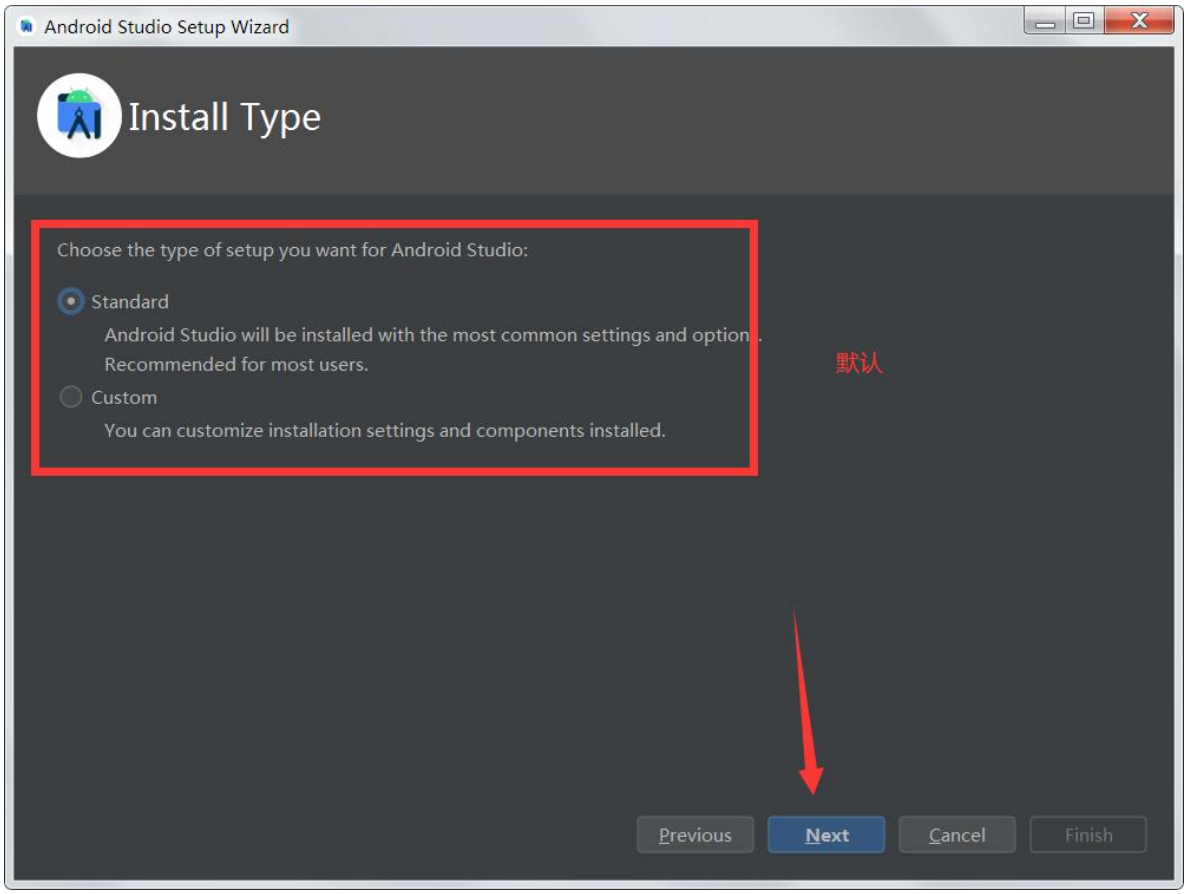

Android Studio Setup
Completing Android Studio Setup
Android Studio has been installed on your computer.
Click Finish to close Setup.
Start Android Studio
Android
Studio
< Back
Finish
Cancel

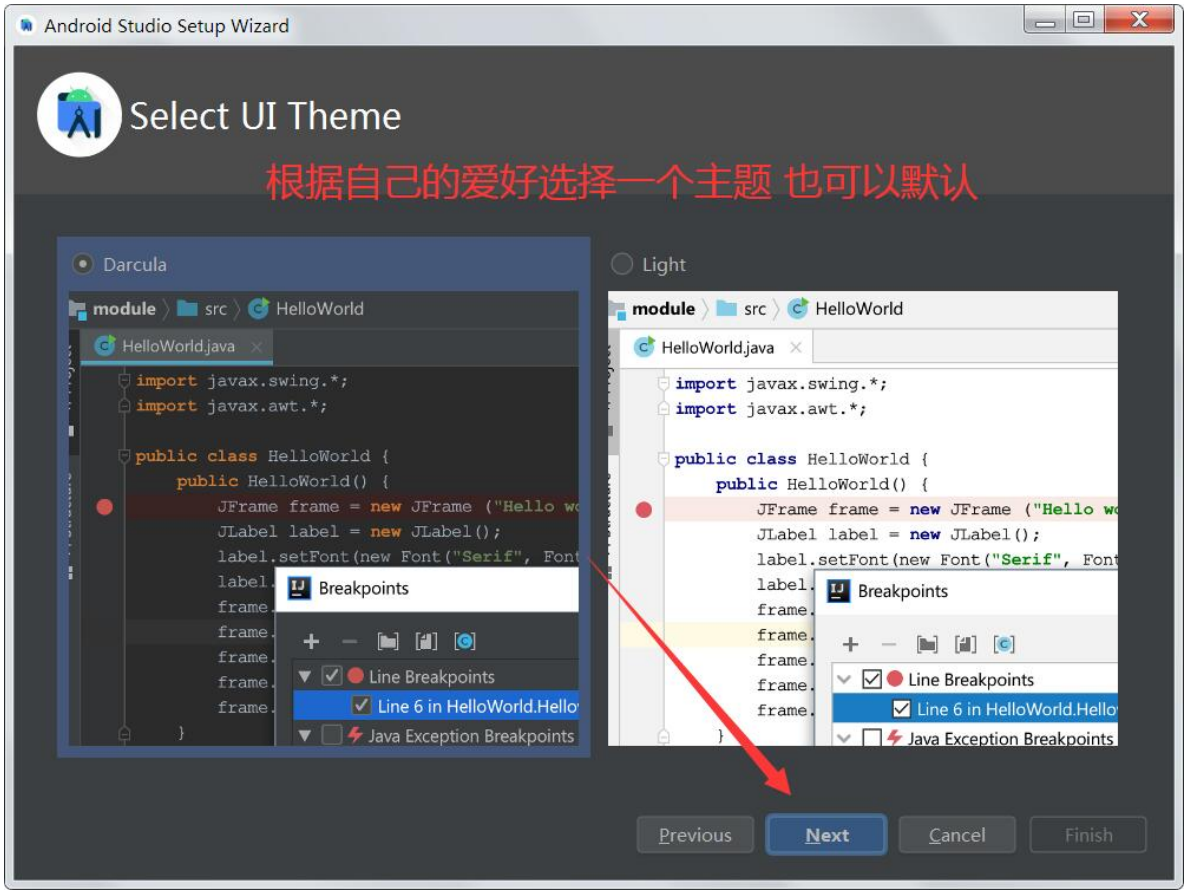

Import Android Studio Settings
Config or installation directory
Do not import settings
OK

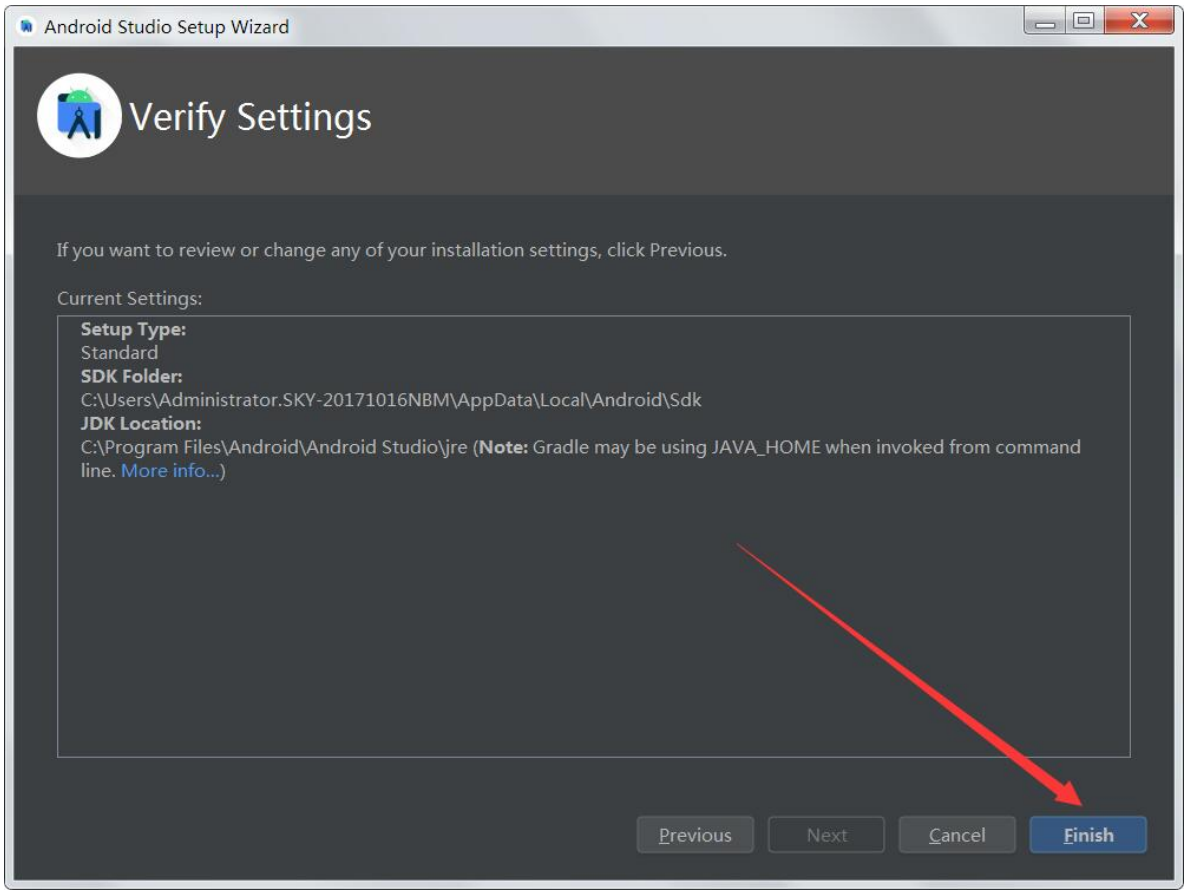

android studio
Chipmunk // 2021.2.1
Powered by the IntelliJ® Platform
Finding Available SDK Components
Starting download...
https://dl.google.com/android/repository/addons_list-5.xml

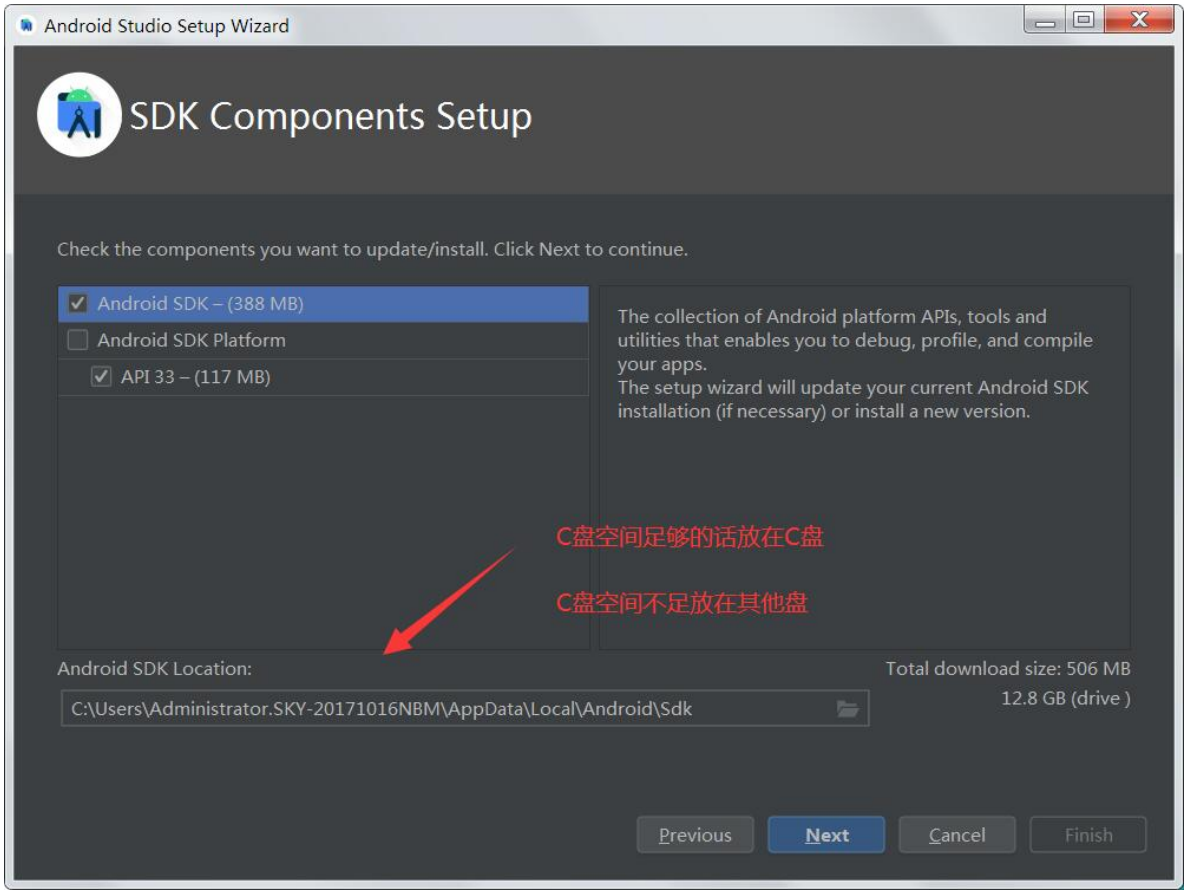

Android Studio Setup Wizard
Welcome
Android Studio
Help improve Android Studio
Allow Google to collect usage data for Android Studio and its related tools, such as how you use features and resource usage along with software identifiers such as package name and class names and plugin configuration. This data helps improve Android Studio and is collected in accordance with Google's Privacy
You can always change this behavior in Settings | Appearance & Behavior | System Settings | Data Sharing.
Don't send
Send usage statistics to Google
2
Previous
Next
Cancel
Finish

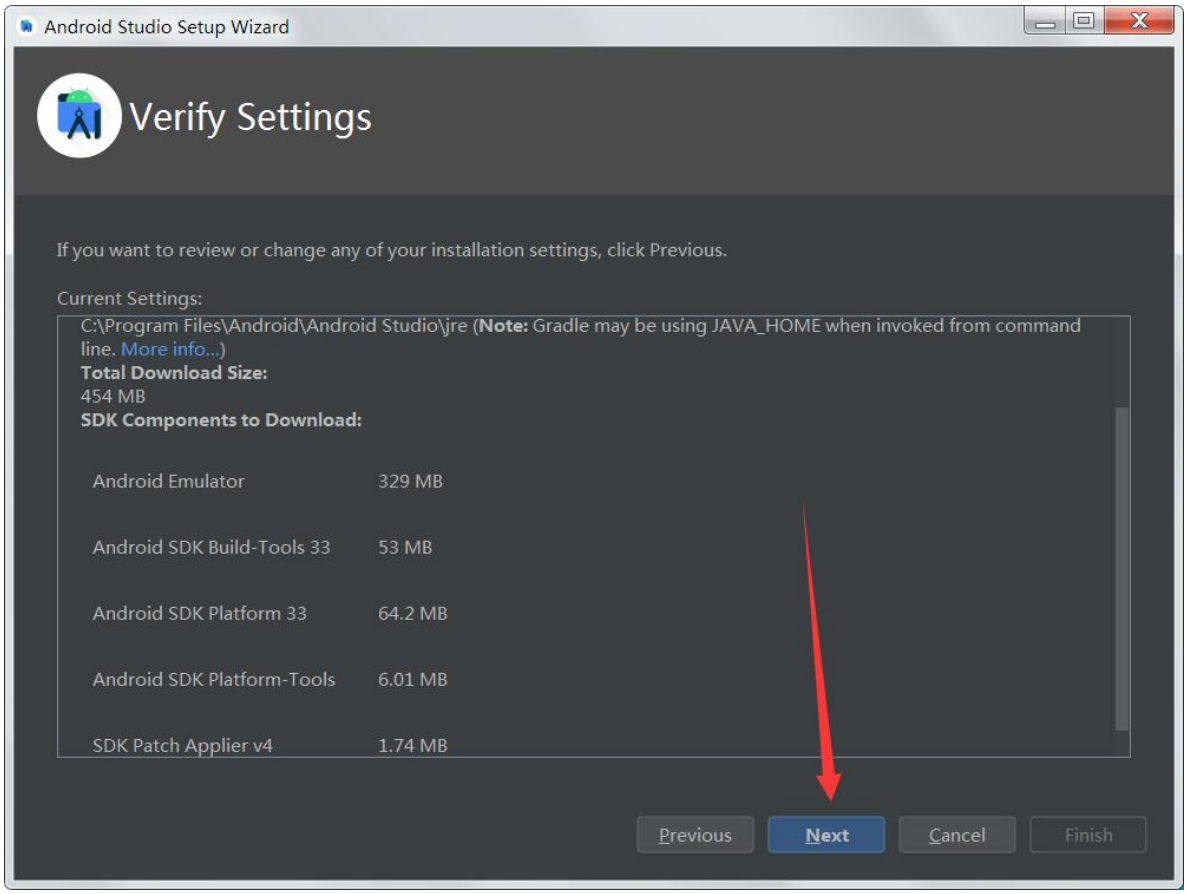

Android Studio Setup Wizard
Welcome
Android Studio
Welcome!
Addition
bid Studio.
d Studio
Android Studio First Run
Unable to access Android SDK add-on list
Setup Proxy
Cancel
Previous
Next
Cancel
Finish

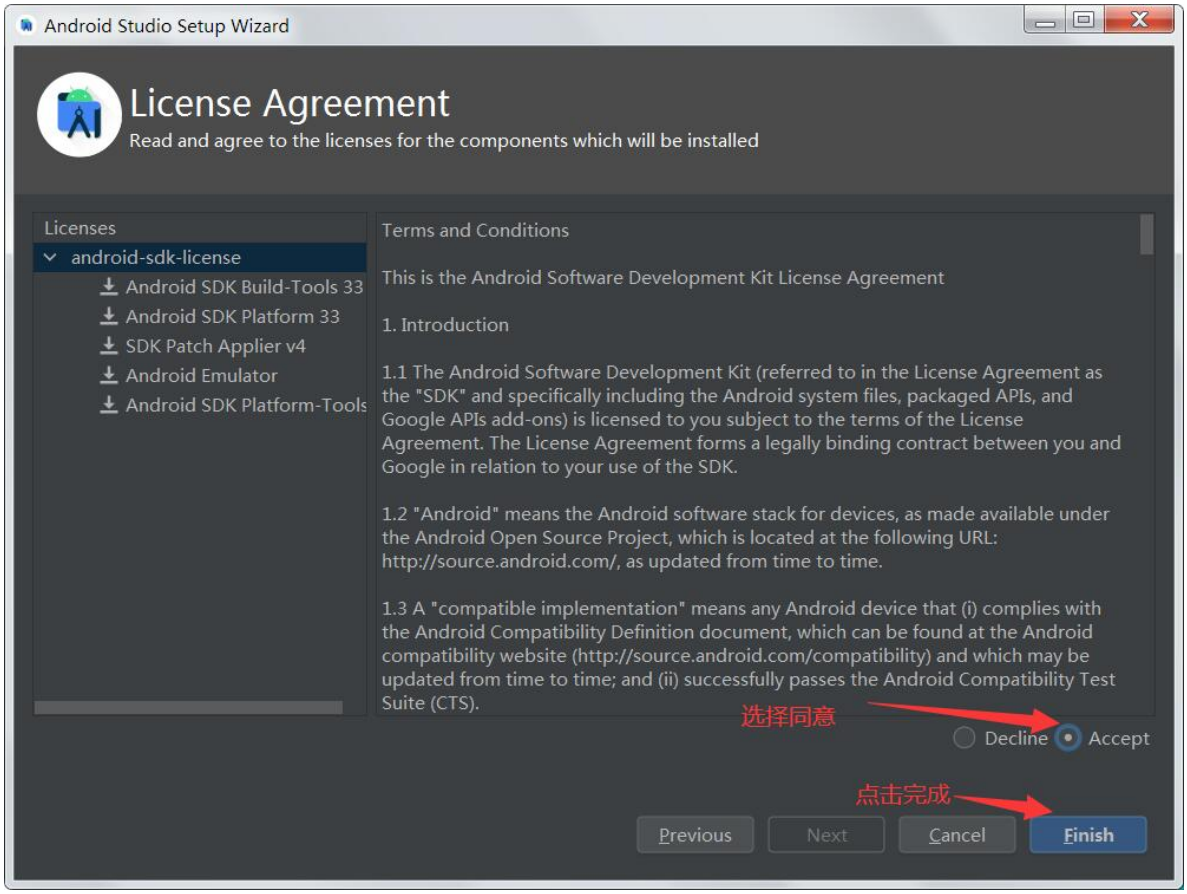

Android Studio Setup Wizard
Install Type
Choose the type of setup you want for Android Studio:
Standard
Android Studio will be installed with the most common settings and option
Recommended for most users.
Custom
You can customize installation settings and components installed.
默认
Previous
Next
Cancel
Finish

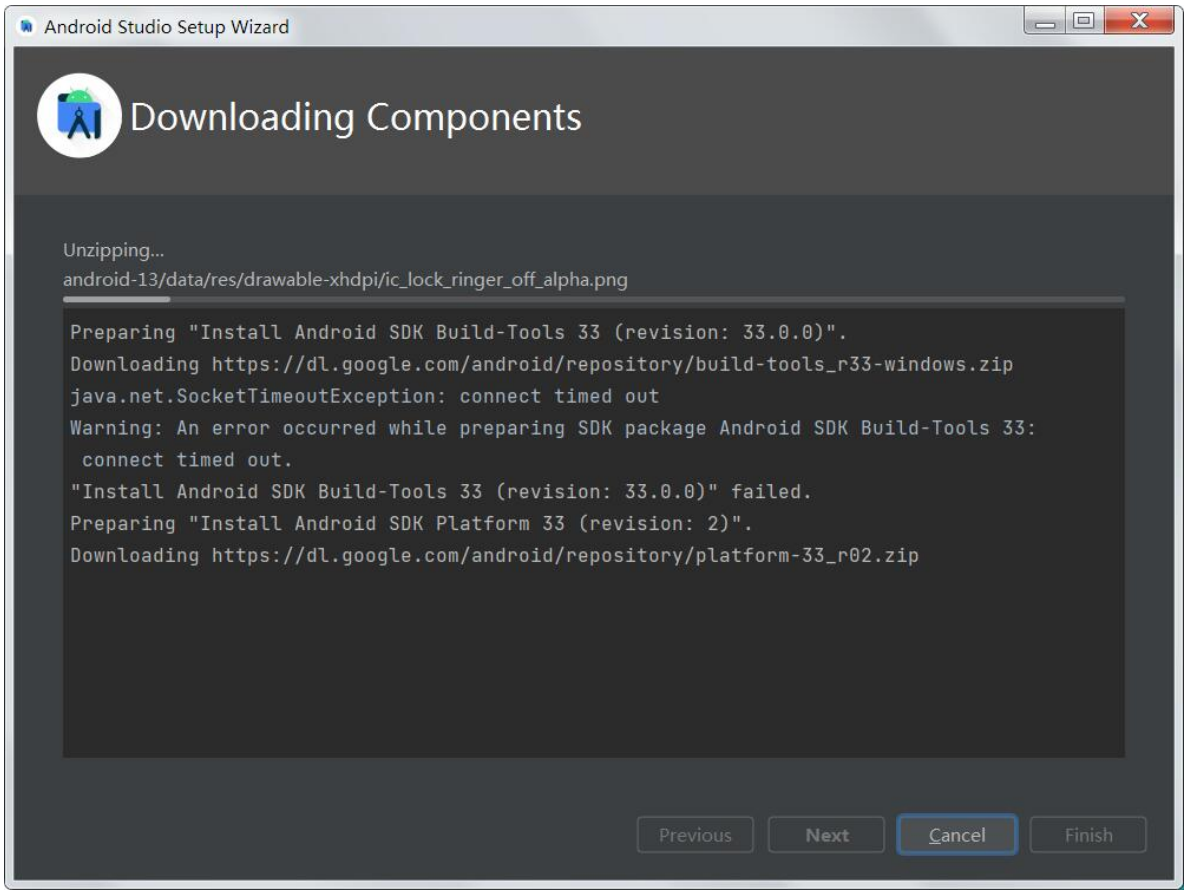

Android Studio Setup Wizard
Select UI Theme
根据自己的爱好选择一个主题 也可以默认
Darcula
Light
module
src
HelloWorld
HelloWorld.java
import javax.swing.*;
import javax.awt.*;
public class HelloWorld {
public HelloWorld() {
JFrame frame = new JFrame ("Hello
JLabel label = new JLabel();
label.setFont(new Font("Serif", Font
Breakpoints
Line Breakpoints
Line 6 in HelloWorld.Hello
Java Exception Breakpoints
Previous
Next
Cancel
Finish

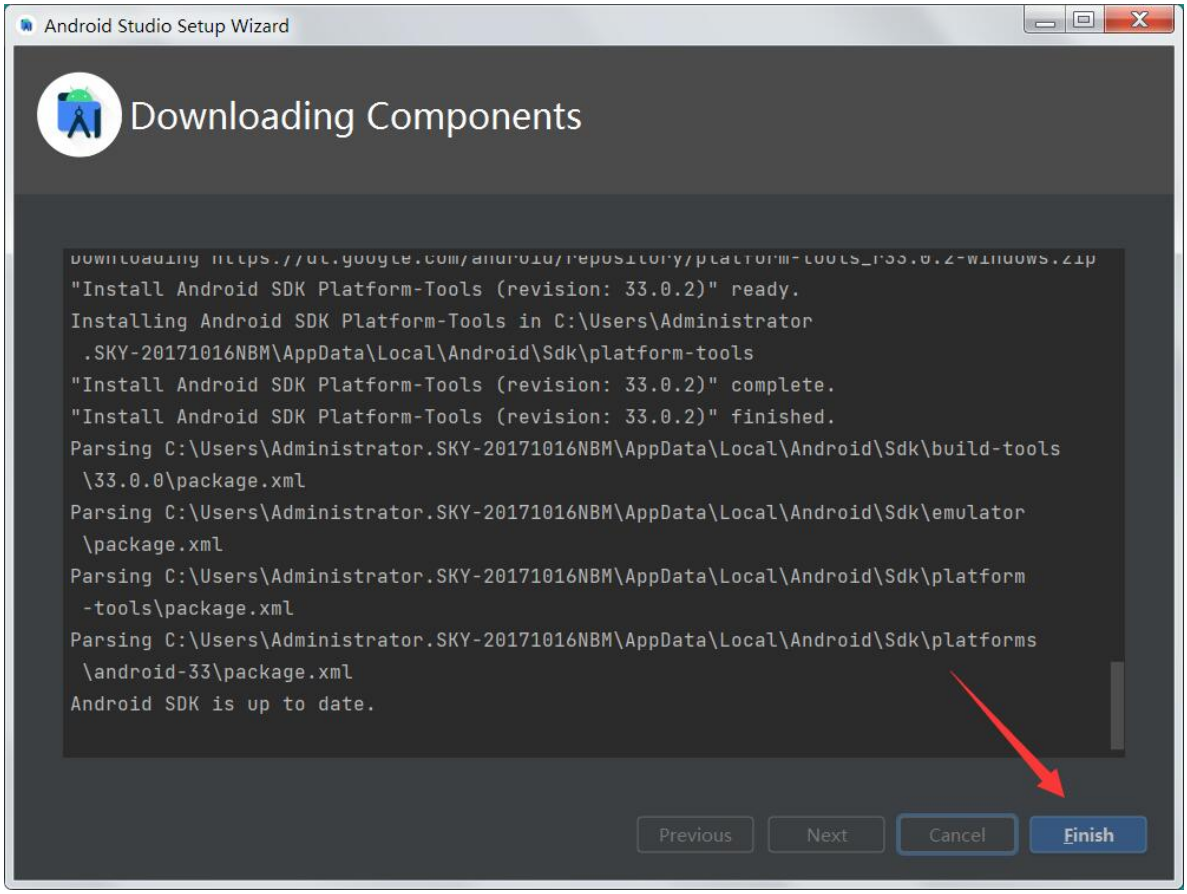

Android Studio Setup Wizard
Verify Settings
If you want to review or change any of your installation settings, click Previous.
Current Settings:
Setup Type:
Standard
SDK Folder:
C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk
JDK Location:
C:\Program Files\Android\Android Studio\jre (Note: Gradle may be using JAVA_HOME when invoked from command line. More info...)
Previous
Next
Cancel
Finish

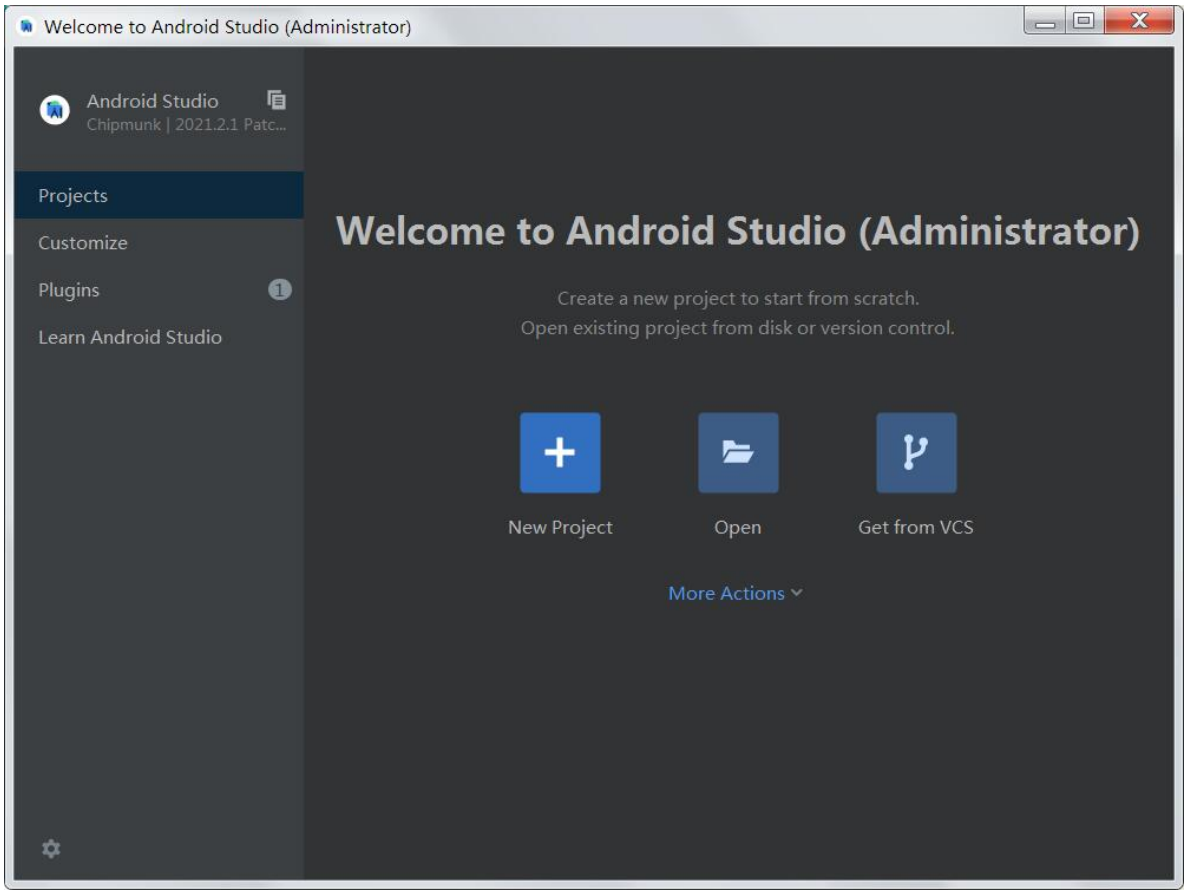

Android Studio Setup Wizard
SDK Components Setup
Check the components you want to update/install. Click Next to continue.
Android SDK – (388 MB)
Android SDK Platform
API 33 – (117 MB)
The collection of Android platform APIs, tools and utilities that enables you to debug, profile, and compile your apps.
The setup wizard will update your current Android SDK installation (if necessary) or install a new version.
C盘空间足够的话放在C盘
C盘空间不足放在其他盘
Android SDK Location:
C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk
Total download size: 506 MB
12.8 GB (drive)
Previous
Next
Cancel
Finish

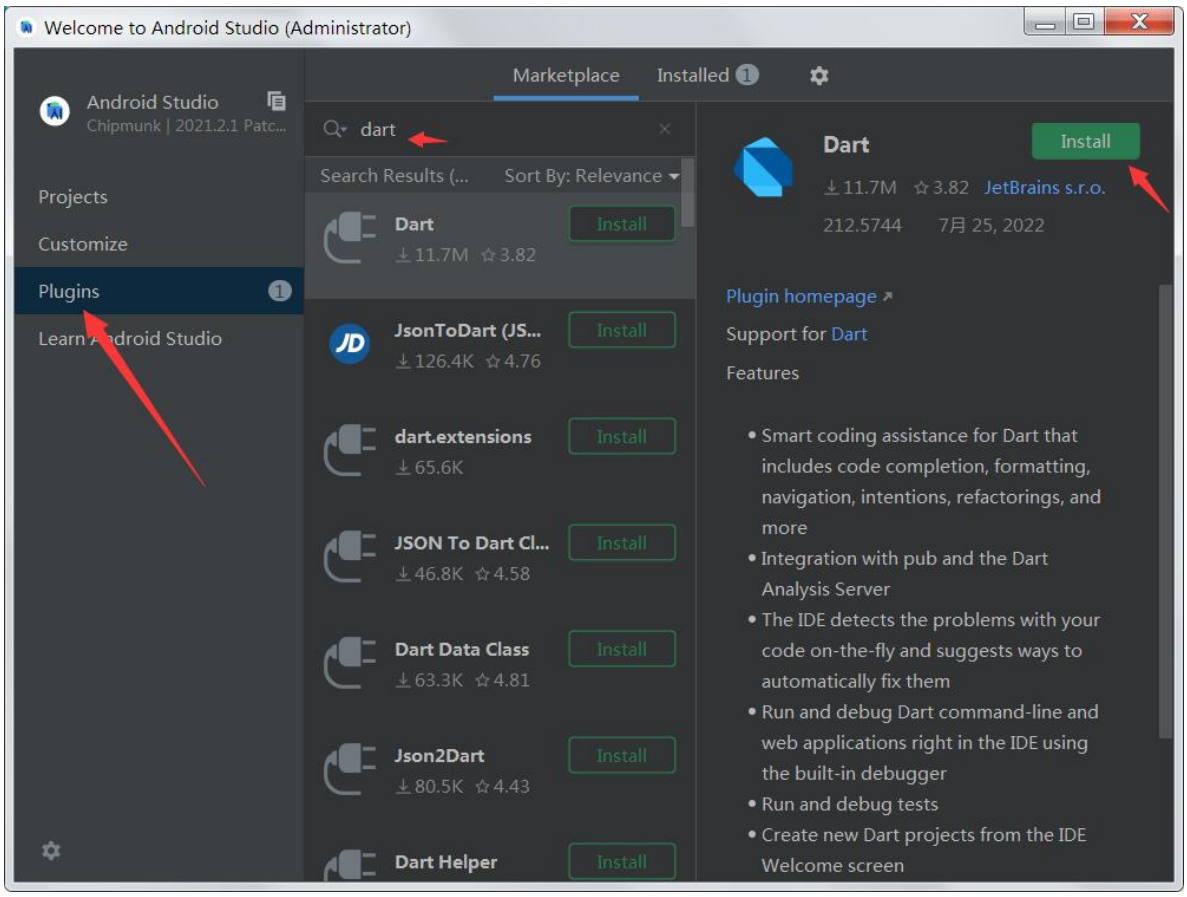

Android Studio Setup Wizard
Verify Settings
If you want to review or change any of your installation settings, click Previous.
Current Settings:
C:\Program Files\Android\Android Studio\jre (Note: Gradle may be using JAVA_HOME when invoked from command line. More info...)
Total Download Size:
454 MB
SDK Components to Download:
Android Emulator 329 MB
Android SDK Build-Tools 33 53 MB
Android SDK Platform 33 64.2 MB
Android SDK Platform-Tools 6.01 MB
SDK Patch Applier v4 1.74 MB
Previous
Next
Cancel
Finish

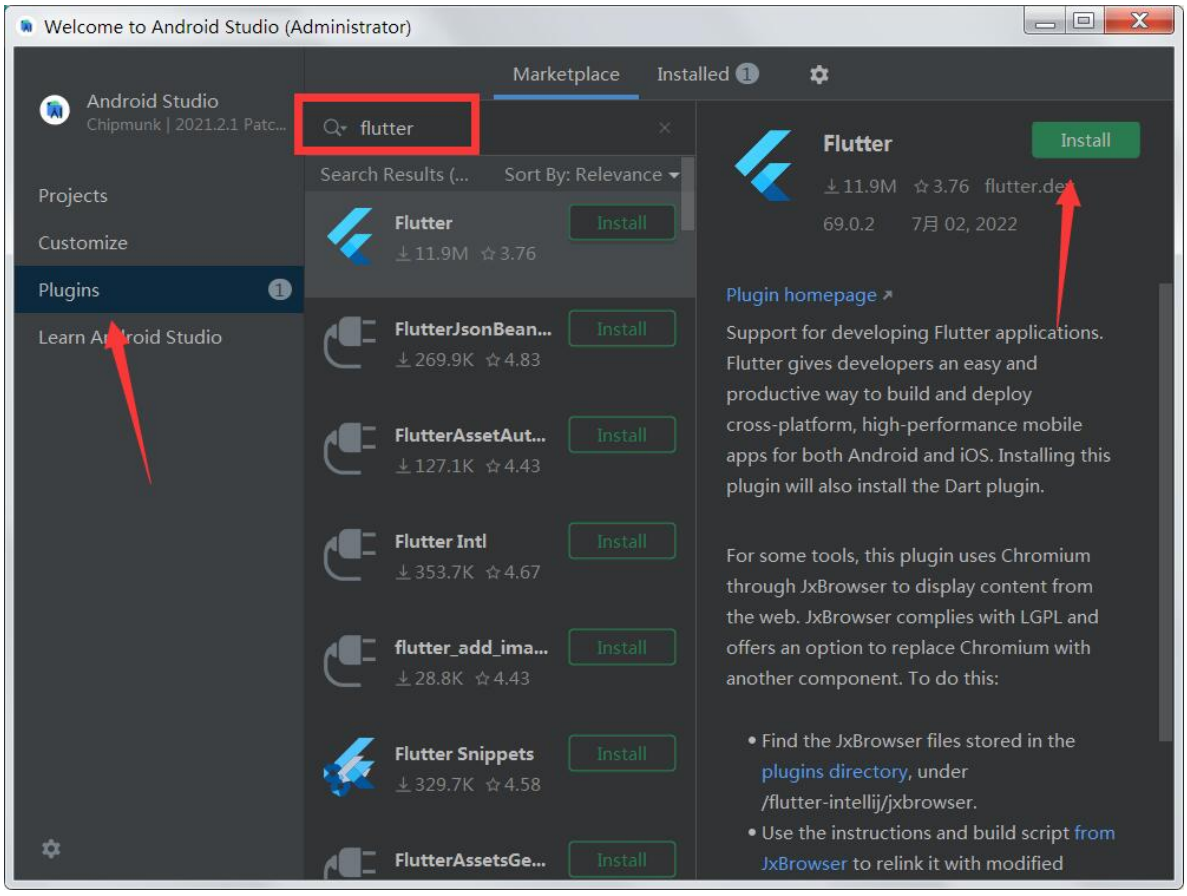

Android Studio Setup Wizard
License Agreement
Read and agree to the licenses for the components which will be installed
Licenses
android-sdk-license
Android SDK Build-Tools 33
Android SDK Platform 33
SDK Patch Applier v4
Android Emulator
Android SDK Platform-Tools
Terms and Conditions
This is the Android Software Development Kit License Agreement
1. Introduction
1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.
1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).
选择同意
Decline
Accept
点击完成
Previous
Next
Cancel
Finish

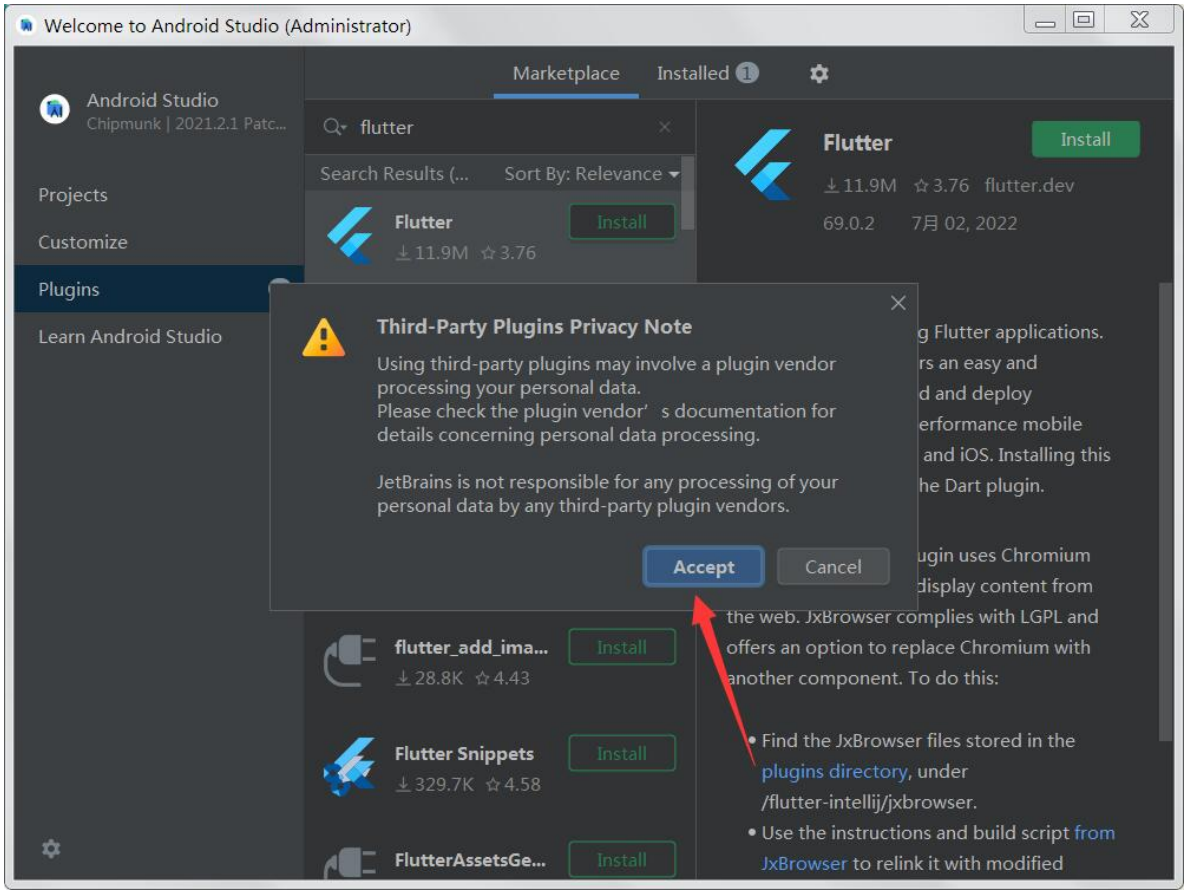

Android Studio Setup Wizard
Downloading Components
Unzipping...
android-13/data/res/drawable-xhdpi/ic_lock_ringer_off_alpha.png
Preparing "Install Android SDK Build-Tools 33 (revision: 33.0.0)".
Downloading https://dl.google.com/android/repository/build-tools_r33-windows.zip
java.net.SocketTimeoutException: connect timed out
Warning: An error occurred while preparing SDK package Android SDK Build-Tools 33:
connect timed out.
"Install Android SDK Build-Tools 33 (revision: 33.0.0)" failed.
Preparing "Install Android SDK Platform 33 (revision: 2)".
Downloading https://dl.google.com/android/repository/platform-33_r02.zip
Previous
Next
Cancel
Finish

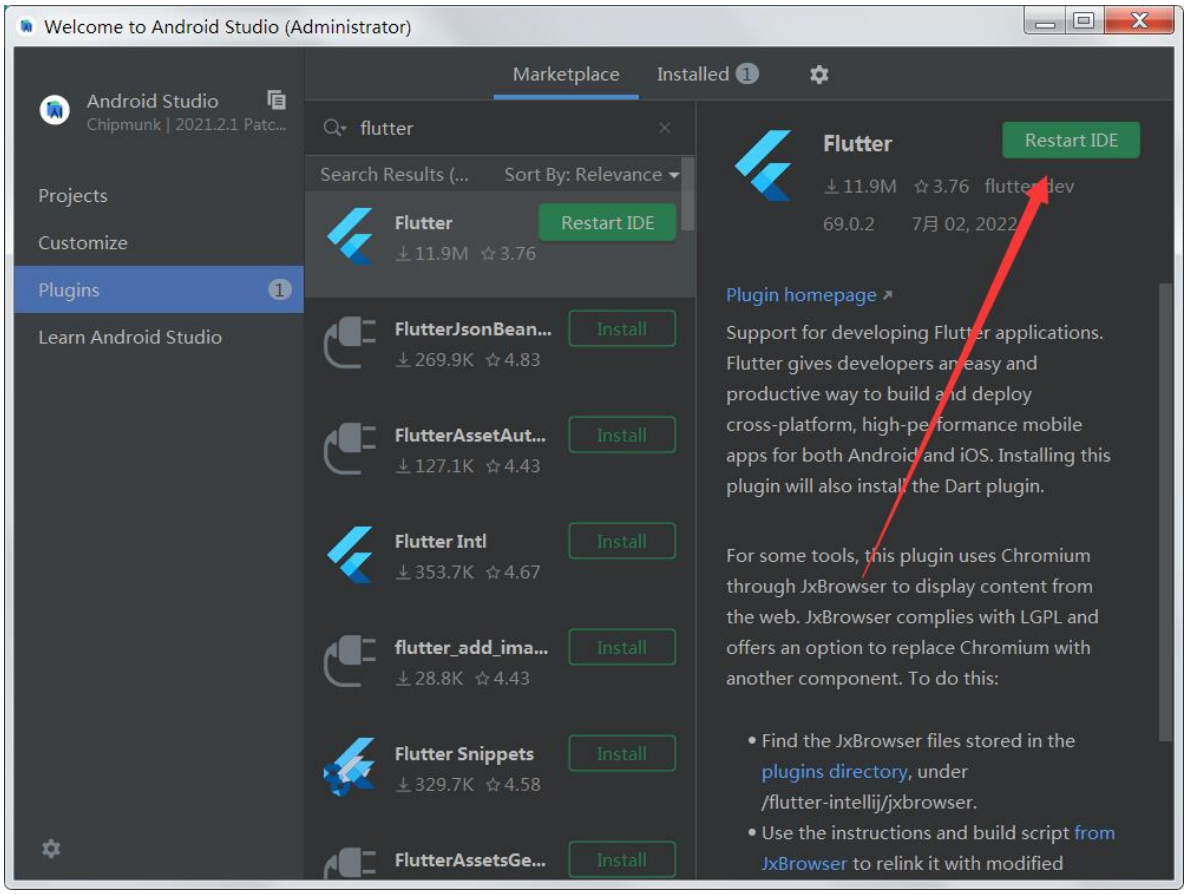

Android Studio Setup Wizard
Downloading Components
"Install Android SDK Platform-Tools (revision: 33.0.2)" ready.
Installing Android SDK Platform-Tools in C:\Users\Administrator
.SKY-20171016NBM\AppData\Local\Android\Sdk\platform-tools
"Install Android SDK Platform-Tools (revision: 33.0.2)" complete.
"Install Android SDK Platform-Tools (revision: 33.0.2)" finished.
Parsing C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk\build-tools
\33.0.0\package.xml
Parsing C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk\emulator
\package.xml
Parsing C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk\platform
-tools\package.xml
Parsing C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk\platforms
\android-33\package.xml
Android SDK is up to date.
Previous
Next
Cancel
Finish

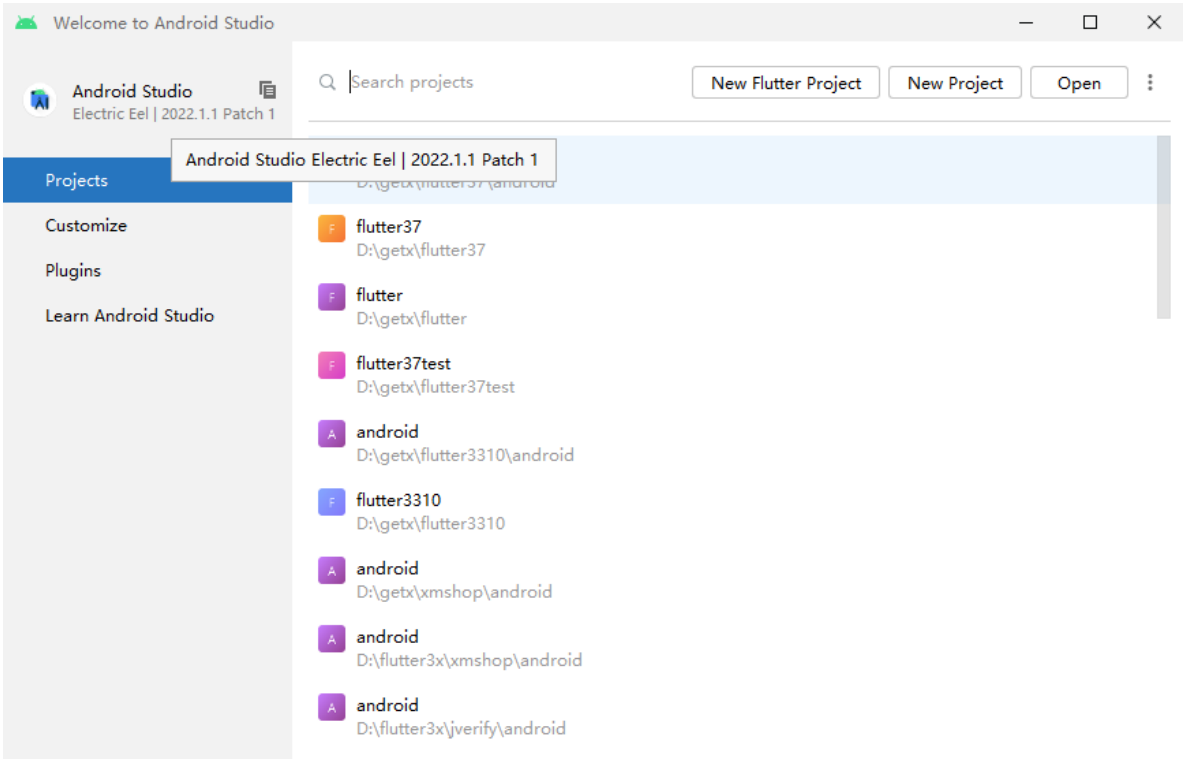

Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio (Administrator)
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
More Actions


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
dart
Search Results (...
Sort By: Relevance
Dart
Install
11.7M ☆3.82
JsonToDart (JS...
126.4K ☆4.76
dart.extensions
65.6K
JSON To Dart Cl...
46.8K ☆4.58
Dart Data Class
63.3K ☆4.81
Json2Dart
80.5K ☆4.43
Dart Helper
Dart
11.7M ☆3.82 JetBrains s.r.o.
212.5744 7月 25, 2022
Plugin homepage
Support for Dart
Features
• Smart coding assistance for Dart that includes code completion, formatting, navigation, intentions, refactorings, and more
• Integration with pub and the Dart Analysis Server
• The IDE detects the problems with your code on-the-fly and suggests ways to automatically fix them
• Run and debug Dart command-line and web applications right in the IDE using the built-in debugger
• Run and debug tests
• Create new Dart projects from the IDE Welcome screen


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutter
Search Results (...
Sort By: Relevance
Flutter
Install
11.9M ☆3.76
FlutterJsonBean...
269.9K ☆4.83
FlutterAssetAut...
127.1K ☆4.43
Flutter Intl
353.7K ☆4.67
flutter_add_ima...
28.8K ☆4.43
Flutter Snippets
329.7K ☆4.58
FlutterAssetsGe...
Flutter
11.9M ☆3.76 flutter.de
69.0.2 7月 02, 2022
Plugin homepage
Support for developing Flutter applications. Flutter gives developers an easy and productive way to build and deploy cross-platform, high-performance mobile apps for both Android and iOS. Installing this plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium through JxBrowser to display content from the web. JxBrowser complies with LGPL and offers an option to replace Chromium with another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutter
Search Results (...
Sort By: Relevance
Flutter
Install
11.9M 3.76
Third-Party Plugins Privacy Note
Using third-party plugins may involve a plugin vendor processing your personal data.
Please check the plugin vendor' s documentation for details concerning personal data processing.
JetBrains is not responsible for any processing of your personal data by any third-party plugin vendors.
Accept
Cancel
flutter_add_ima...
28.8K 4.43
Flutter Snippets
329.7K 4.58
FlutterAssetsGe...
Flutter
Install
11.9M 3.76 flutter.dev
69.0.2 7月 02, 2022
g Flutter applications.
rs an easy and
d and deploy
erformance mobile
and iOS. Installing this
he Dart plugin.
ugin uses Chromium
display content from
the web. JxBrowser complies with LGPL and
offers an option to replace Chromium with
another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
1
Learn Android Studio
Marketplace
Installed
flutter
Search Results (...
Sort By: Relevance
Flutter
Restart IDE
11.9M 3.76
FlutterJsonBean...
269.9K 4.83
FlutterAssetAut...
127.1K 4.43
Flutter Intl
353.7K 4.67
flutter_add_ima...
28.8K 4.43
Flutter Snippets
329.7K 4.58
FlutterAssetsGe...
Install
Restart IDE
11.9M 3.76 flutter.dev
69.0.2 7月 02, 2022
Plugin homepage
Support for developing Flutter applications.
Flutter gives developers an easy and
productive way to build and deploy
cross-platform, high-performance mobile
apps for both Android and iOS. Installing this
plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium
through JxBrowser to display content from
the web. JxBrowser complies with LGPL and
offers an option to replace Chromium with
another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified



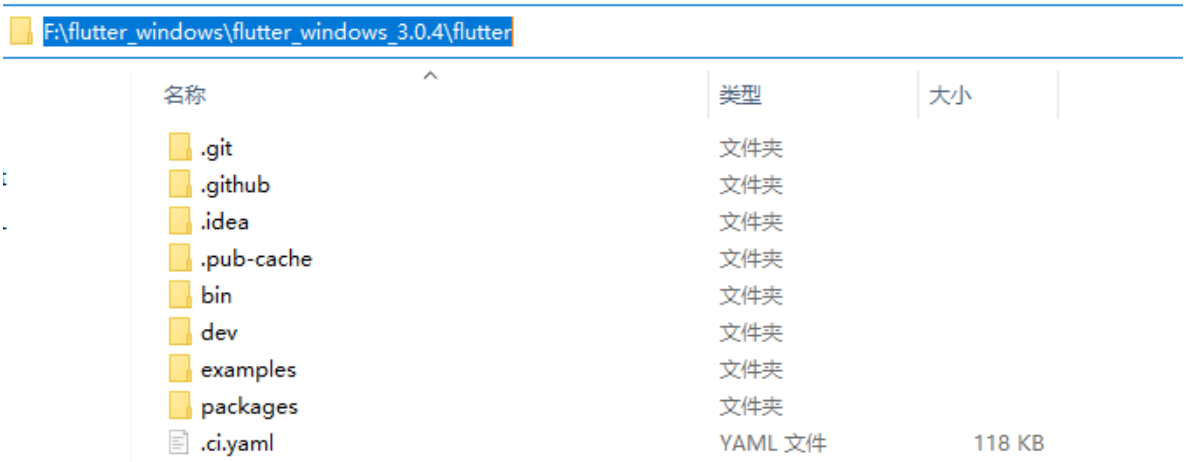
2.3、电脑上面下载配置Flutter Sdk（所有版本方法一样）

1、下载Flutter SDK

https://flutter.dev/docs/development/tools/sdk/releases#windows

2、把下载好的Flutter SDK随便减压到你想安装Sdk的目录

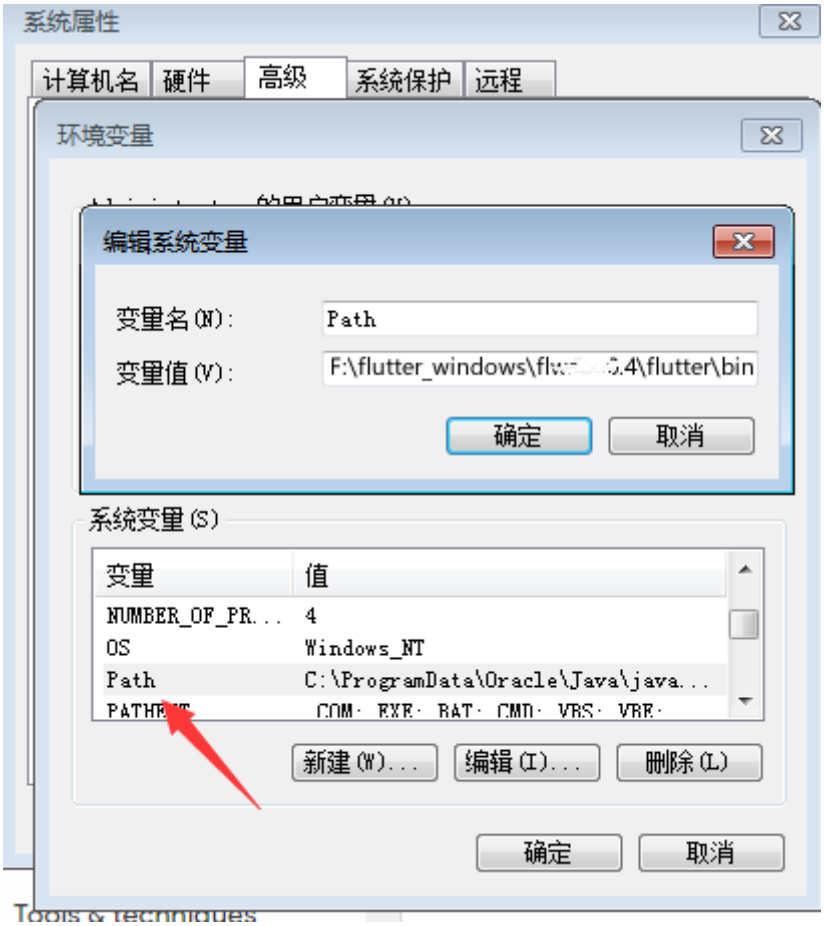
如减压到（F:\flutter_windows\flutter_windows_3.0.4\flutter）



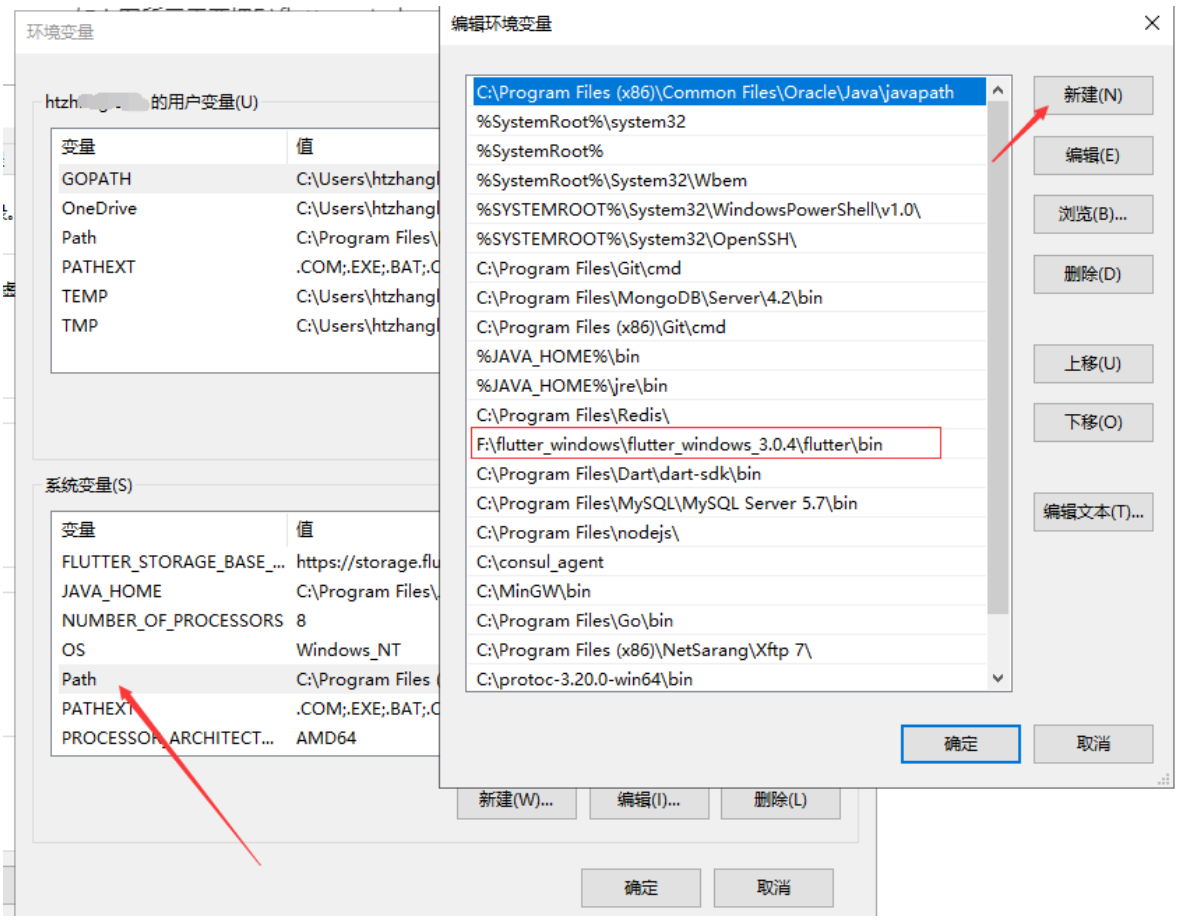
3、把 Flutter安装目录的bin目录配置到环境变量。

如上图所示需要把F:\flutter_windows\flutter_windows_3.0.4\flutter\bin目录配置到path环境变量里面

windows7：



windows10、windows11：



2.4、电脑上配置Flutter国内镜像

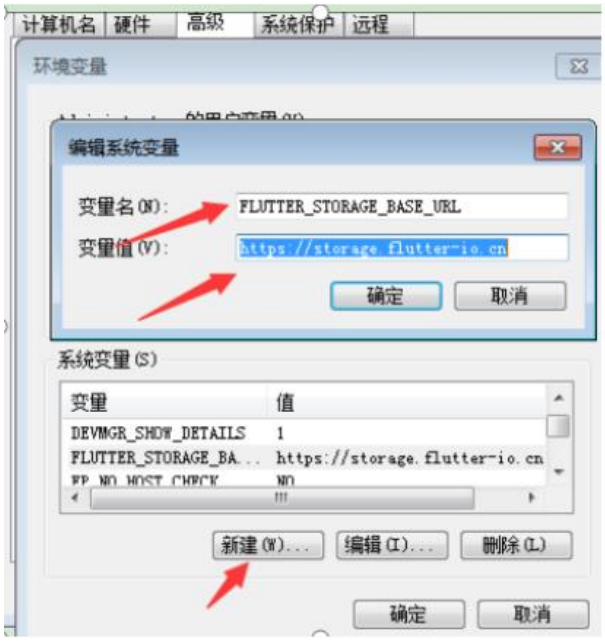
搭建环境过程中要下载很多资源文件，当一些资源下载不了的时候，可能会报各种错误。在国内访问Flutter的时候有可能会受到限制。Flutter官方为我们提供了国内的镜像

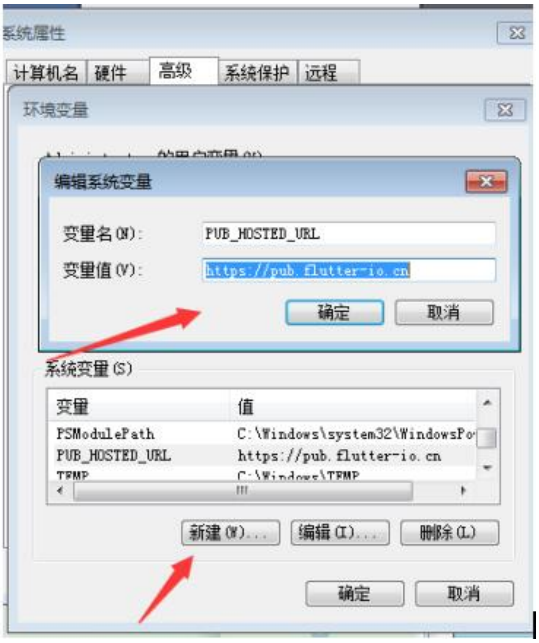
https://flutter.dev/community/china

https://flutter-io.cn/

拉到Flutter中文网最下面有配置方式，把下面两句配置到环境变量里面

```
FLUTTER_STORAGE_BASE_URL: https://storage.flutter-io.cn
PUB_HOSTED_URL: https://pub.flutter-io.cn
```





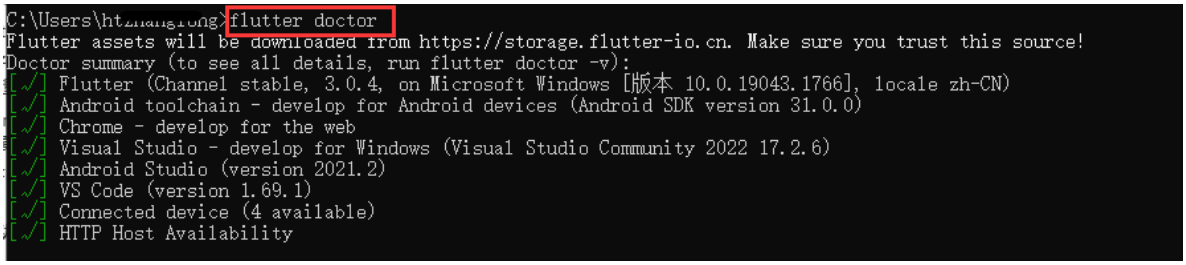
Flutter 社区镜像

```
FLUTTER_STORAGE_BASE_URL: https://storage.flutter-io.cn
PUB_HOSTED_URL: https://pub.flutter-io.cn
```

清华大学 TUNA 协会镜像

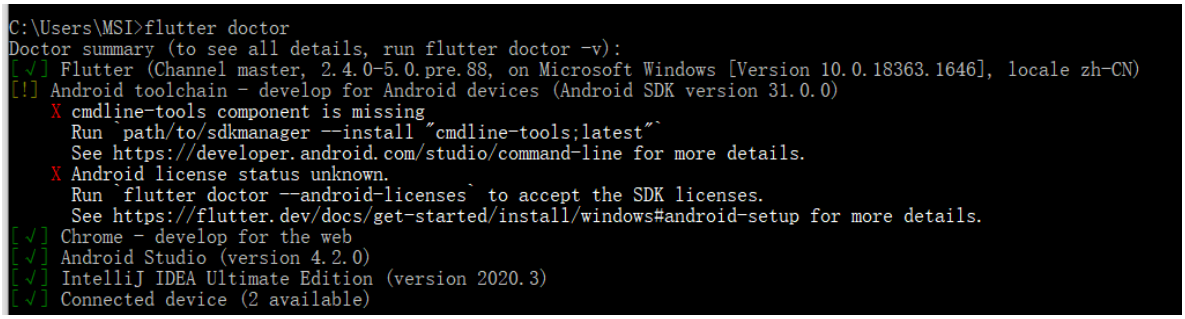
```
FLUTTER_STORAGE_BASE_URL: https://mirrors.tuna.tsinghua.edu.cn/flutter
PUB_HOSTED_URL: https://mirrors.tuna.tsinghua.edu.cn/dart-pub
```

2.5、运行 flutter doctor命令检测环境是否配置成功

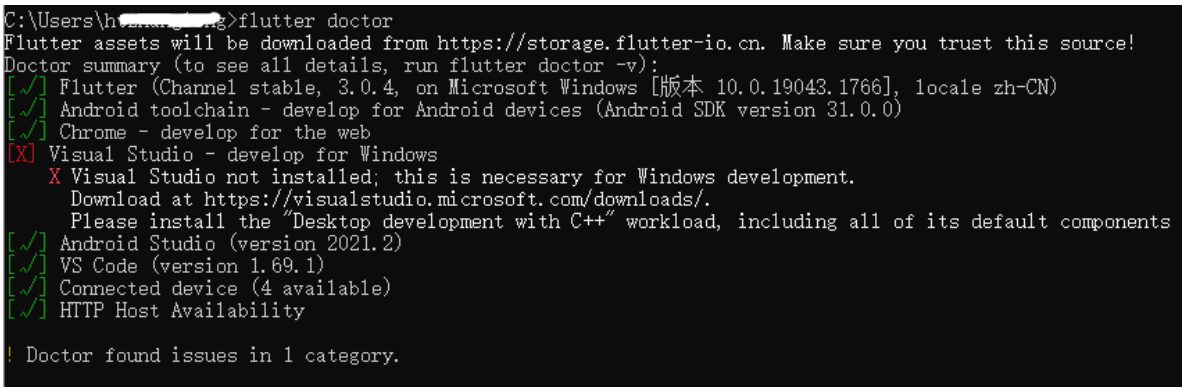


第一次执行可能会提示下面错误：

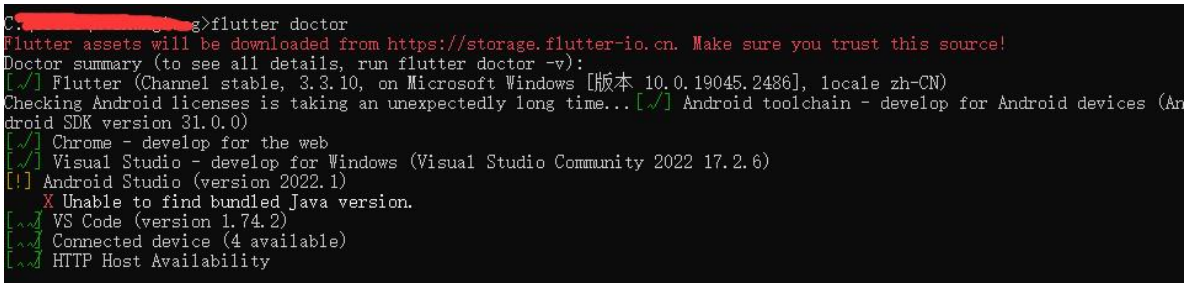
1、错误一：`cmdline-tools component is missing`



2、错误二：`Visual Studio not installed` 如果只是开发Flutter APP可以忽略此错误信息



3、错误三： Android Studio Unable to find bundled Java version.



Android Studio Unable to find bundled Java version 解决方法：

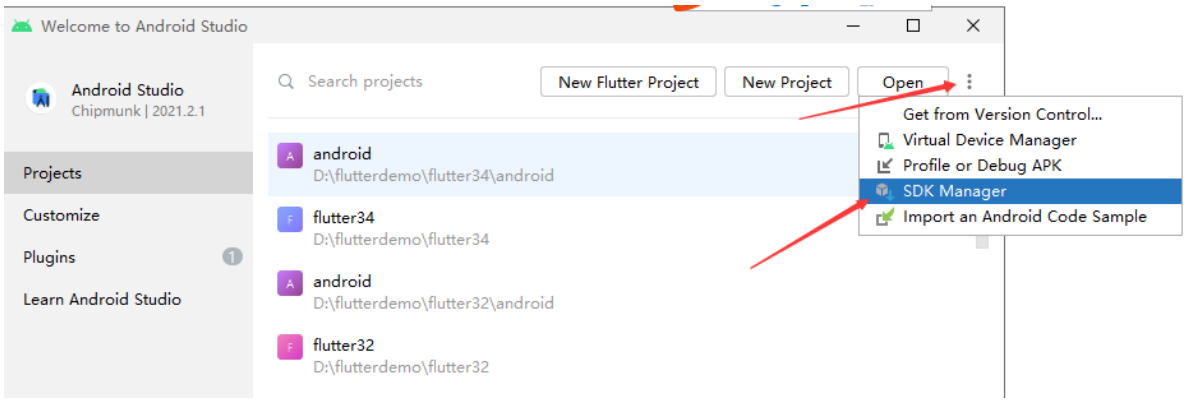
如果是macOS系统，在jbr同目录下创建一个jre目录，然后将jbr目录内的全部文件复制一份到jre目录下即可。

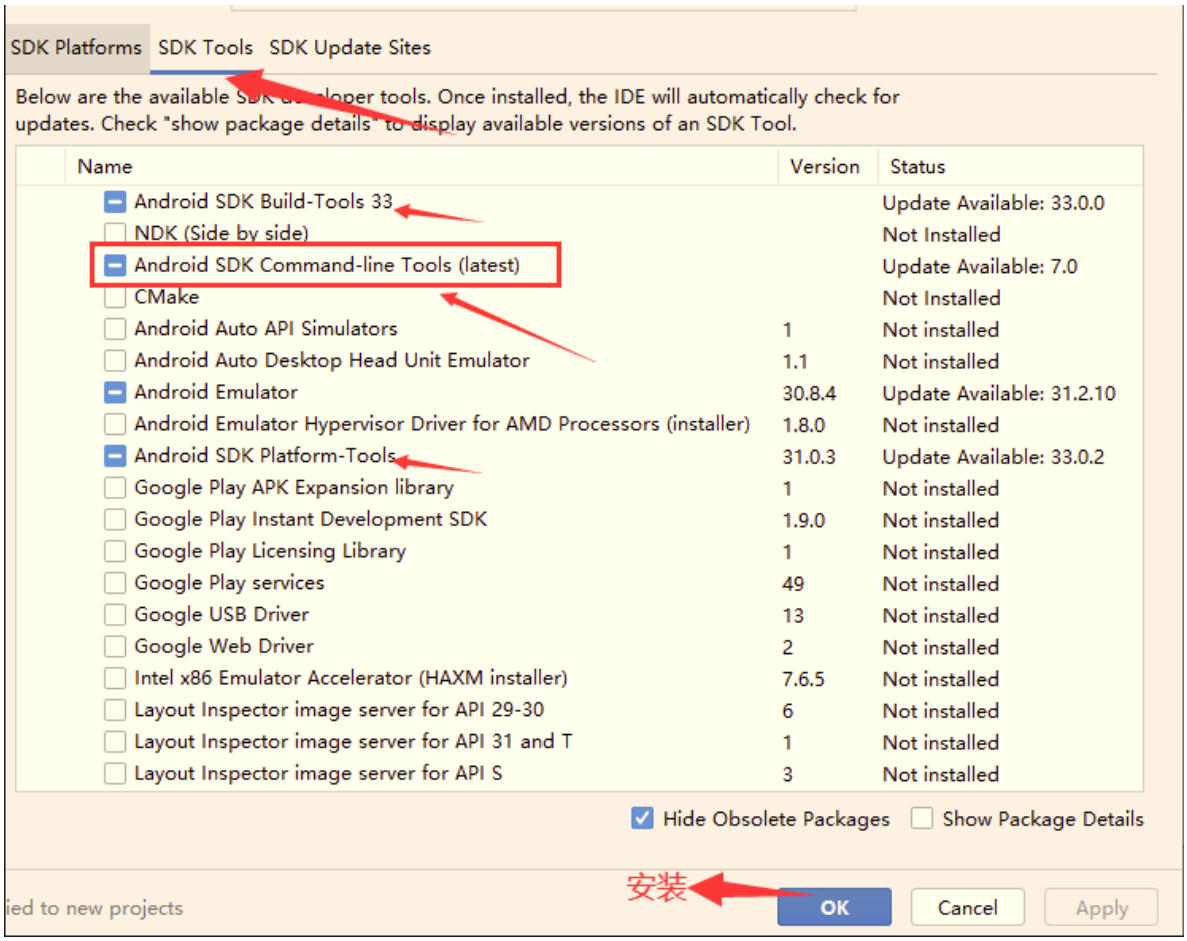
如果是Windows系统，jre目录是存在的，不过里面几乎没东西，可以直接将jbr目录内的全部文件复制一份到jre目录下即可

详情参考：http://bbs.itying.com/topic/63eafd840d0a6c0aecbe5436

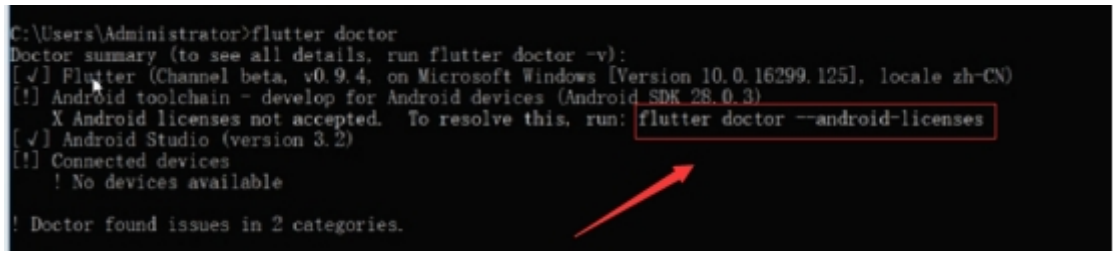
4、错误一的解决方法安装cmdline-tools 以及配置android-licenses：

4.1 安装cmdline-tools





4.2 配置android-licenses



这个时候复制上面红色框框内的命令

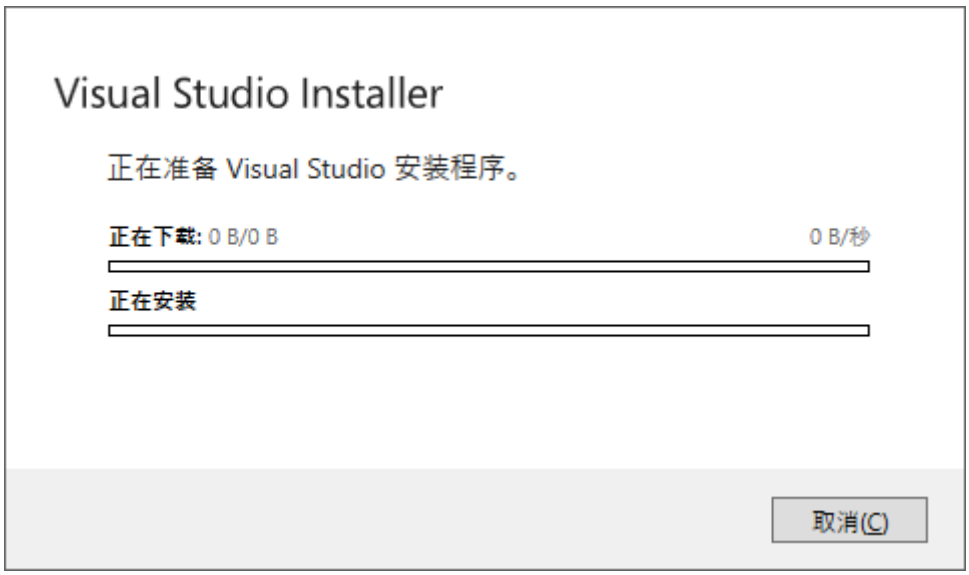
```
flutter doctor --android-licenses
```

注意：提示输入Y/N的地方全部输入Y

5、错误二的解决方法安装Visual Studio：

Visual Studio主要用于flutter 桌面软件开发，如果您只是开发flutter app可以不用安装Visual Studio

https://visualstudio.microsoft.com/zh-hans/downloads/



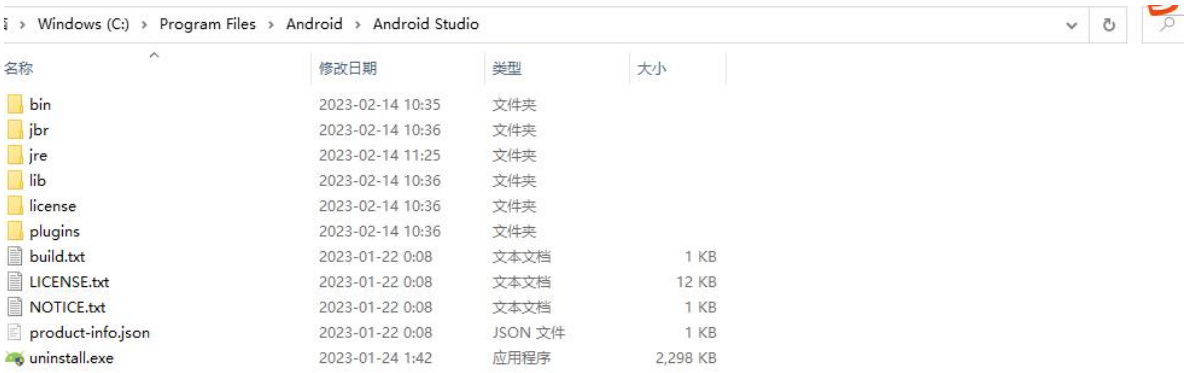
如果安装失败可以修改DNS尝试

6、错误三的解决方法

Android Studio Unable to find bundled Java version 解决方法：

如果是macOS系统，在jbr同目录下创建一个jre目录，然后将jbr目录内的全部文件复制一份到jre目录下即可。

如果是Windows系统，jre目录是存在的，不过里面几乎没东西，可以直接将jbr目录内的全部文件复制一份到jre目录下即可。

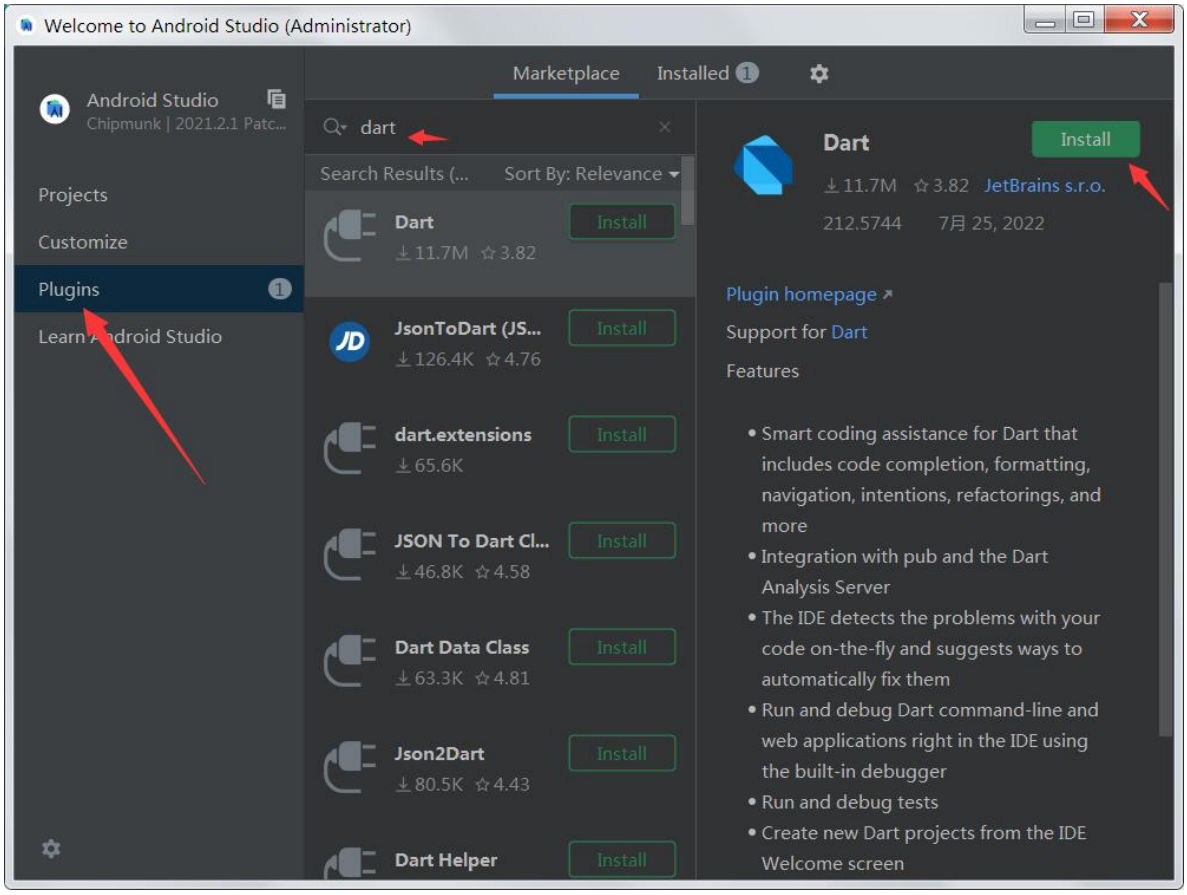


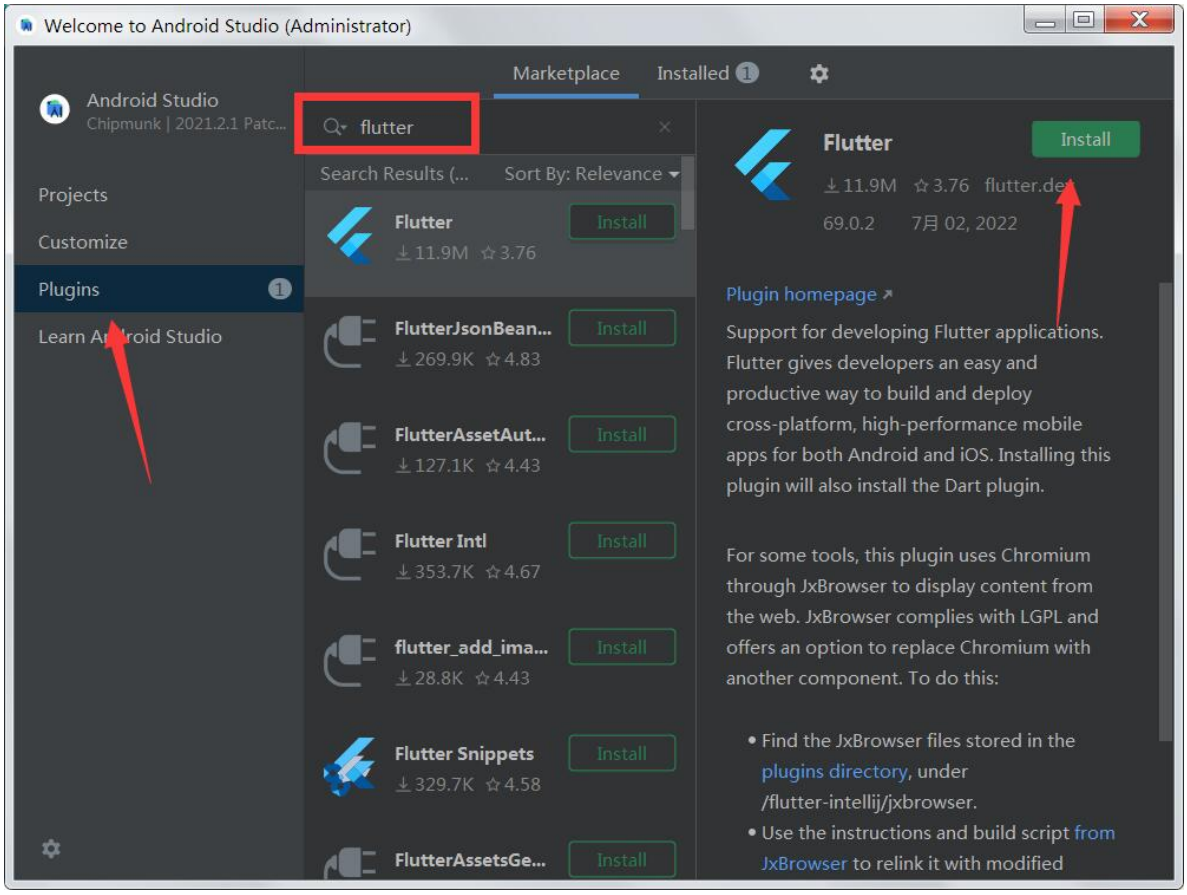
找到 Android Studio安装目录，把jbr里面的所有文件复制到jre

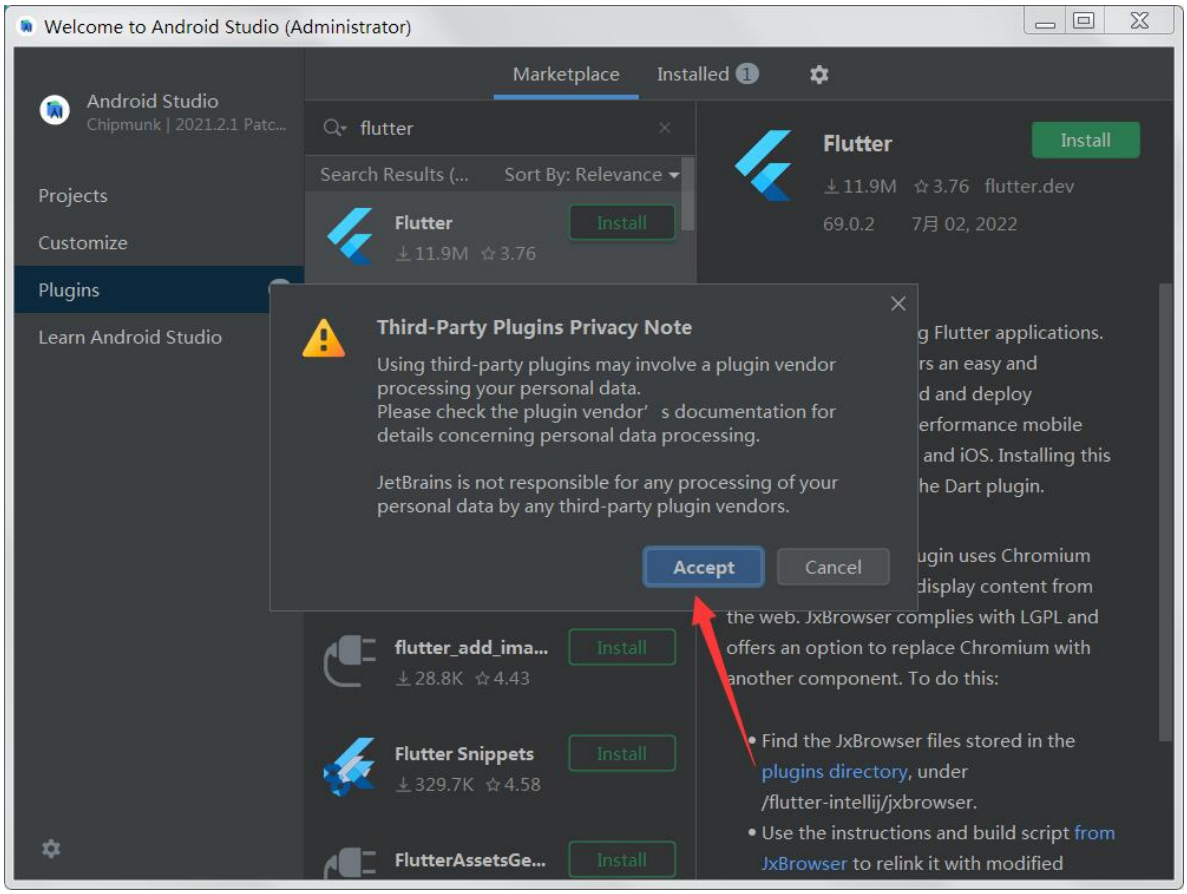
详情参考：http://bbs.itying.com/topic/63eafd840d0a6c0aecbe5436

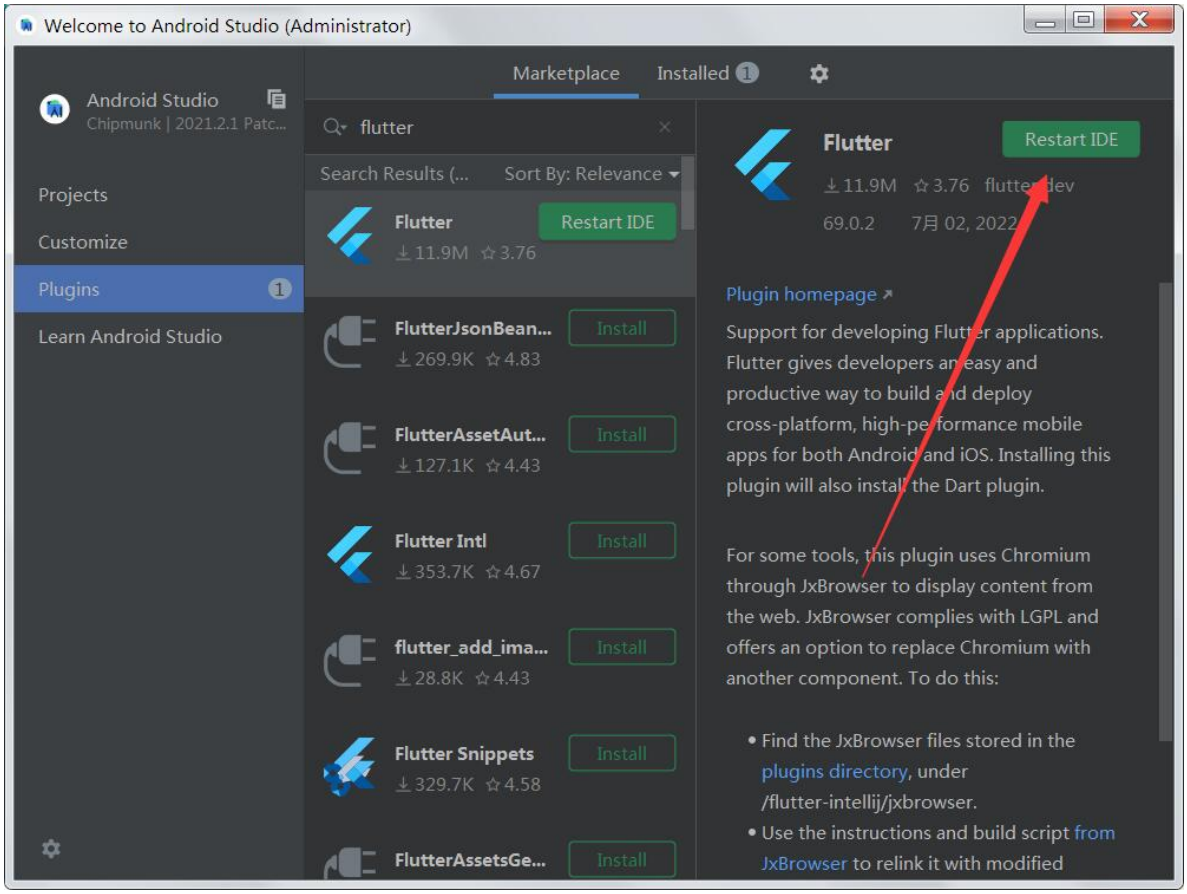
2.6、打开Android Studio 安装Flutter插件

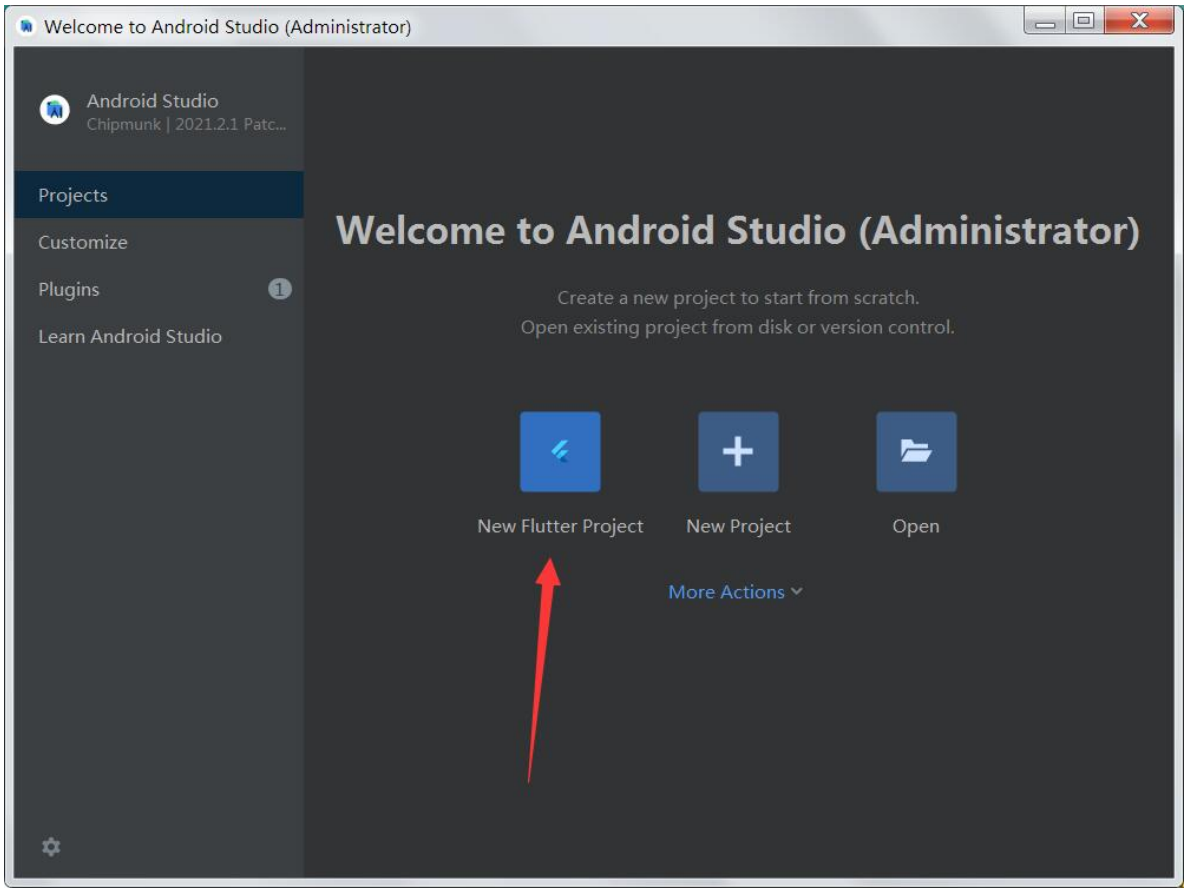
新版Android Studio配置



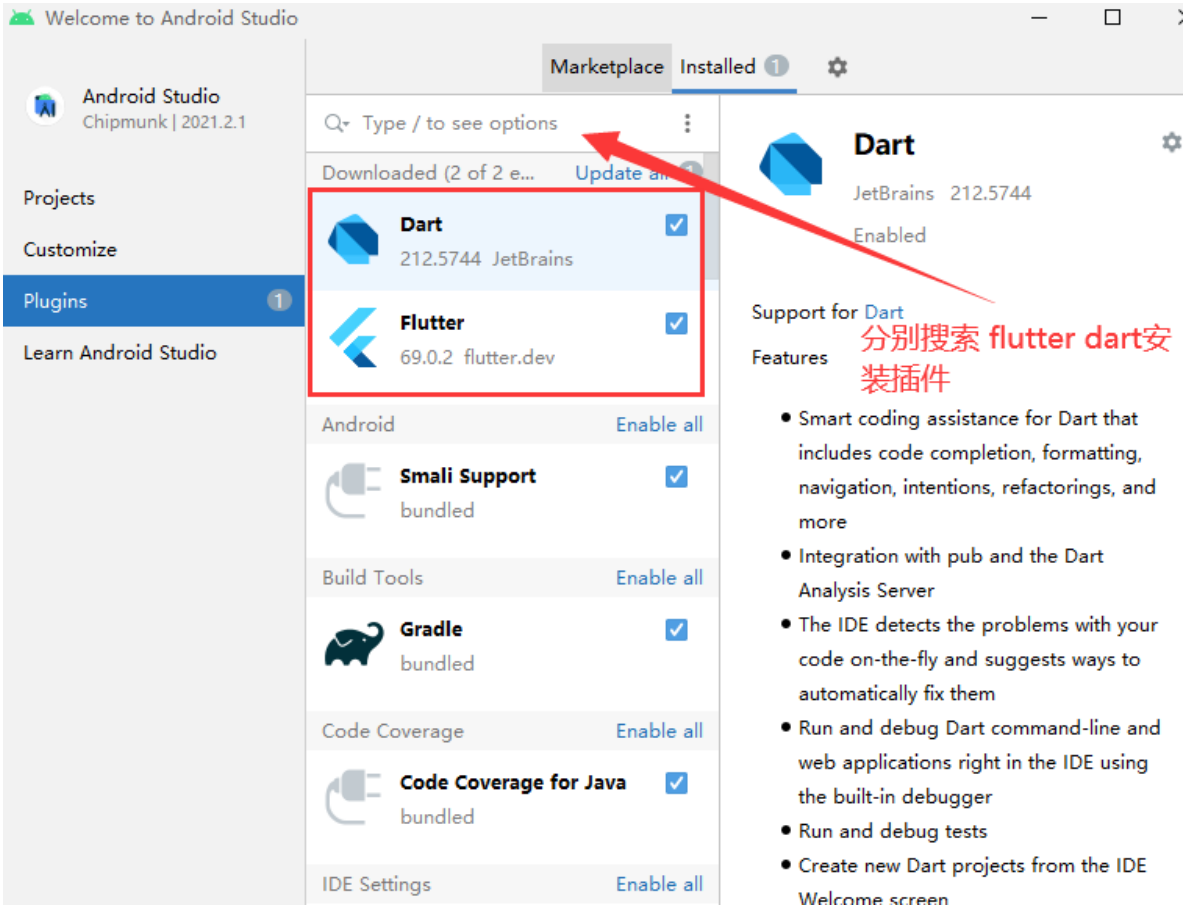

Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutter
Search Results (...
Sort By: Relevance
Flutter
Install
11.9M 3.76
FlutterJsonBean...
269.9K 4.83
FlutterAssetAut...
127.1K 4.43
Flutter Intl
353.7K 4.67
flutter_add_ima...
28.8K 4.43
Flutter Snippets
329.7K 4.58
FlutterAssetsGe...
Flutter
Install
11.9M 3.76 flutter.dev
69.0.2 7月 02, 2022
Plugin homepage
Support for developing Flutter applications. Flutter gives developers an easy and productive way to build and deploy cross-platform, high-performance mobile apps for both Android and iOS. Installing this plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium through JxBrowser to display content from the web. JxBrowser complies with LGPL and offers an option to replace Chromium with another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified

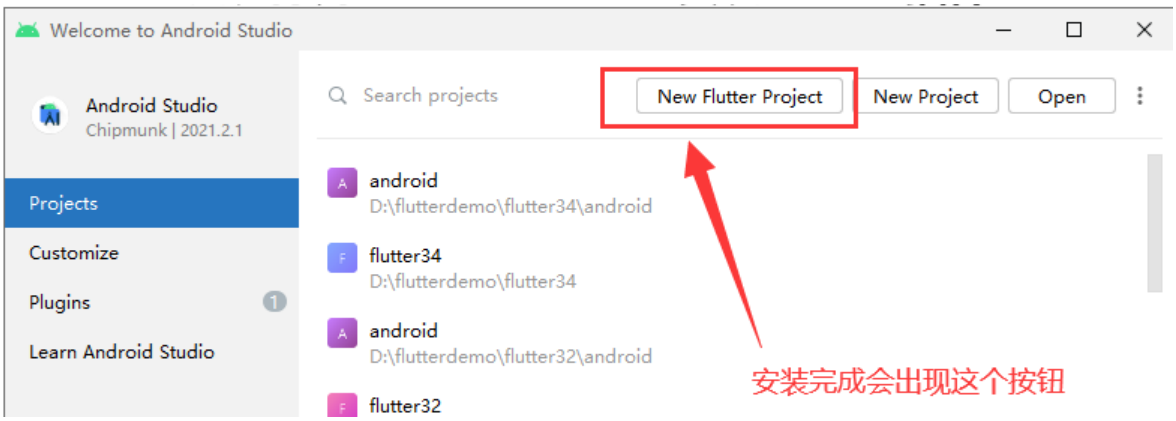

Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutter
Search Results (...
Sort By: Relevance
Flutter
Install
11.9M 3.76
Third-Party Plugins Privacy Note
Using third-party plugins may involve a plugin vendor processing your personal data.
Please check the plugin vendor' s documentation for details concerning personal data processing.
JetBrains is not responsible for any processing of your personal data by any third-party plugin vendors.
Accept
Cancel
flutter_add_ima...
28.8K 4.43
Flutter Snippets
329.7K 4.58
FlutterAssetsGe...
Flutter
Install
11.9M 3.76 flutter.dev
69.0.2 7月 02, 2022
g Flutter applications.
rs an easy and
d and deploy
erformance mobile
and iOS. Installing this
he Dart plugin.
ugin uses Chromium
display content from
the web. JxBrowser complies with LGPL and
offers an option to replace Chromium with
another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified



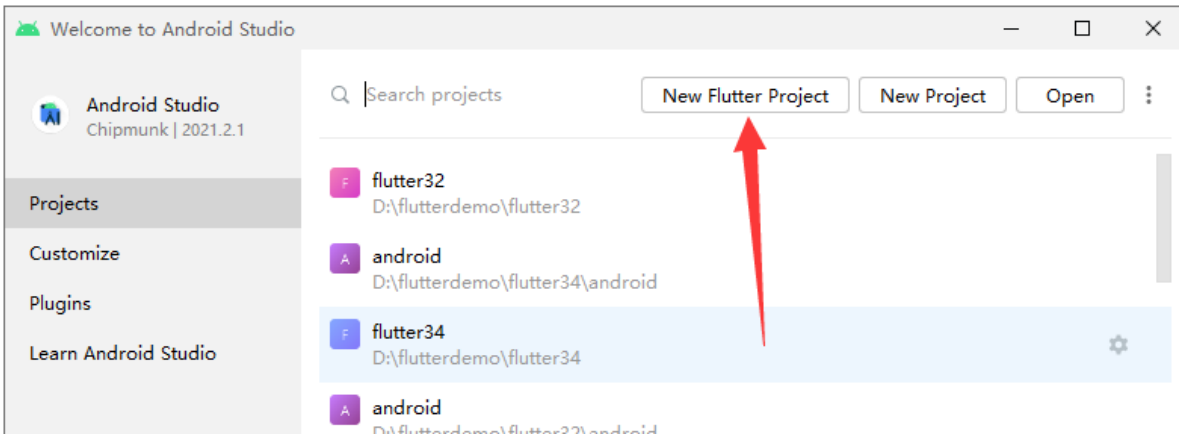


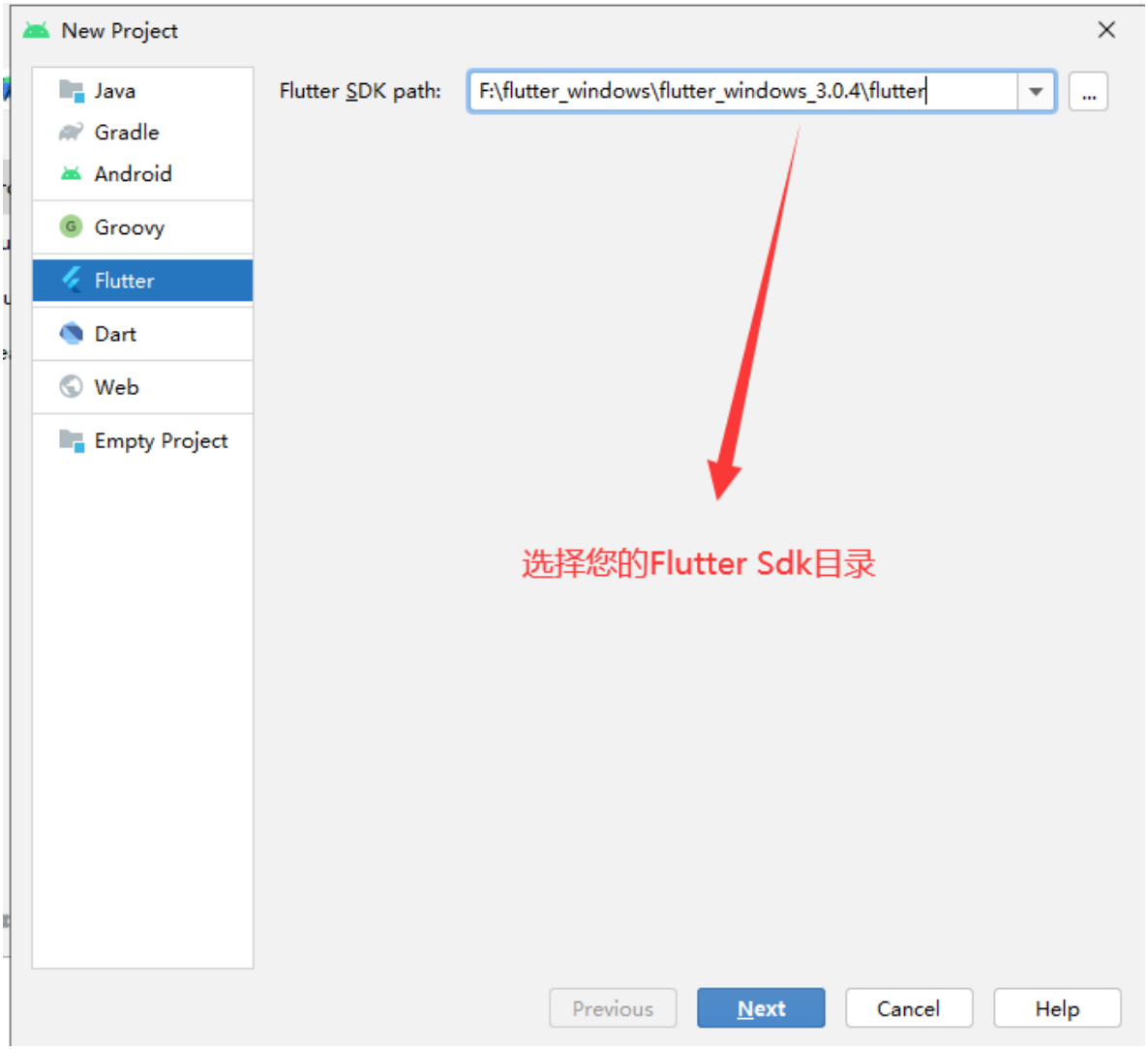
提示：新版本android studio也可能是下面界面

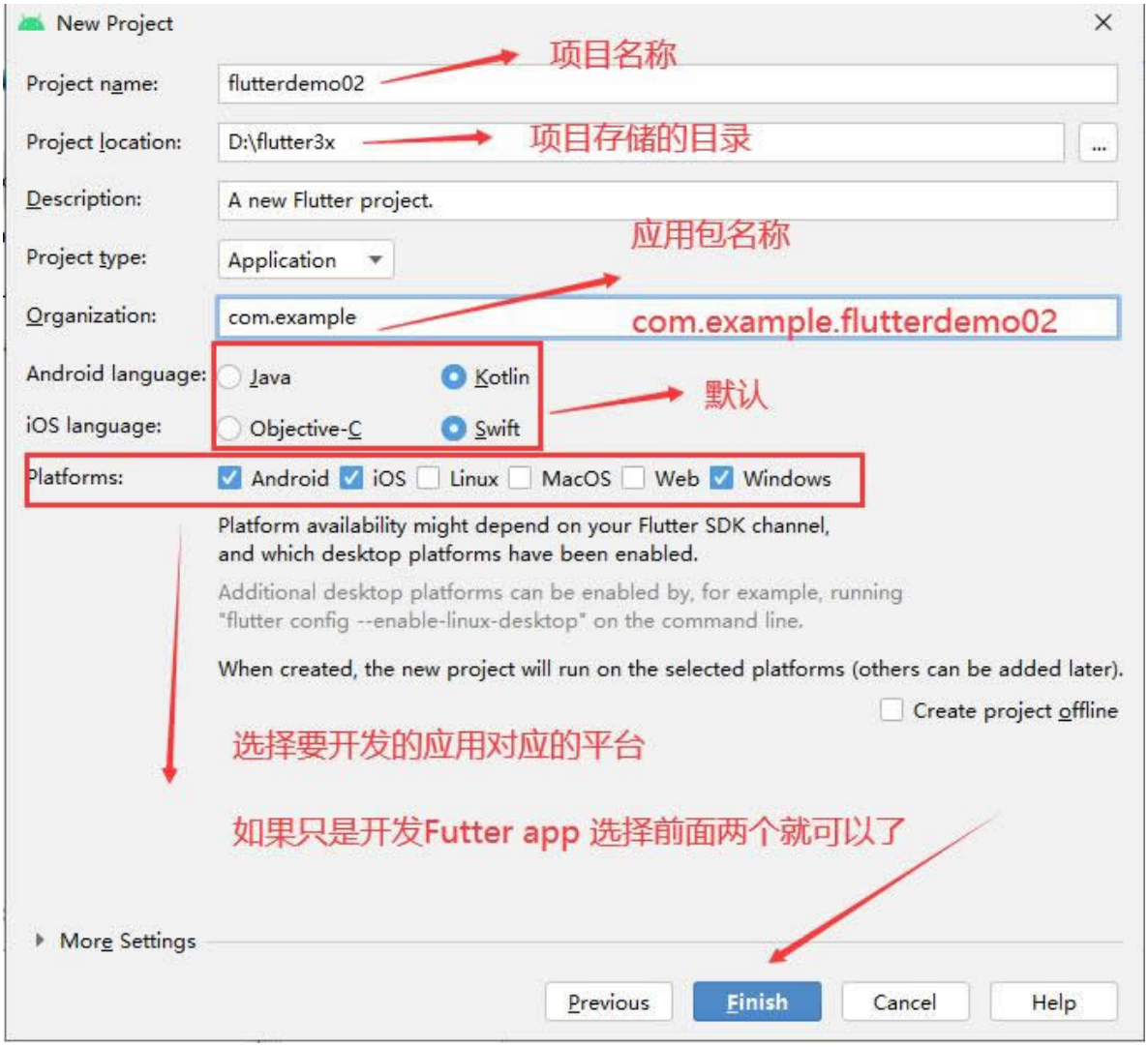


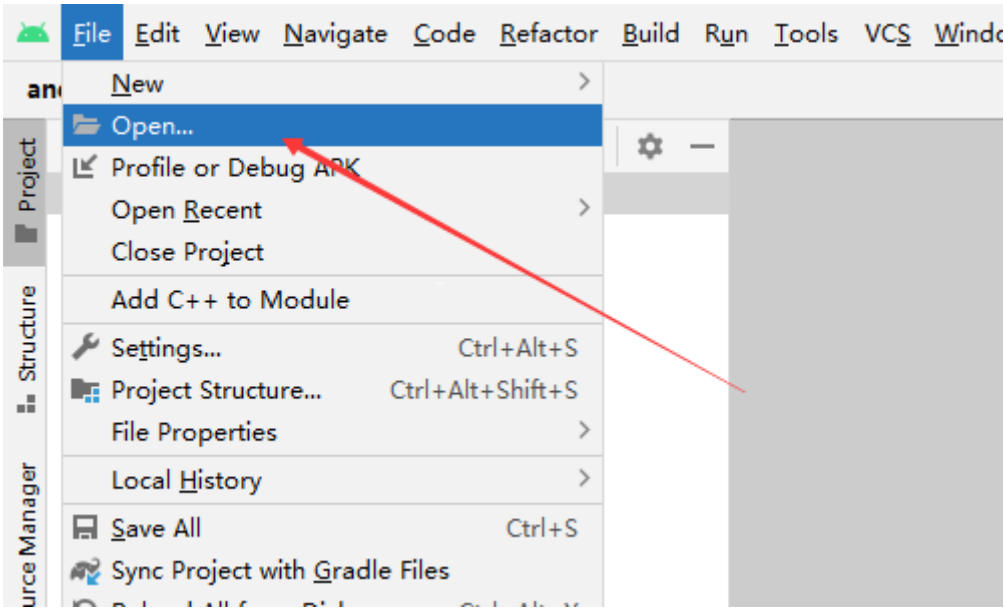


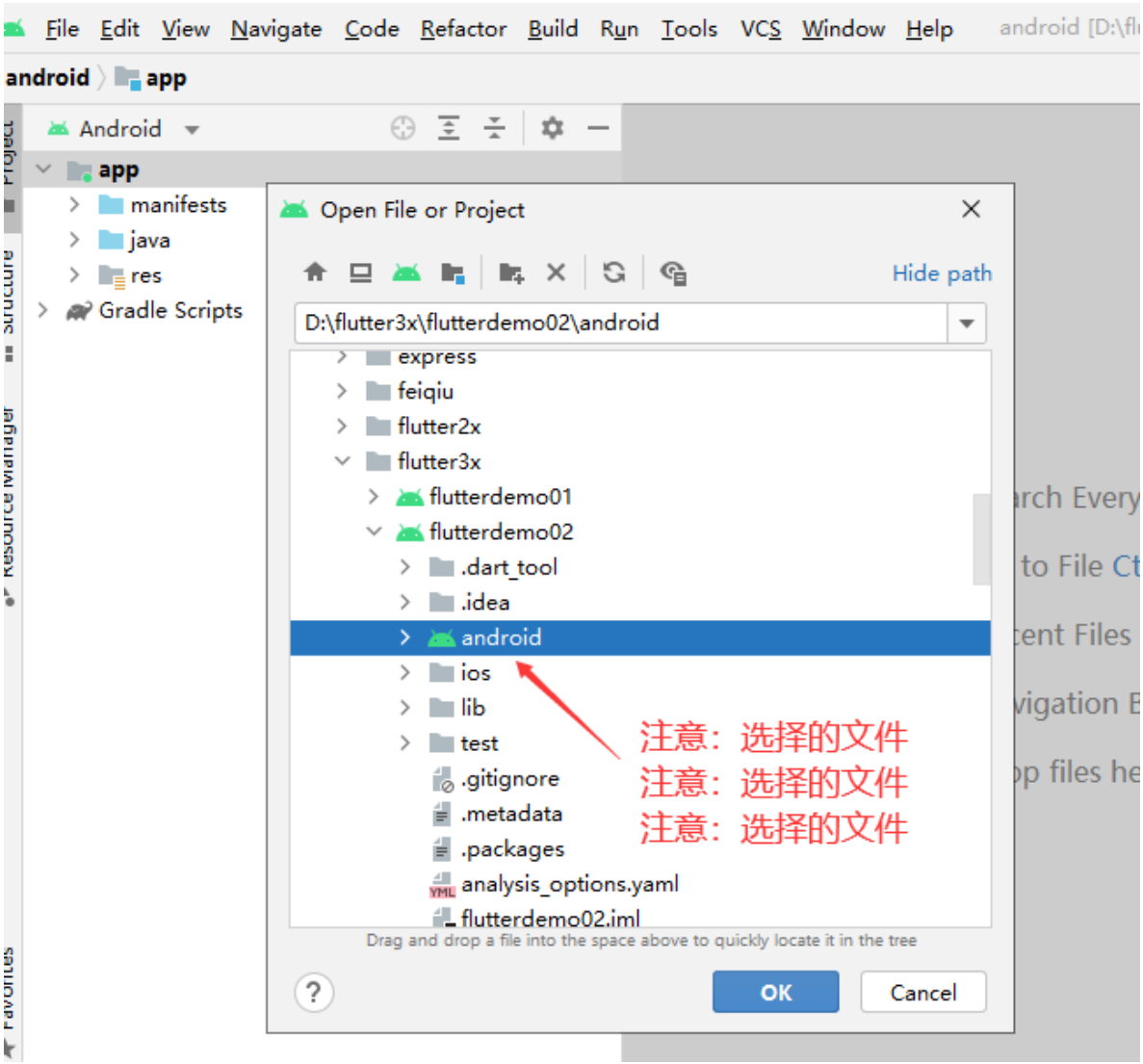
2.7、创建 运行Flutter项目

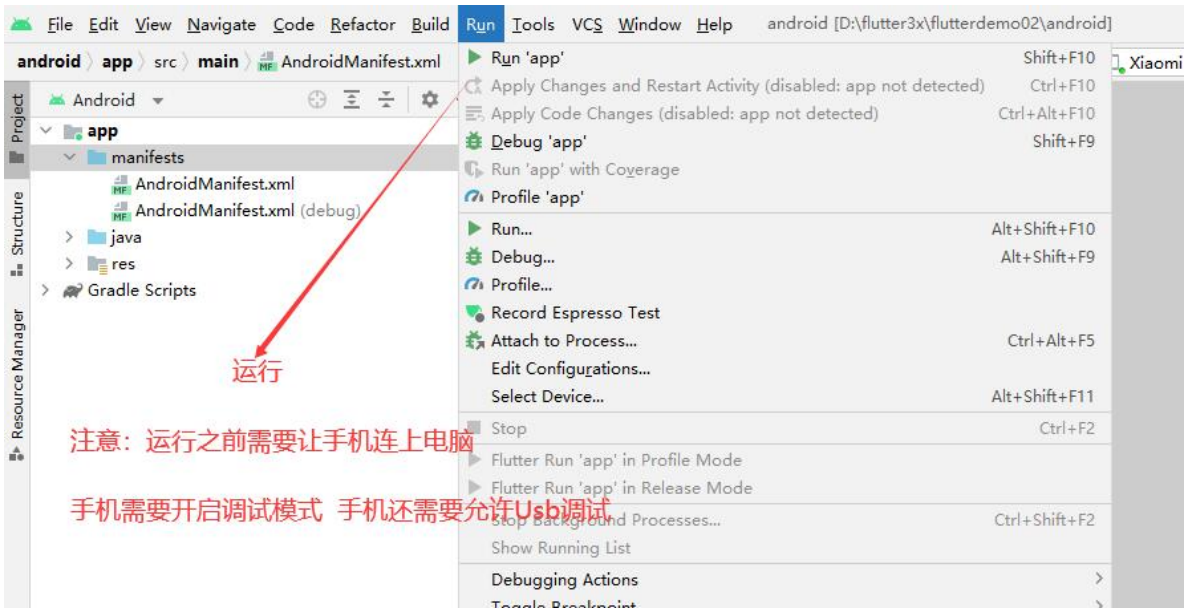

Welcome to Android Studio
Android Studio
Chipmunk | 2021.2.1
Projects
Customize
Plugins
Learn Android Studio
Search projects
New Flutter Project
New Project
Open
flutter32
D:\flutterdemo\flutter32
android
D:\flutterdemo\flutter34\android
flutter34
D:\flutterdemo\flutter34
android


New Project
Java
Gradle
Android
Groovy
Flutter
Dart
Web
Empty Project
Flutter SDK path:
F:\flutter_windows\flutter_windows_3.0.4\flutter
...
选择您的Flutter Sdk目录
Previous
Next
Cancel
Help


New Project
项目名称
Project name:
flutterdemo02
项目存储的目录
Project location:
D:\flutter3x
Description:
A new Flutter project.
Project type:
Application
应用包名称
Organization:
com.example
com.example.flutterdemo02
Android language:
Java
Kotlin
默认
iOS language:
Objective-C
Swift
Platforms:
Android
iOS
Linux
MacOS
Web
Windows
Platform availability might depend on your Flutter SDK channel,
and which desktop platforms have been enabled.
Additional desktop platforms can be enabled by, for example, running
"flutter config --enable-linux-desktop" on the command line.
When created, the new project will run on the selected platforms (others can be added later).
Create project offline
选择要开发的应用对应的平台
如果只是开发Futter app 选择前面两个就可以了
More Settings
Previous
Finish
Cancel
Help


File
Edit
View
Navigate
Code
Refactor
Build
Run
Tools
VCS
New
Open...
Profile or Debug APK
Open Recent
Close Project
Add C++ to Module
Settings...
Ctrl+Alt+S
Project Structure...
Ctrl+Alt+Shift+S
File Properties
Local History
Save All
Ctrl+S
Sync Project with Gradle Files
Project
Structure



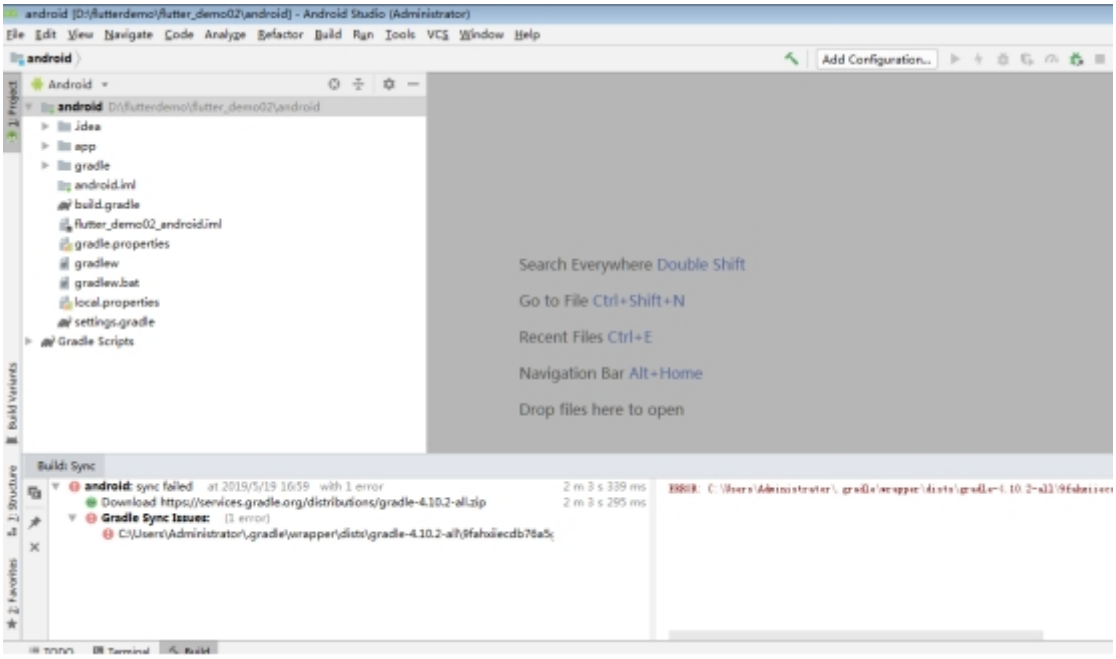


通过 `flutter devices` 可以检测检测可用的设备

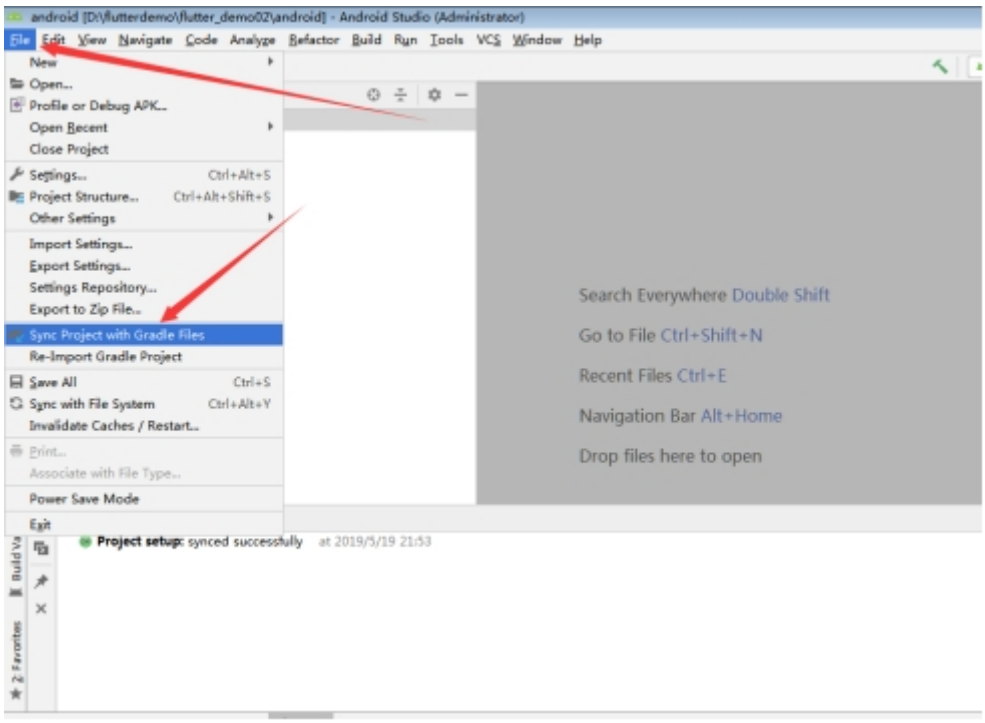
```
C:\Users\itying>flutter devices

4 connected devices:
Redmi K30 (mobile) • beac2700 • android-arm64  • Android 12 (API 31)
Windows (desktop)  • windows  • windows-x64    • Microsoft Windows [版本
10.0.19043.1766]
Chrome (web)       • chrome   • web-javascript • Google Chrome 103.0.5060.114
Edge (web)         • edge     • web-javascript • Microsoft Edge 103.0.1264.49
```

2.8、可能遇到的错误

1、导入项目提示Gradle相关错误



如果报错点击`Flie->Sync Project With Gradle Files` 重新下载Gradle ,这个过程比较慢**10-30分钟**左右。



按照上面方法重试后Gradle 还是失败的解决方法

1、开启手机热点重试

2、谷歌或者百度搜索 “android Gradle编译时下载依赖失败”

2、找不到运行的设备

参考下一讲真机调试，下一讲会详细讲解...

二、Windows上面搭建Flutter Android运行环境（Flutter3.3.10之前的版本）

Flutter Android环境搭建：

- 电脑上面安装配置JDK （**版本** jdk-8u341-windows-x64）
- 电脑上下载安装Android Studio （**版本** android-studio-2021.2.1.14）
- 电脑上面下载配置Flutter Sdk （**版本** Flutter3.3.10以及之前的版本）
- 电脑上配置Flutter国内镜像
- 运行 flutter doctor命令检测环境是否配置成功，根据提示配置安装对应软件
- 打开Android Studio 安装Flutter插件
- 创建运行Flutter项目

2.1、 电脑上面安装配置JDK

1、下载安装JDK （8u361版本）

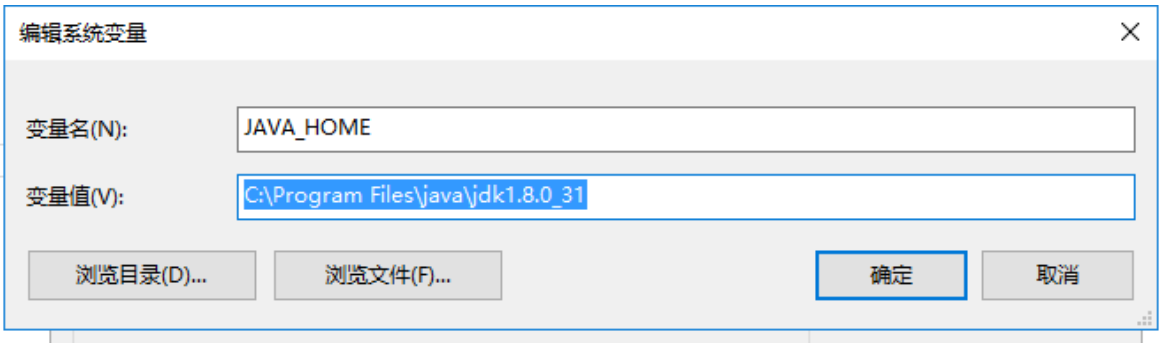
https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Linux macOS Solaris Windows

Product/file description	File size	Download
x86 Installer	159.21 MB	jdk-8u333-windows-i586.exe
x64 Installer	172.66 MB	jdk-8u333-windows-x64.exe

2、配置JDK

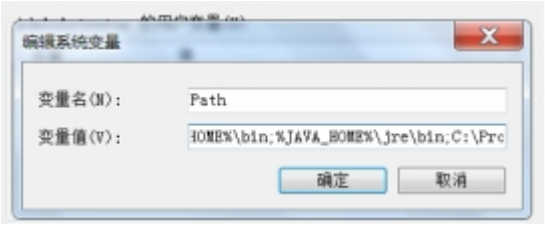
1、**系统变量** 里面新增JAVA_HOME，设置值为java sdk 根目录：



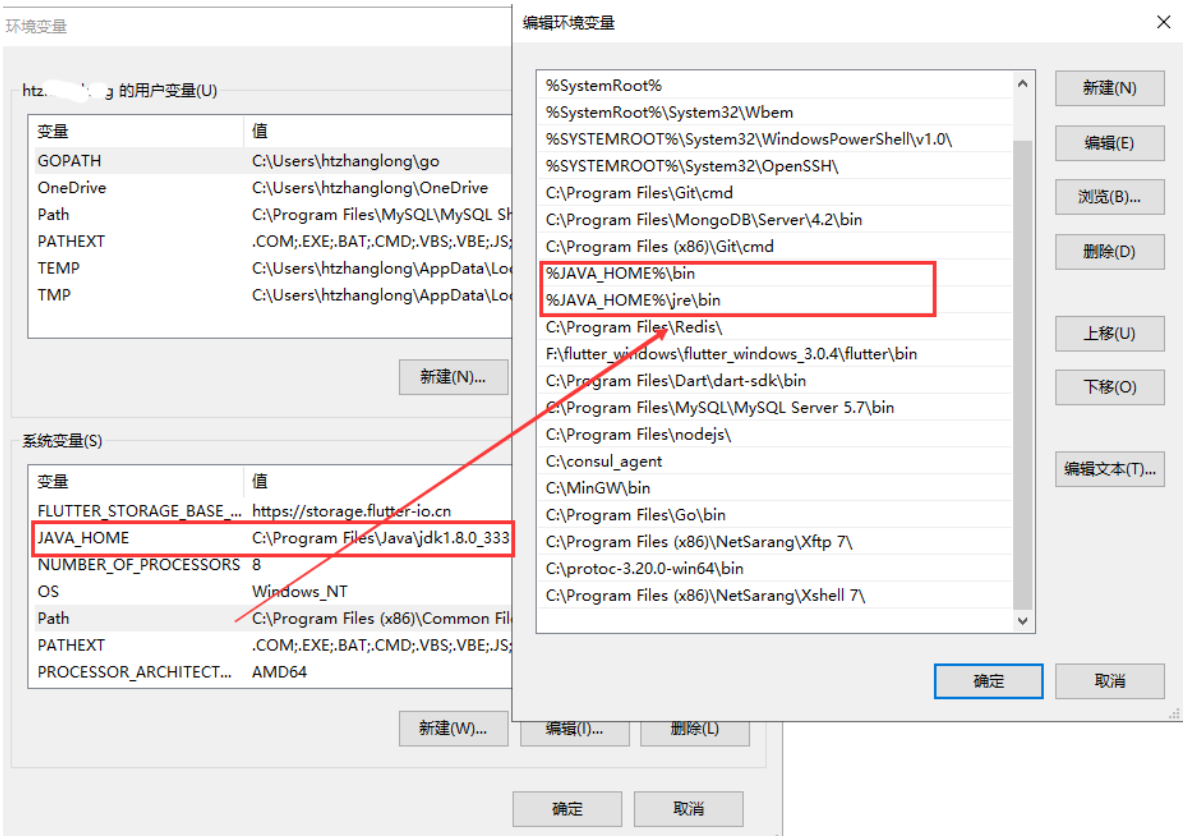
2、**系统变量** 找到Path 在Path环境变量里面增加如下代码

```
;%JAVA_HOME%\bin;%JAVA_HOME%\jre\bin
```

win7 注意环境变量之间的;（英文分号）



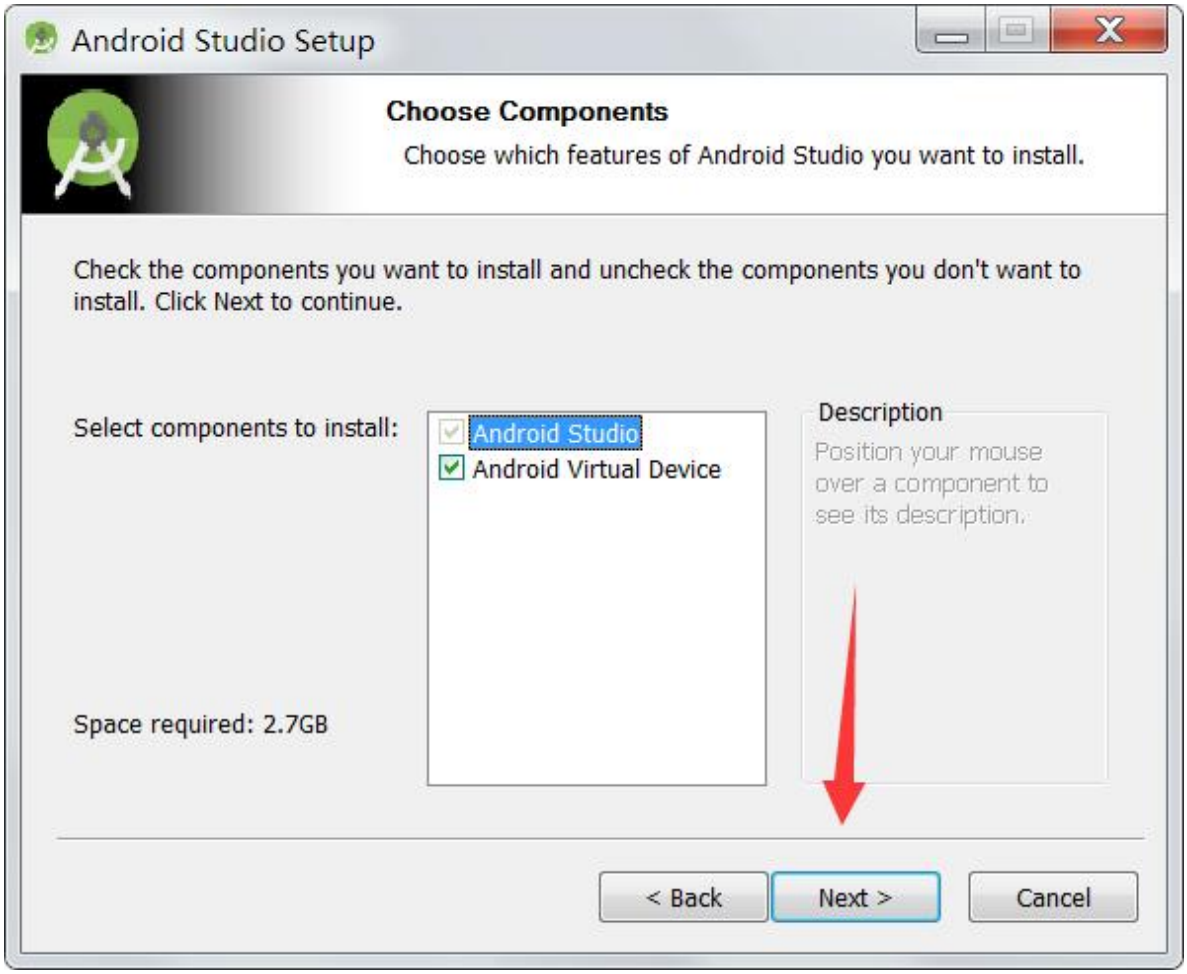
win10、win11中的配置

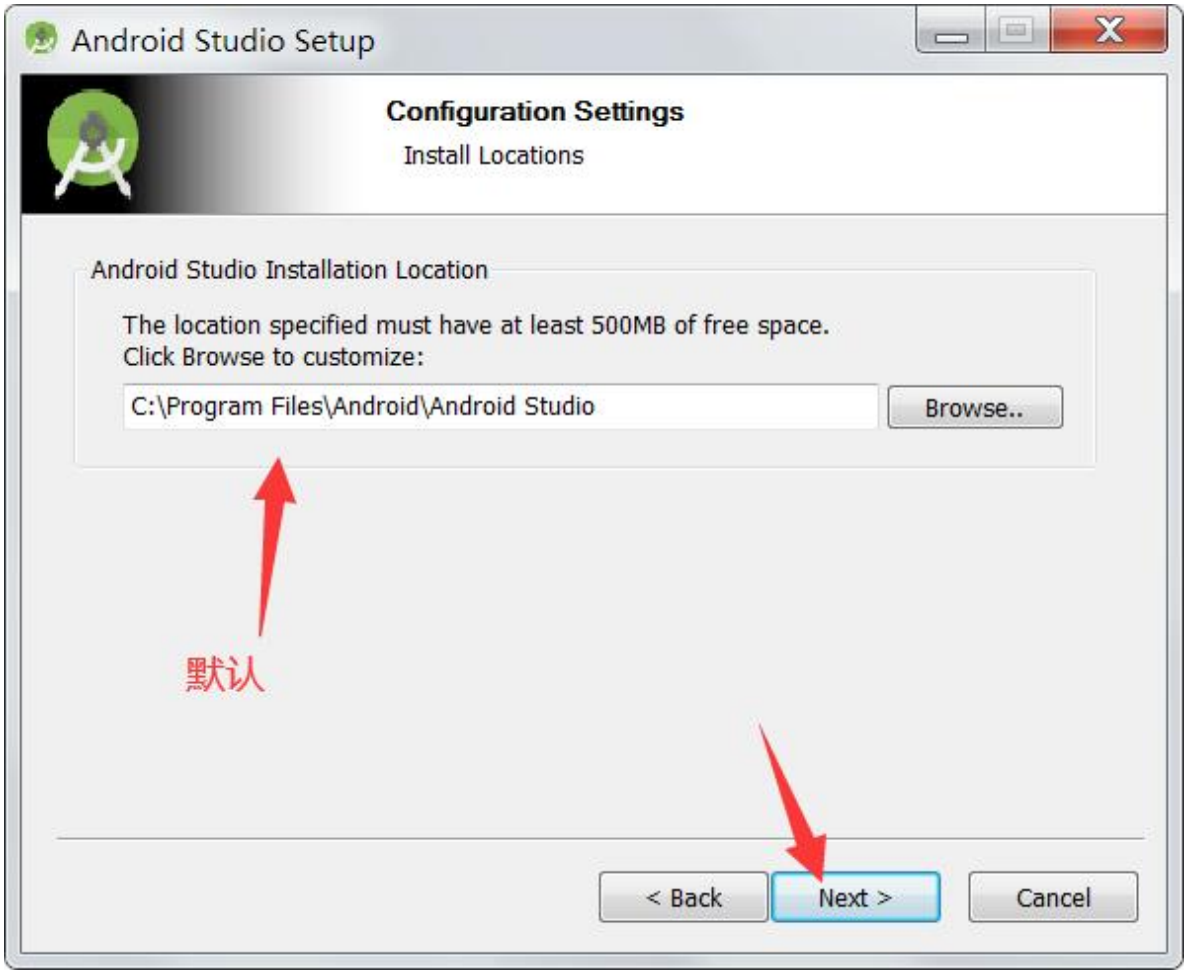


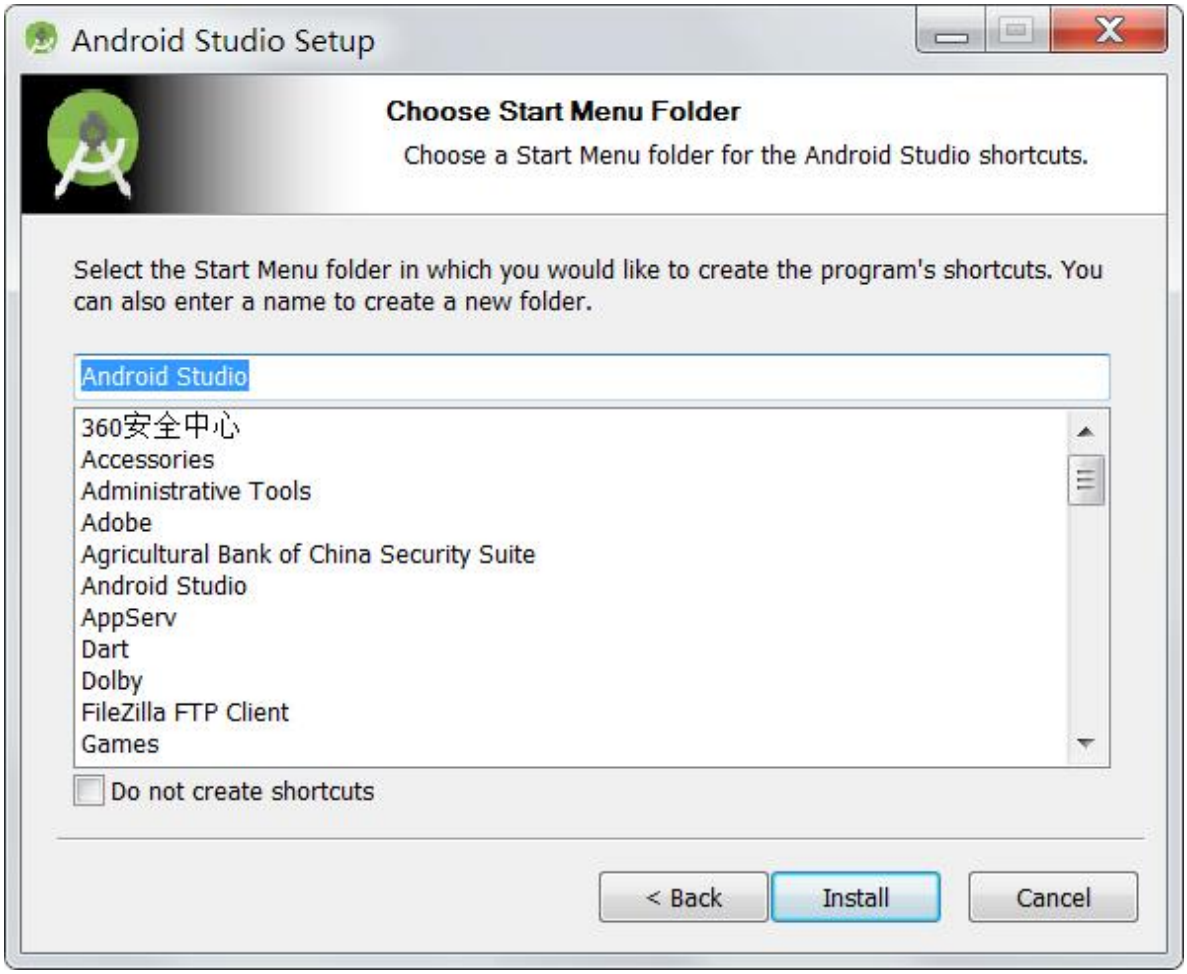
2.2 、电脑上下载安装Android Studio

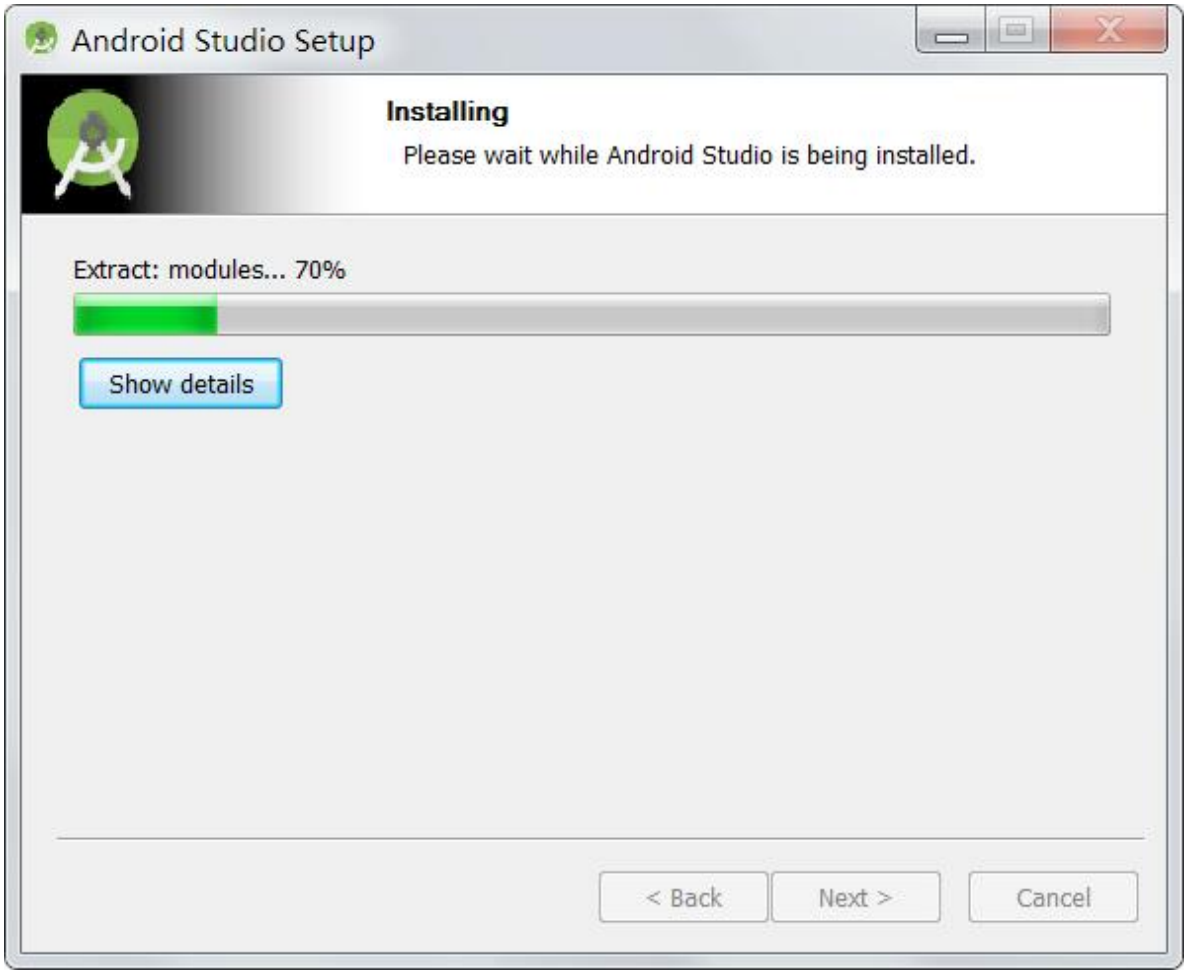
https://developer.android.google.cn/studio

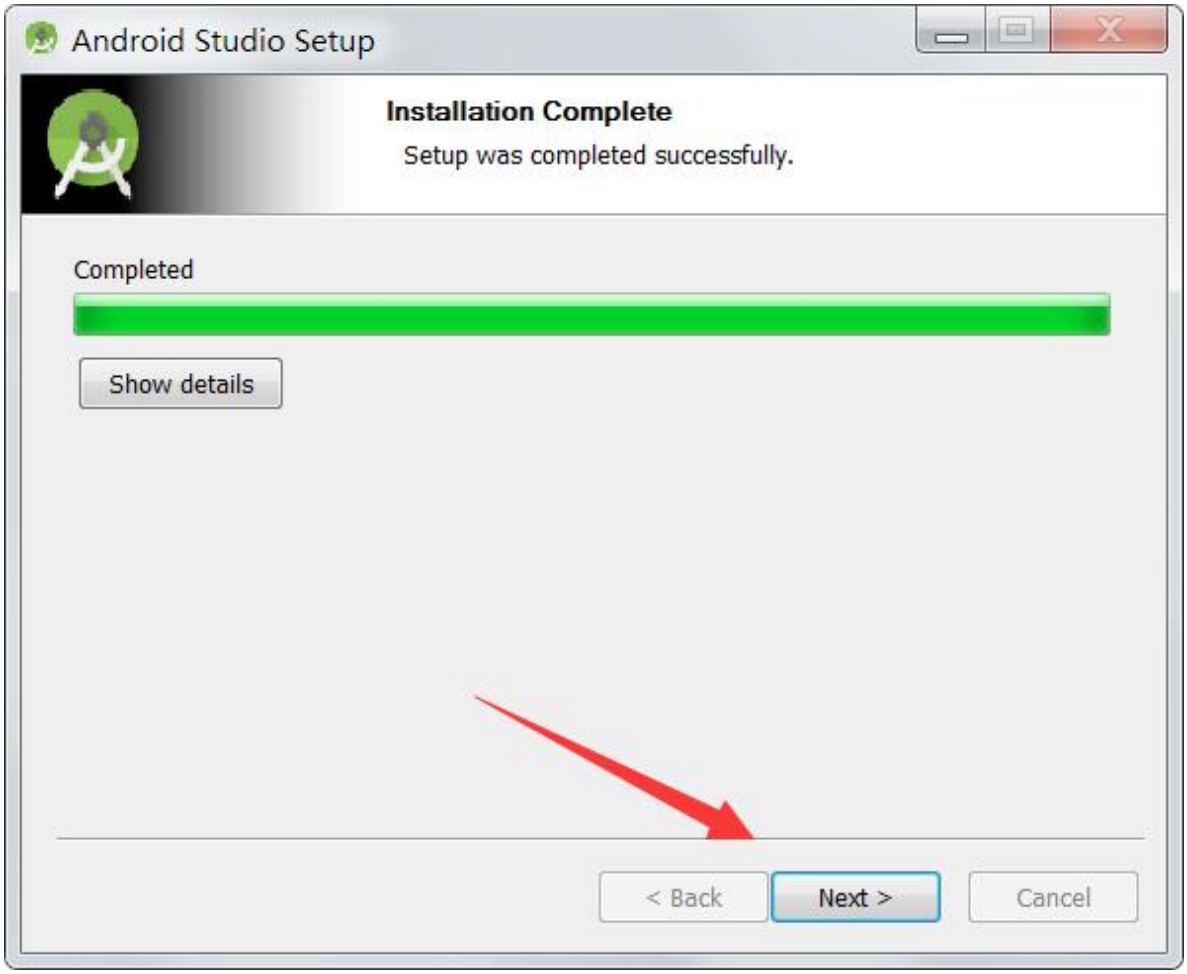

Android Studio Setup
Welcome to Android Studio Setup
Setup will guide you through the installation of Android Studio.
It is recommended that you close all other applications before starting Setup. This will make it possible to update relevant system files without having to reboot your computer.
Click Next to continue.
Android Studio
< Back
Next >
Cancel

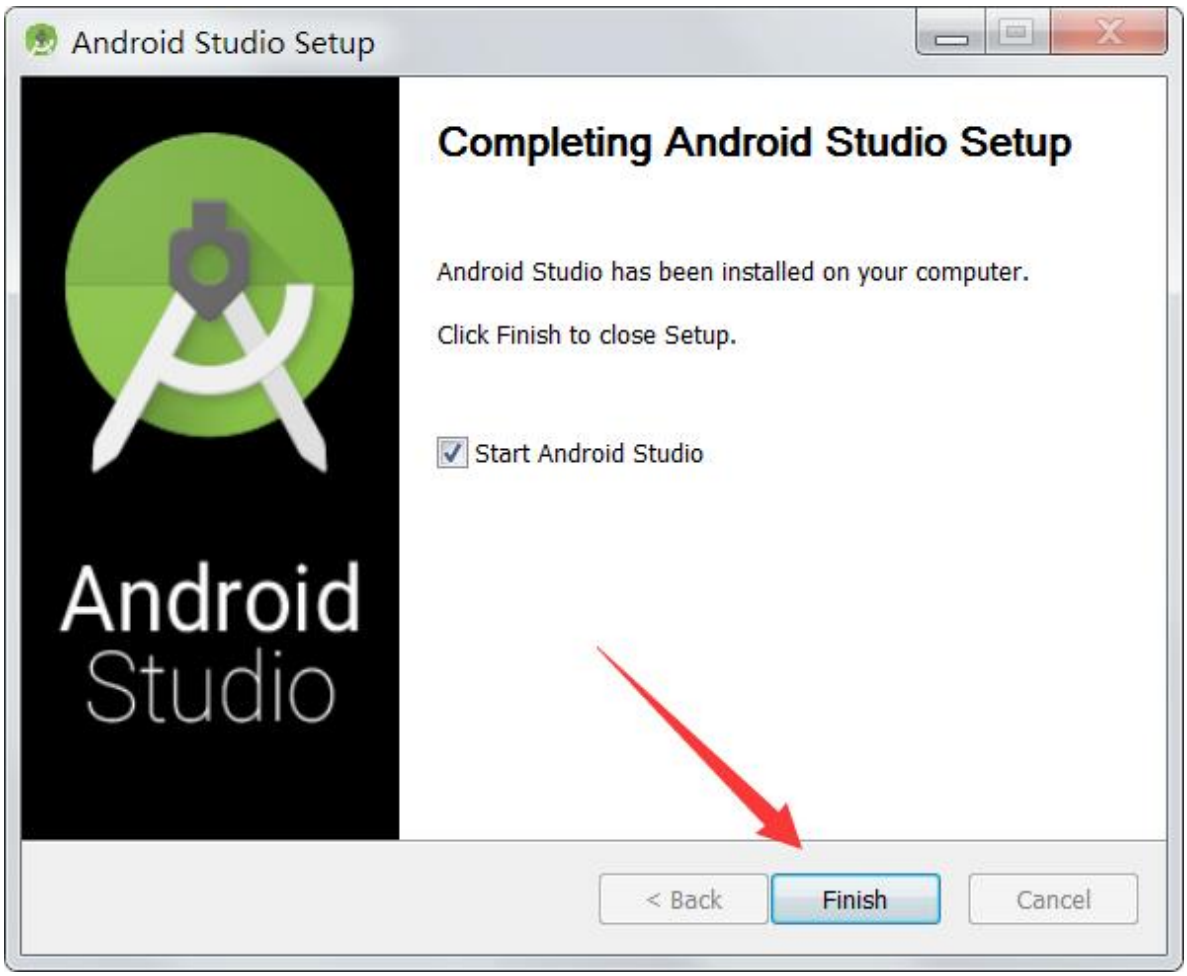

Android Studio Setup
Choose Components
Choose which features of Android Studio you want to install.
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.
Select components to install:
Android Studio
Android Virtual Device
Description
Position your mouse over a component to see its description.
Space required: 2.7GB
< Back
Next >
Cancel

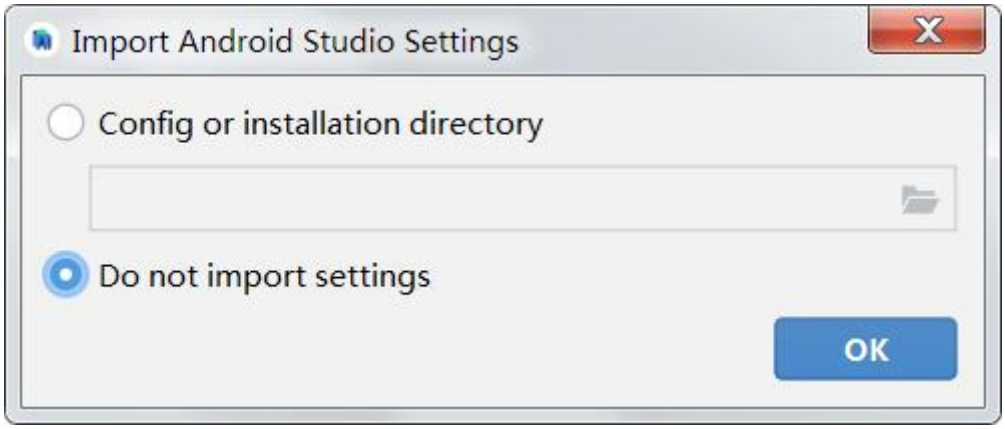

Android Studio Setup
Configuration Settings
Install Locations
Android Studio Installation Location
The location specified must have at least 500MB of free space.
Click Browse to customize:
C:\Program Files\Android\Android Studio
Browse..
默认
< Back
Next >
Cancel

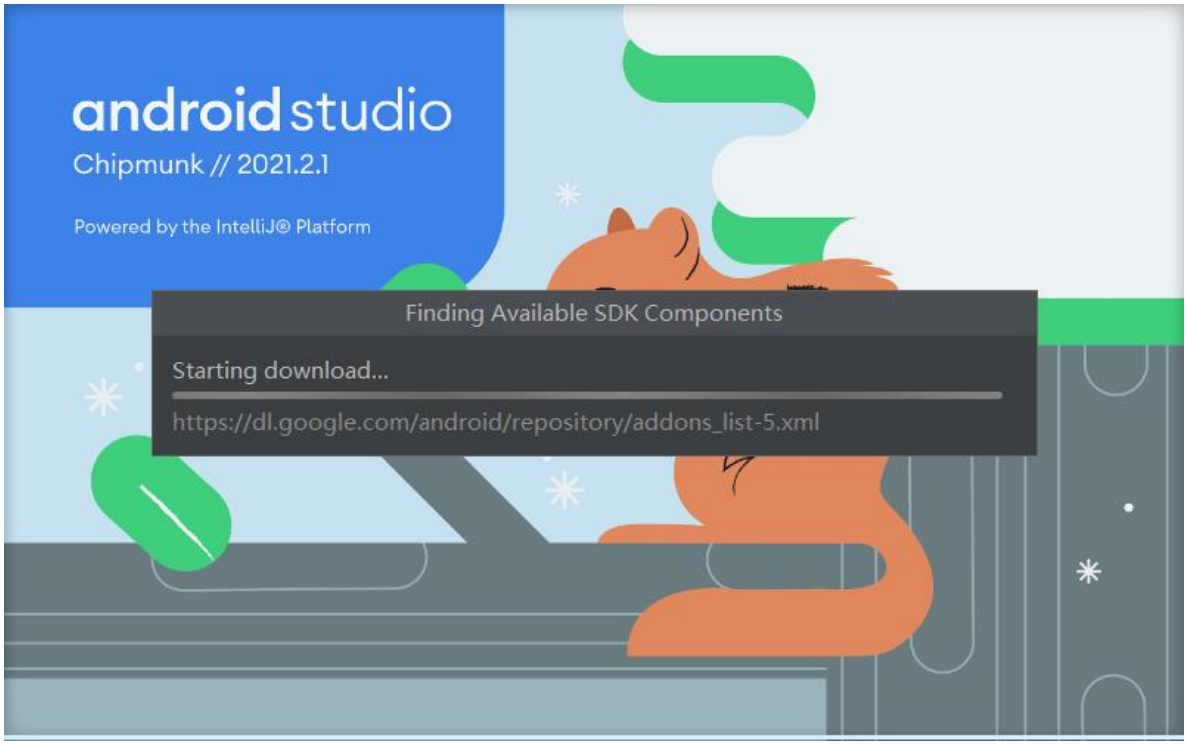

Android Studio Setup
Choose Start Menu Folder
Choose a Start Menu folder for the Android Studio shortcuts.
Select the Start Menu folder in which you would like to create the program's shortcuts. You can also enter a name to create a new folder.
Android Studio
360安全中心
Accessories
Administrative Tools
Adobe
Agricultural Bank of China Security Suite
Android Studio
AppServ
Dart
Dolby
FileZilla FTP Client
Games
Do not create shortcuts
< Back
Install
Cancel

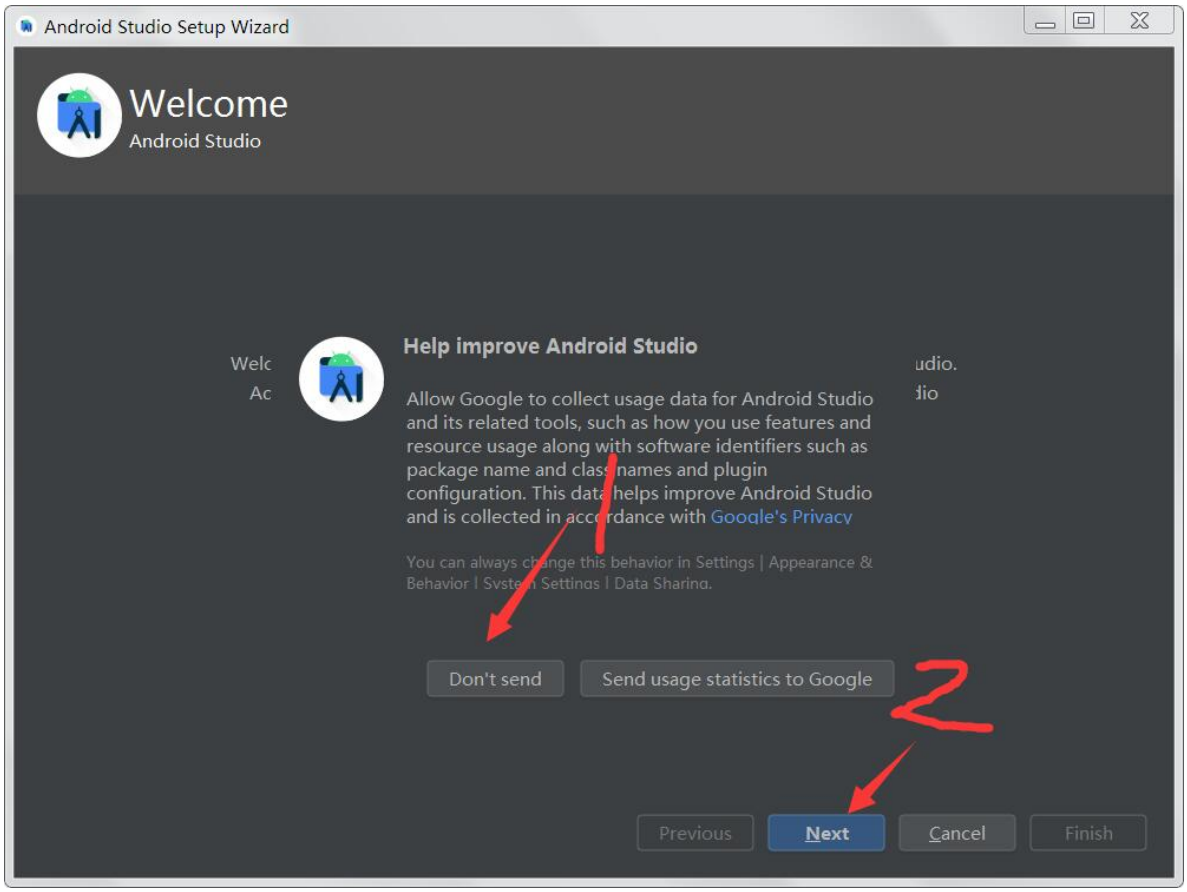

Android Studio Setup
Installing
Please wait while Android Studio is being installed.
Extract: modules... 70%
Show details
< Back
Next >
Cancel

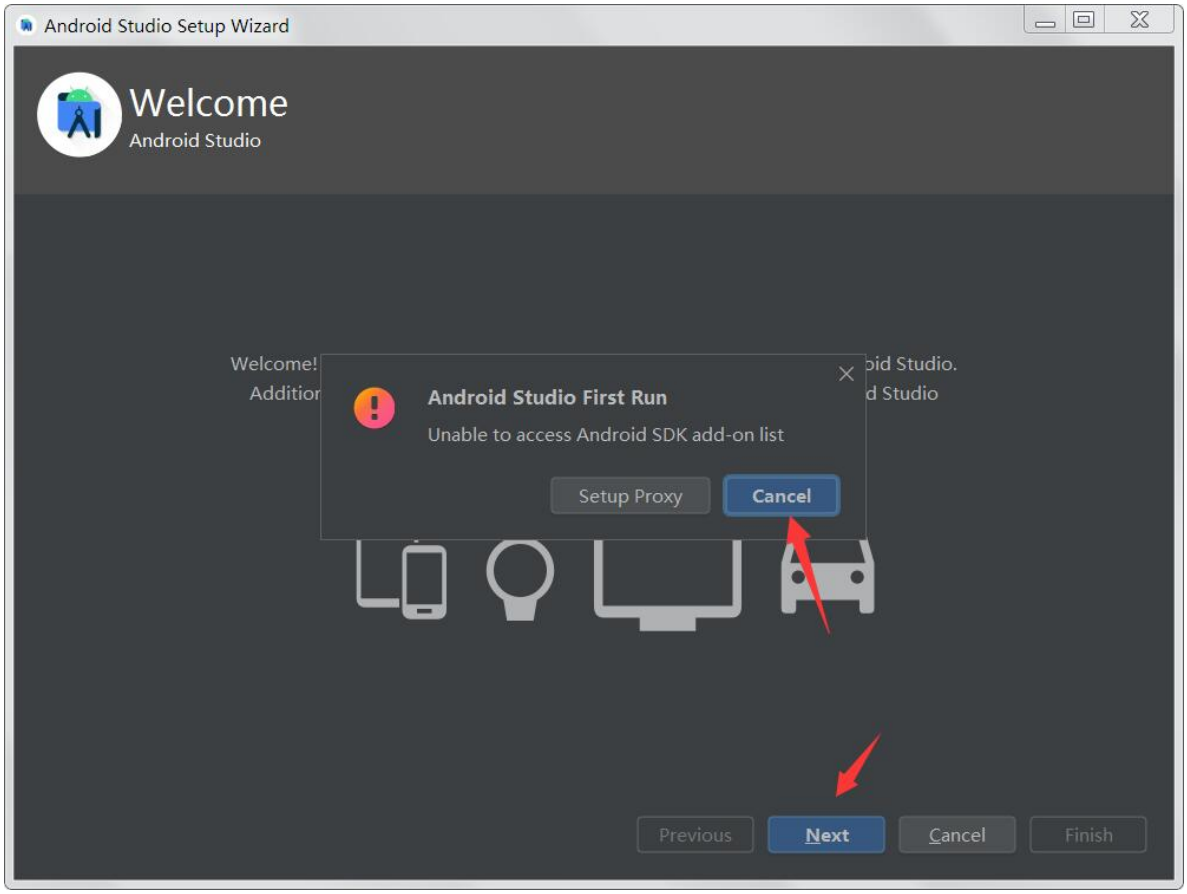

Android Studio Setup
Installation Complete
Setup was completed successfully.
Completed
Show details
< Back
Next >
Cancel

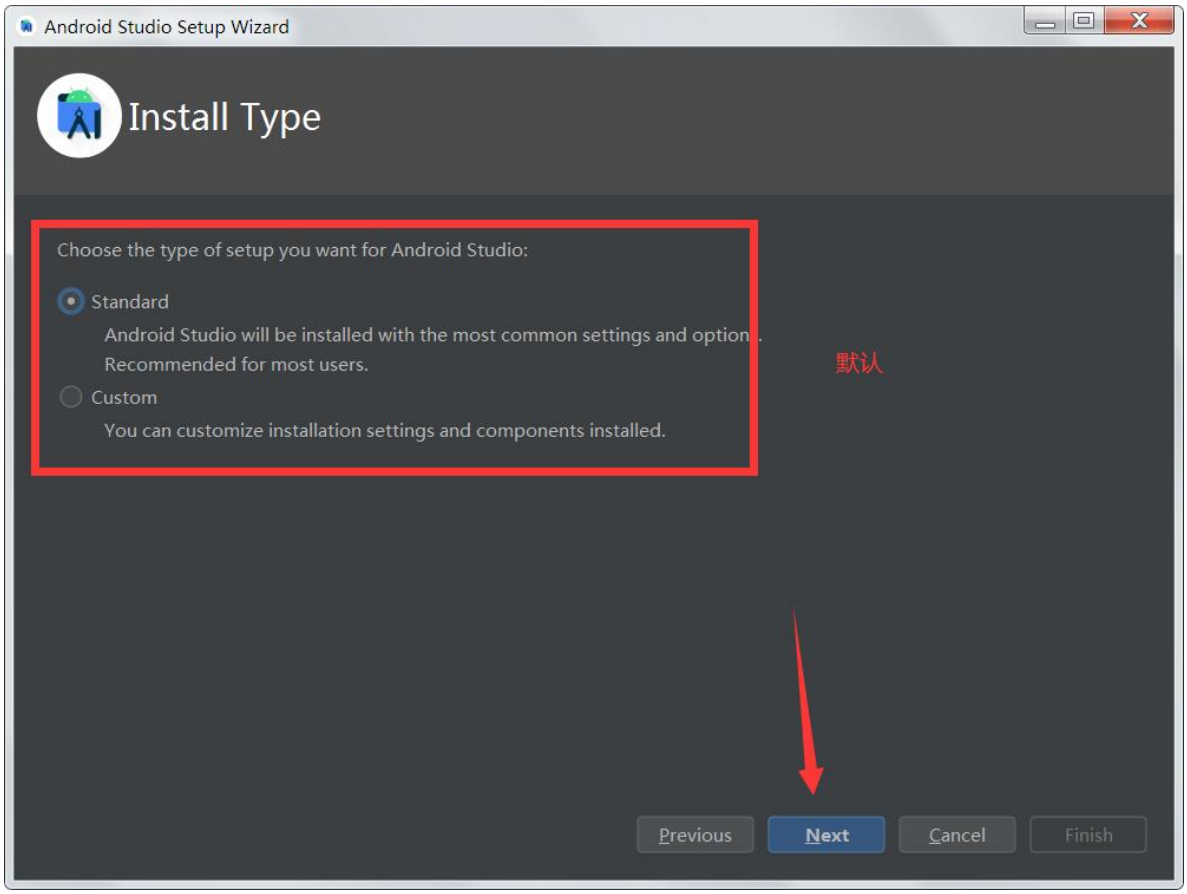

Android Studio Setup
Completing Android Studio Setup
Android Studio has been installed on your computer.
Click Finish to close Setup.
Start Android Studio
Android
Studio
< Back
Finish
Cancel

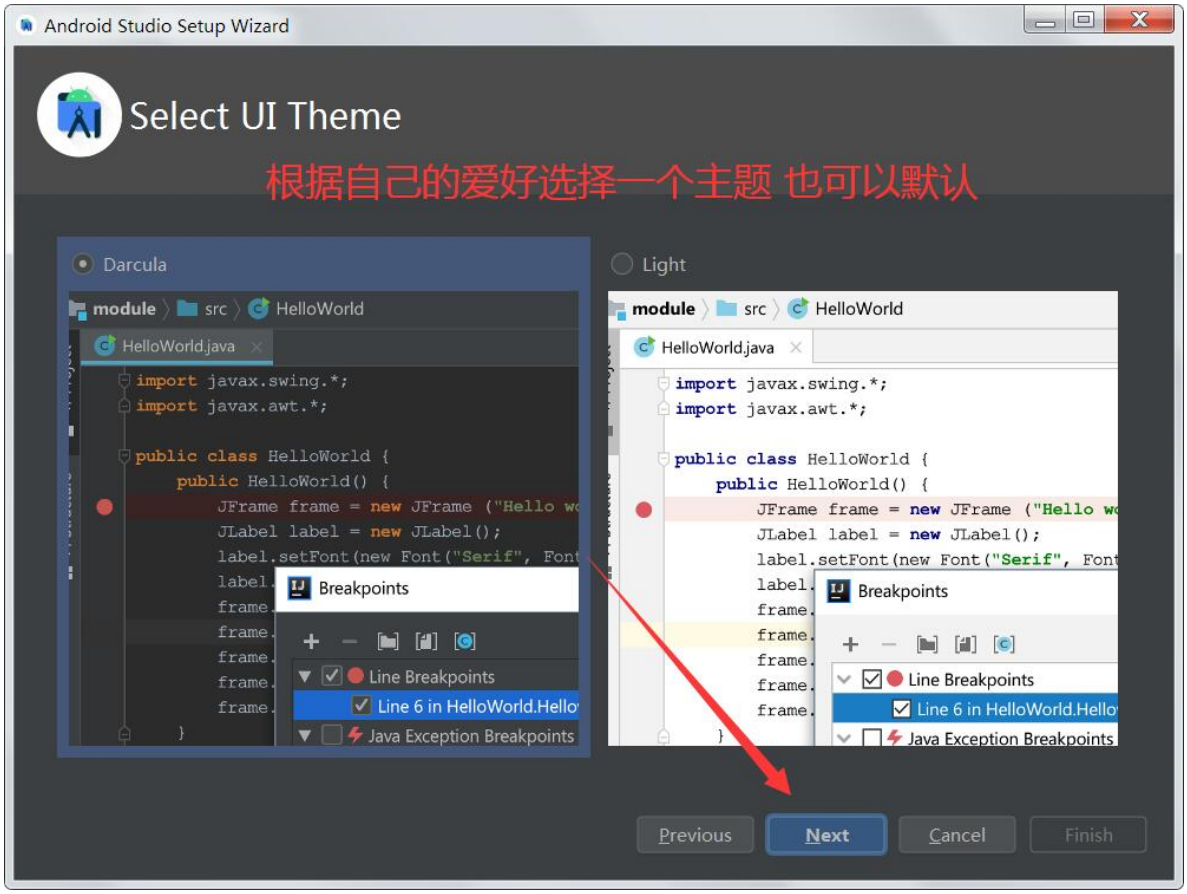

Import Android Studio Settings
Config or installation directory
Do not import settings
OK

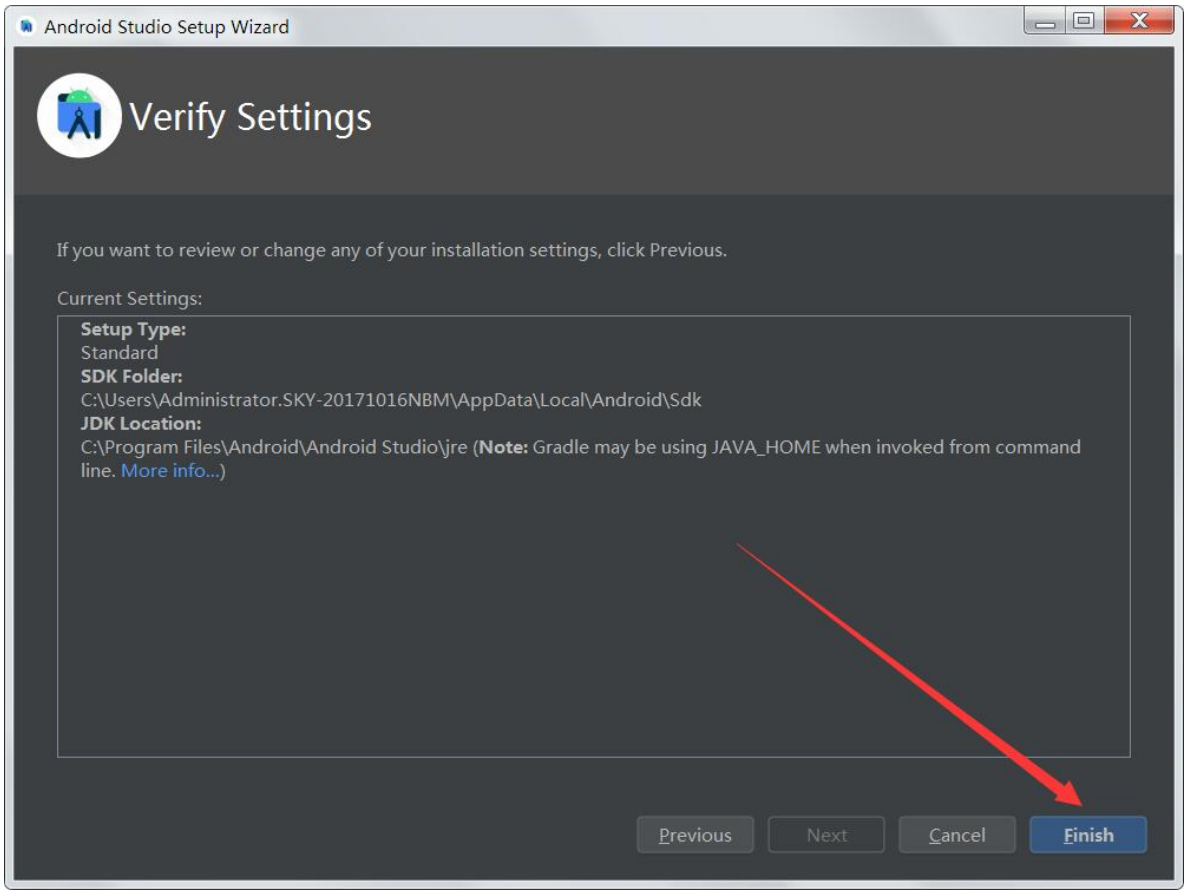

android studio
Chipmunk // 2021.2.1
Powered by the IntelliJ® Platform
Finding Available SDK Components
Starting download...
https://dl.google.com/android/repository/addons_list-5.xml

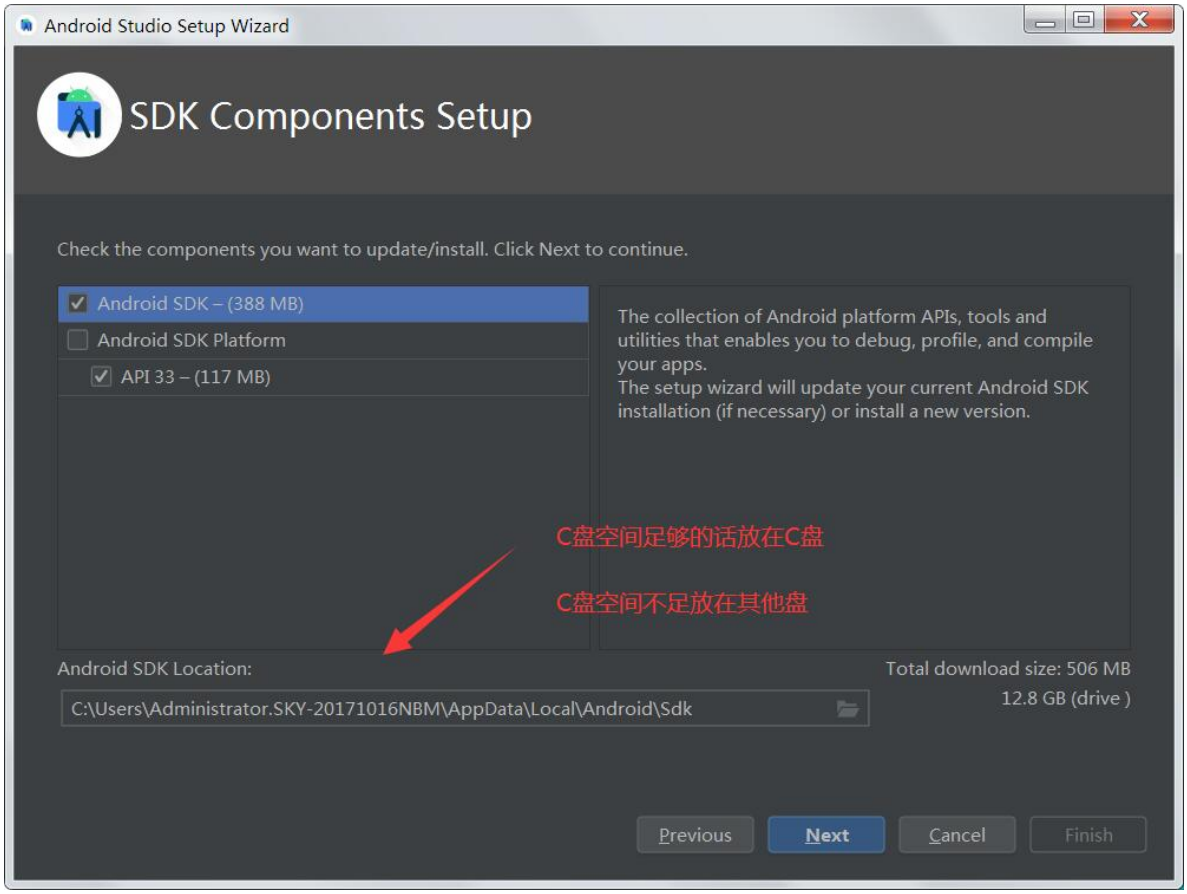

Android Studio Setup Wizard
Welcome
Android Studio
Help improve Android Studio
Allow Google to collect usage data for Android Studio and its related tools, such as how you use features and resource usage along with software identifiers such as package name and class names and plugin configuration. This data helps improve Android Studio and is collected in accordance with Google's Privacy
You can always change this behavior in Settings | Appearance & Behavior | System Settings | Data Sharing.
Don't send
Send usage statistics to Google
2
Previous
Next
Cancel
Finish

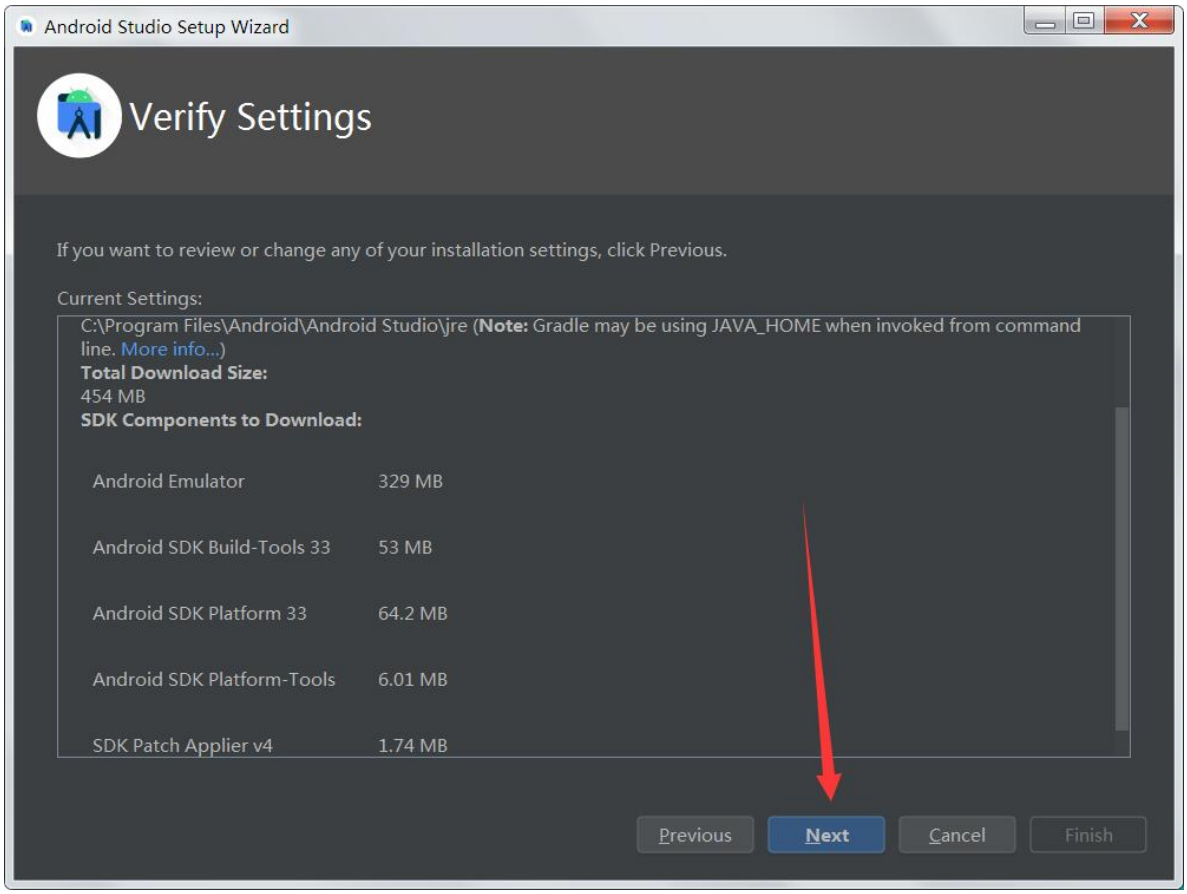

Android Studio Setup Wizard
Welcome
Android Studio
Welcome!
Addition
bid Studio.
d Studio
Android Studio First Run
Unable to access Android SDK add-on list
Setup Proxy
Cancel
Previous
Next
Cancel
Finish

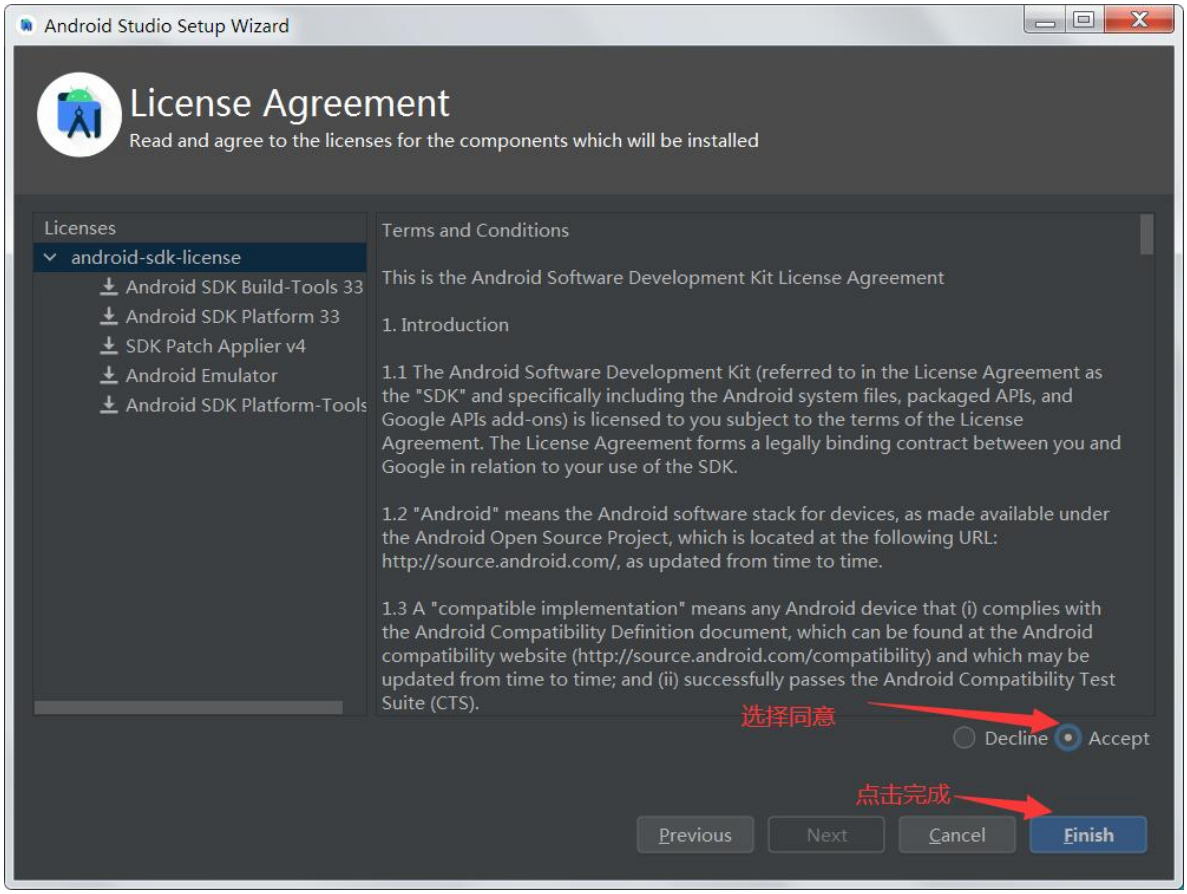

Android Studio Setup Wizard
Install Type
Choose the type of setup you want for Android Studio:
Standard
Android Studio will be installed with the most common settings and option
Recommended for most users.
Custom
You can customize installation settings and components installed.
默认
Previous
Next
Cancel
Finish

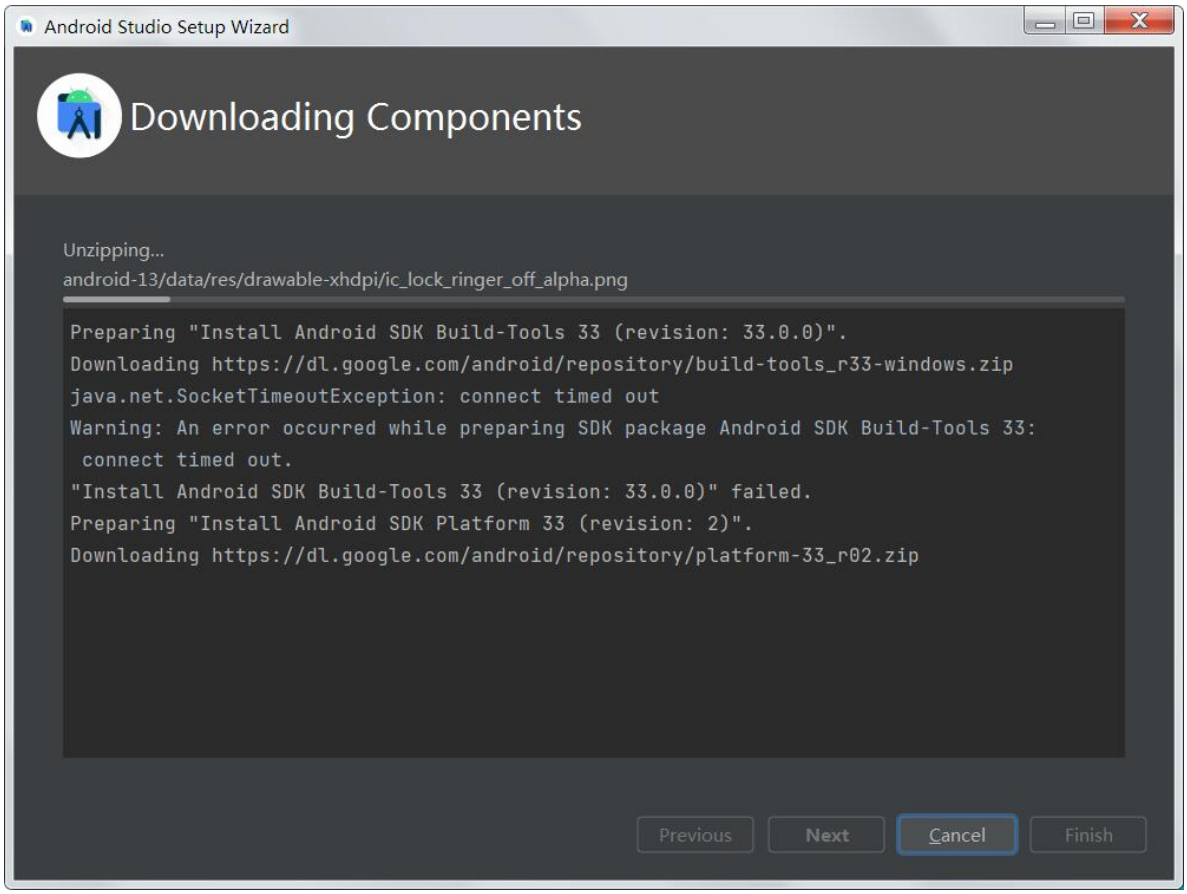

Android Studio Setup Wizard
Select UI Theme
根据自己的爱好选择一个主题 也可以默认
Darcula
Light
module
src
HelloWorld
HelloWorld.java
import javax.swing.*;
import javax.awt.*;
public class HelloWorld {
public HelloWorld() {
JFrame frame = new JFrame ("Hello
JLabel label = new JLabel();
label.setFont(new Font("Serif", Font
Breakpoints
Line Breakpoints
Line 6 in HelloWorld.Hello
Java Exception Breakpoints
Previous
Next
Cancel
Finish

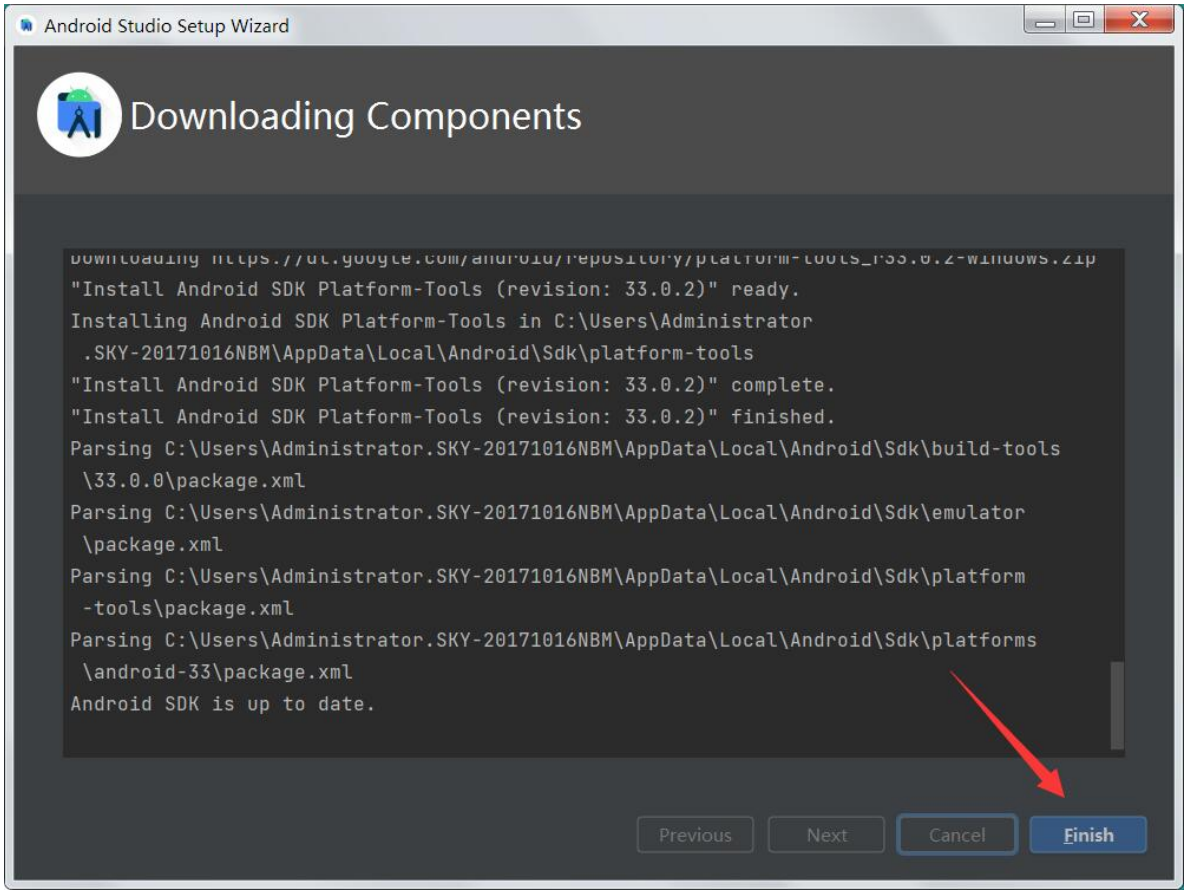

Android Studio Setup Wizard
Verify Settings
If you want to review or change any of your installation settings, click Previous.
Current Settings:
Setup Type:
Standard
SDK Folder:
C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk
JDK Location:
C:\Program Files\Android\Android Studio\jre (Note: Gradle may be using JAVA_HOME when invoked from command line. More info...)
Previous
Next
Cancel
Finish

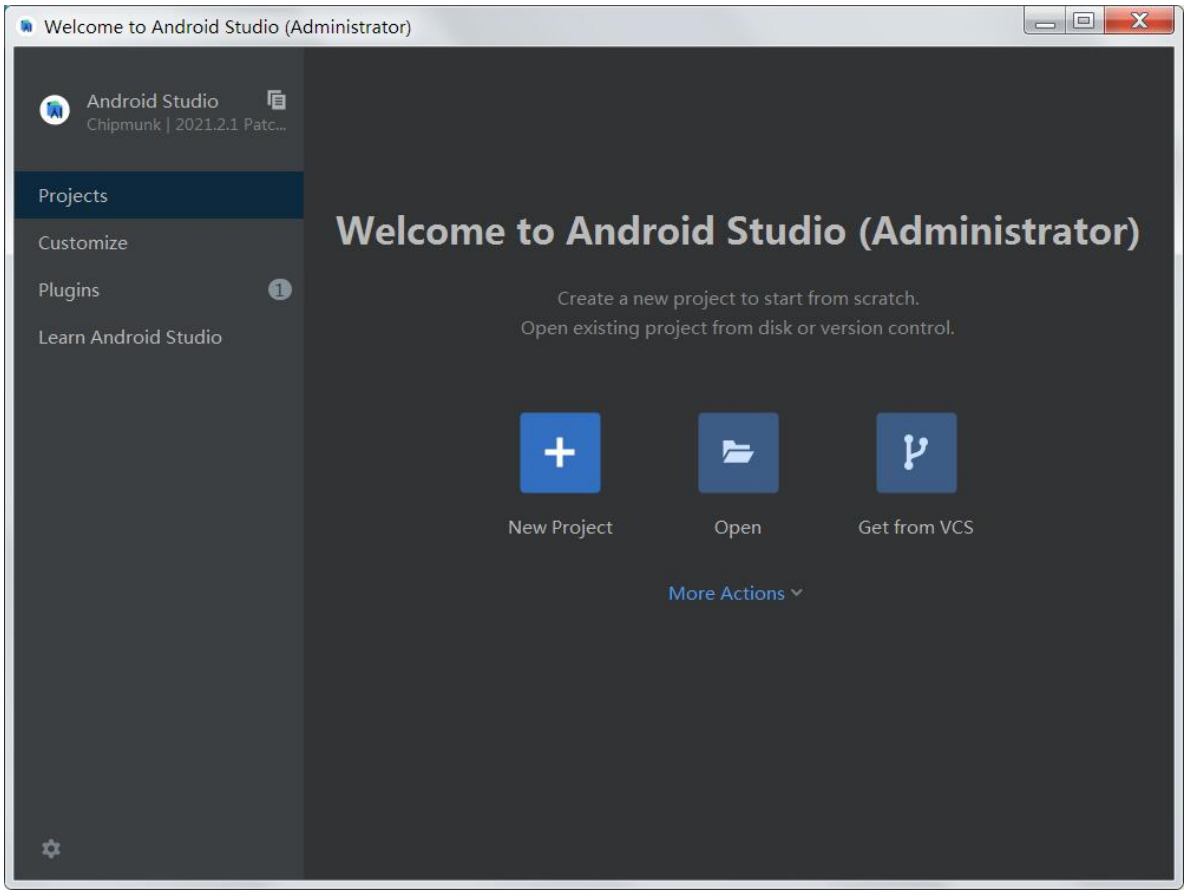

Android Studio Setup Wizard
SDK Components Setup
Check the components you want to update/install. Click Next to continue.
Android SDK – (388 MB)
Android SDK Platform
API 33 – (117 MB)
The collection of Android platform APIs, tools and utilities that enables you to debug, profile, and compile your apps.
The setup wizard will update your current Android SDK installation (if necessary) or install a new version.
C盘空间足够的话放在C盘
C盘空间不足放在其他盘
Android SDK Location:
C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk
Total download size: 506 MB
12.8 GB (drive)
Previous
Next
Cancel
Finish

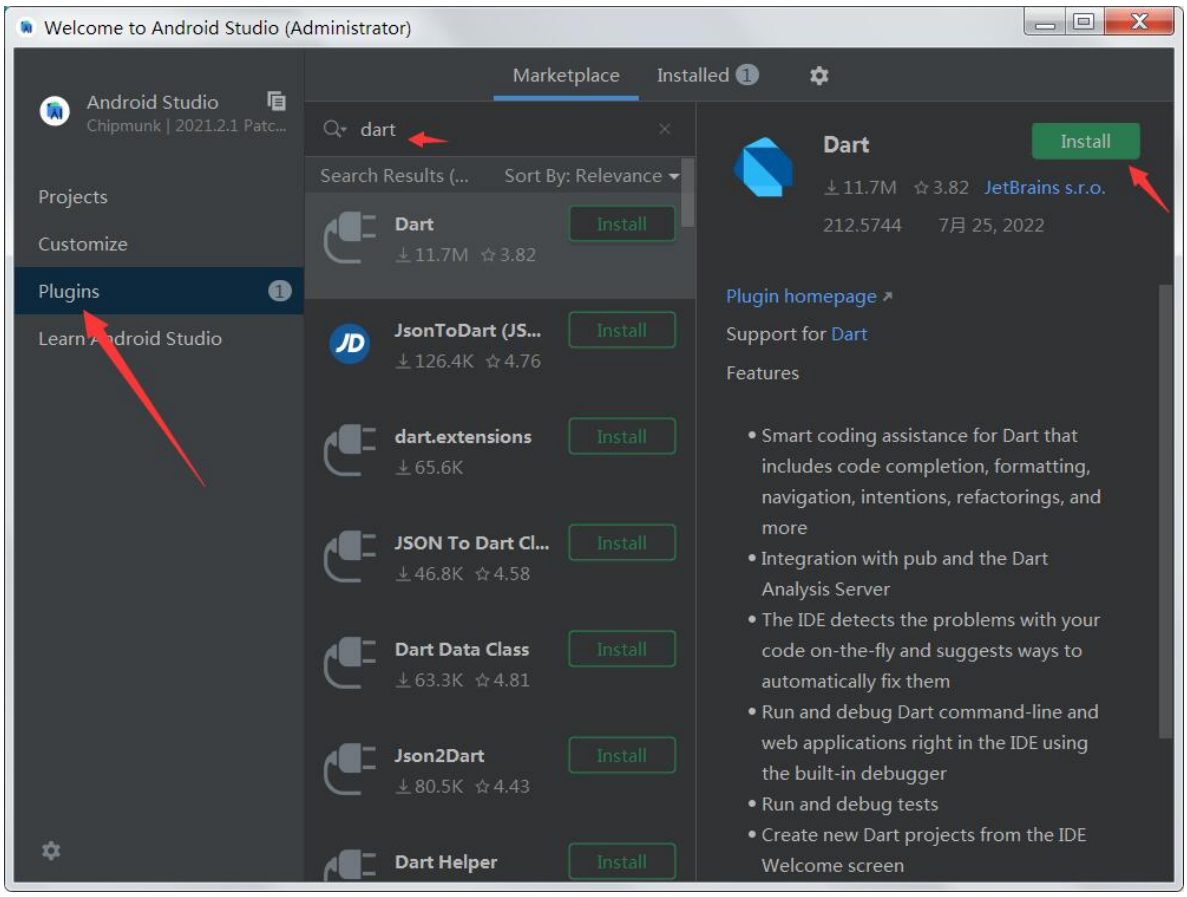

Android Studio Setup Wizard
Verify Settings
If you want to review or change any of your installation settings, click Previous.
Current Settings:
C:\Program Files\Android\Android Studio\jre (Note: Gradle may be using JAVA_HOME when invoked from command line. More info...)
Total Download Size:
454 MB
SDK Components to Download:
Android Emulator 329 MB
Android SDK Build-Tools 33 53 MB
Android SDK Platform 33 64.2 MB
Android SDK Platform-Tools 6.01 MB
SDK Patch Applier v4 1.74 MB
Previous
Next
Cancel
Finish

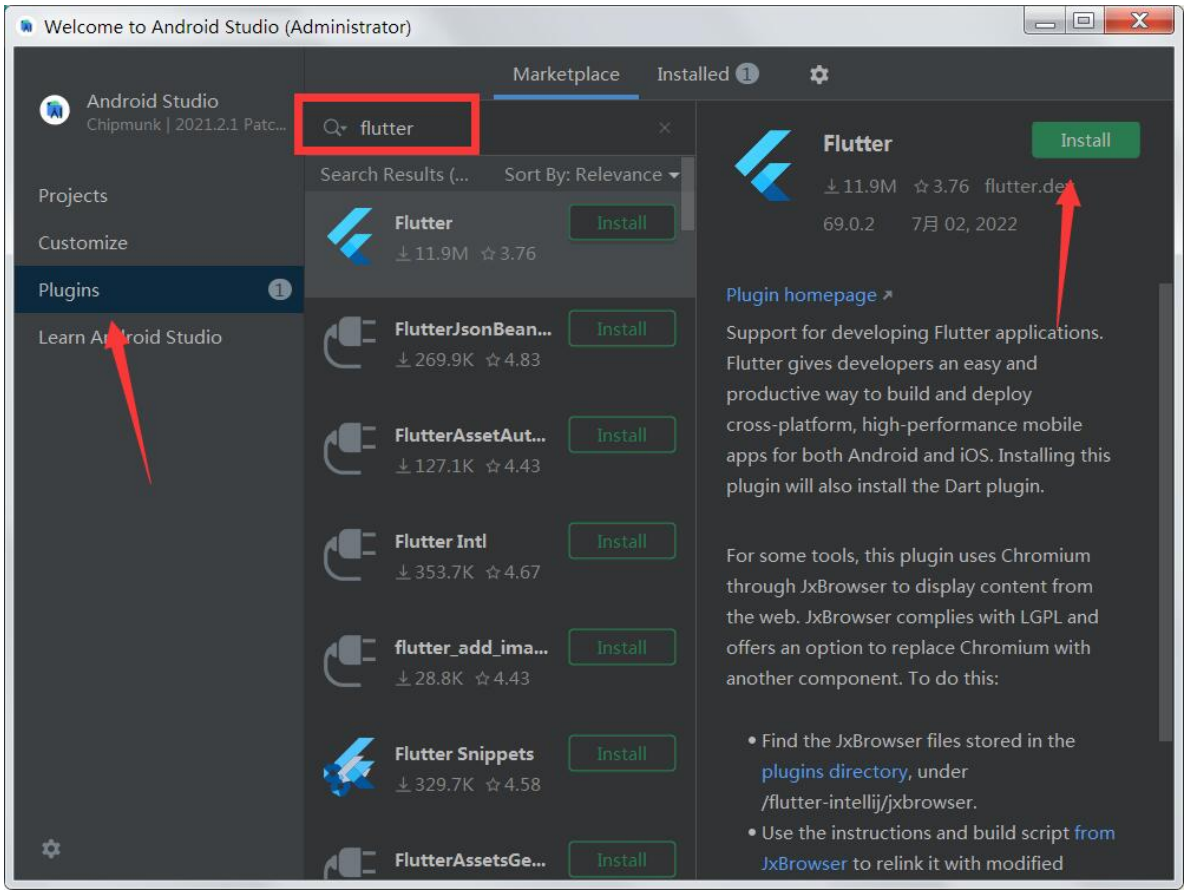

Android Studio Setup Wizard
License Agreement
Read and agree to the licenses for the components which will be installed
Licenses
android-sdk-license
Android SDK Build-Tools 33
Android SDK Platform 33
SDK Patch Applier v4
Android Emulator
Android SDK Platform-Tools
Terms and Conditions
This is the Android Software Development Kit License Agreement
1. Introduction
1.1 The Android Software Development Kit (referred to in the License Agreement as the "SDK" and specifically including the Android system files, packaged APIs, and Google APIs add-ons) is licensed to you subject to the terms of the License Agreement. The License Agreement forms a legally binding contract between you and Google in relation to your use of the SDK.
1.2 "Android" means the Android software stack for devices, as made available under the Android Open Source Project, which is located at the following URL: http://source.android.com/, as updated from time to time.
1.3 A "compatible implementation" means any Android device that (i) complies with the Android Compatibility Definition document, which can be found at the Android compatibility website (http://source.android.com/compatibility) and which may be updated from time to time; and (ii) successfully passes the Android Compatibility Test Suite (CTS).
选择同意
Decline
Accept
点击完成
Previous
Next
Cancel
Finish

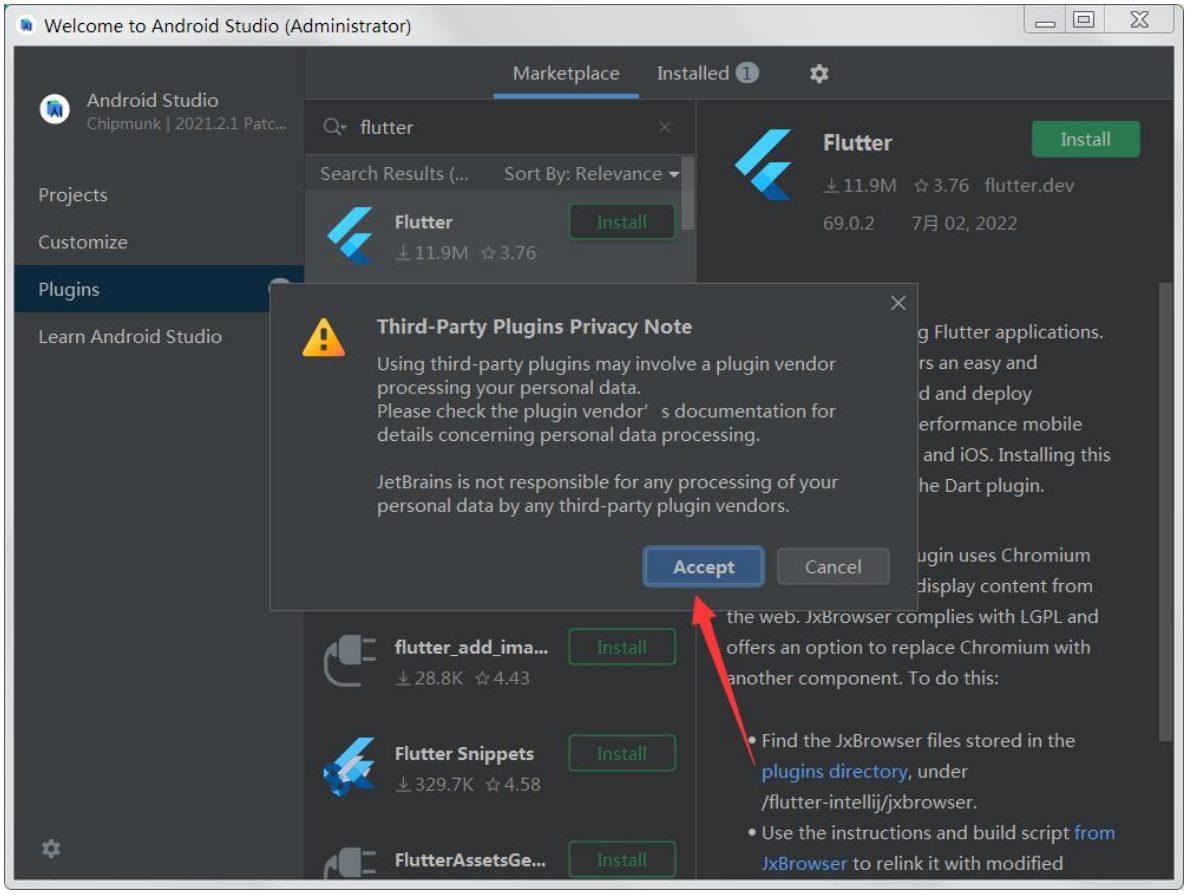

Android Studio Setup Wizard
Downloading Components
Unzipping...
android-13/data/res/drawable-xhdpi/ic_lock_ringer_off_alpha.png
Preparing "Install Android SDK Build-Tools 33 (revision: 33.0.0)".
Downloading https://dl.google.com/android/repository/build-tools_r33-windows.zip
java.net.SocketTimeoutException: connect timed out
Warning: An error occurred while preparing SDK package Android SDK Build-Tools 33:
connect timed out.
"Install Android SDK Build-Tools 33 (revision: 33.0.0)" failed.
Preparing "Install Android SDK Platform 33 (revision: 2)".
Downloading https://dl.google.com/android/repository/platform-33_r02.zip
Previous
Next
Cancel
Finish

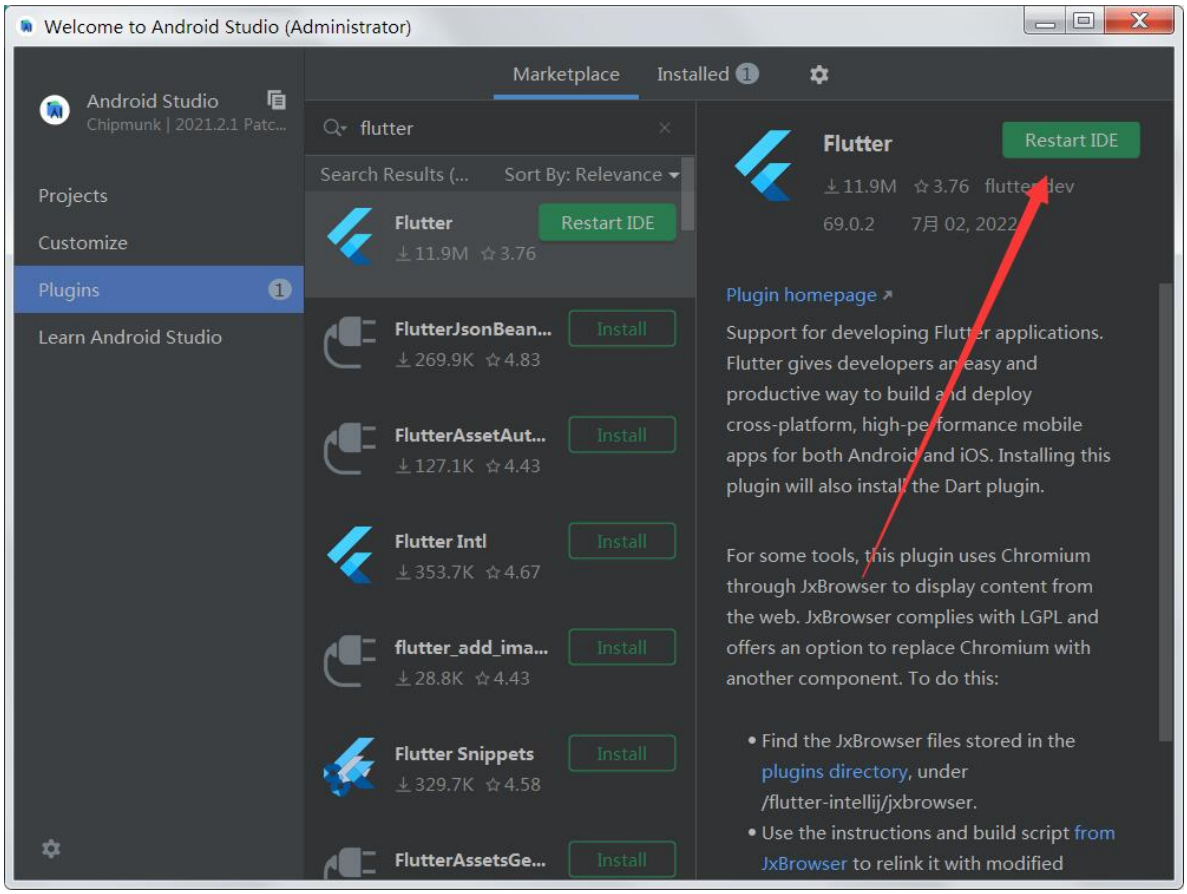

Android Studio Setup Wizard
Downloading Components
"Install Android SDK Platform-Tools (revision: 33.0.2)" ready.
Installing Android SDK Platform-Tools in C:\Users\Administrator
.SKY-20171016NBM\AppData\Local\Android\Sdk\platform-tools
"Install Android SDK Platform-Tools (revision: 33.0.2)" complete.
"Install Android SDK Platform-Tools (revision: 33.0.2)" finished.
Parsing C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk\build-tools
\33.0.0\package.xml
Parsing C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk\emulator
\package.xml
Parsing C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk\platform
-tools\package.xml
Parsing C:\Users\Administrator.SKY-20171016NBM\AppData\Local\Android\Sdk\platforms
\android-33\package.xml
Android SDK is up to date.
Previous
Next
Cancel
Finish

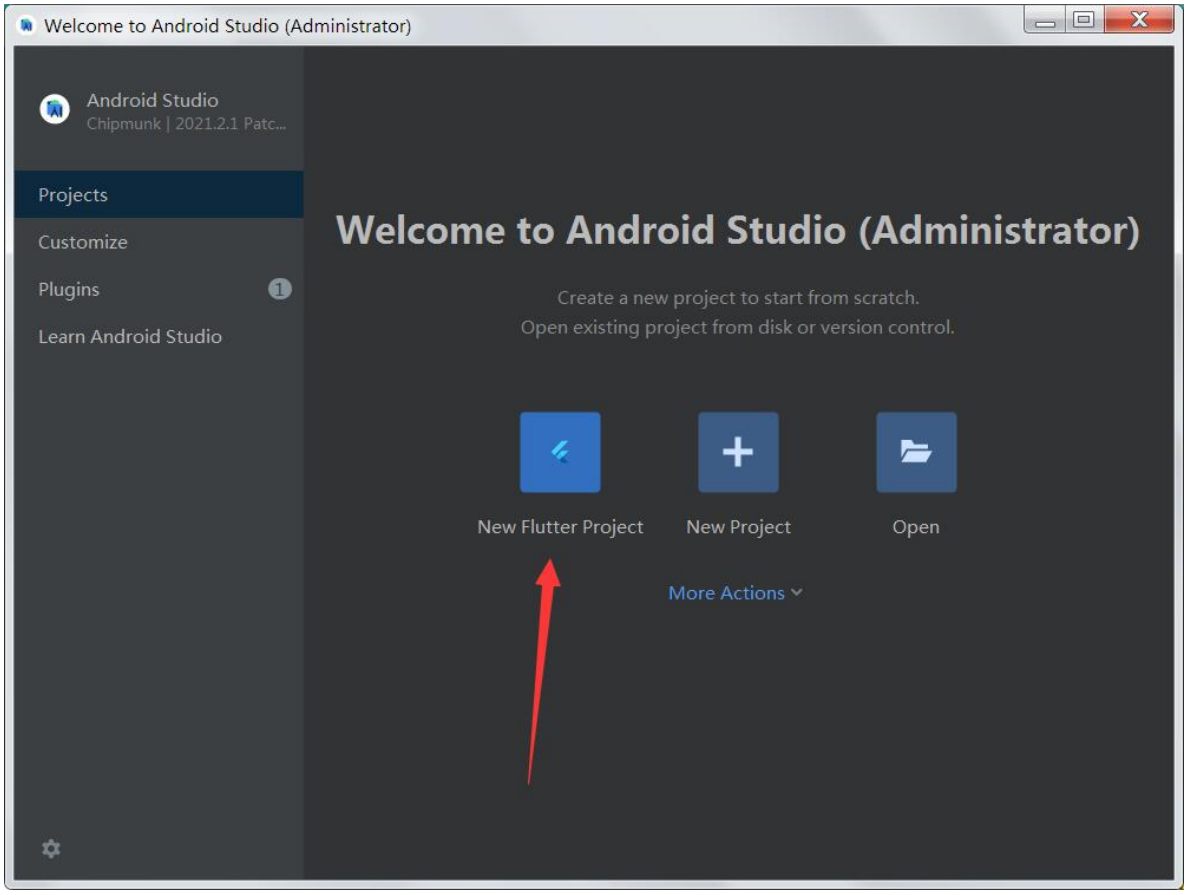

Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Welcome to Android Studio (Administrator)
Create a new project to start from scratch.
Open existing project from disk or version control.
New Project
Open
Get from VCS
More Actions


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
dart
Search Results (...
Sort By: Relevance
Dart
Install
11.7M ☆3.82
JsonToDart (JS...
126.4K ☆4.76
dart.extensions
65.6K
JSON To Dart Cl...
46.8K ☆4.58
Dart Data Class
63.3K ☆4.81
Json2Dart
80.5K ☆4.43
Dart Helper
Dart
11.7M ☆3.82 JetBrains s.r.o.
212.5744 7月 25, 2022
Plugin homepage
Support for Dart
Features
• Smart coding assistance for Dart that includes code completion, formatting, navigation, intentions, refactorings, and more
• Integration with pub and the Dart Analysis Server
• The IDE detects the problems with your code on-the-fly and suggests ways to automatically fix them
• Run and debug Dart command-line and web applications right in the IDE using the built-in debugger
• Run and debug tests
• Create new Dart projects from the IDE Welcome screen


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutter
Search Results (...
Sort By: Relevance
Flutter
Install
11.9M ☆3.76
FlutterJsonBean...
269.9K ☆4.83
FlutterAssetAut...
127.1K ☆4.43
Flutter Intl
353.7K ☆4.67
flutter_add_ima...
28.8K ☆4.43
Flutter Snippets
329.7K ☆4.58
FlutterAssetsGe...
Flutter
11.9M ☆3.76 flutter.de
69.0.2 7月 02, 2022
Plugin homepage
Support for developing Flutter applications. Flutter gives developers an easy and productive way to build and deploy cross-platform, high-performance mobile apps for both Android and iOS. Installing this plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium through JxBrowser to display content from the web. JxBrowser complies with LGPL and offers an option to replace Chromium with another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed
flutter
Search Results (...
Sort By: Relevance
Flutter
Install
11.9M 3.76
Third-Party Plugins Privacy Note
Using third-party plugins may involve a plugin vendor processing your personal data.
Please check the plugin vendor' s documentation for details concerning personal data processing.
JetBrains is not responsible for any processing of your personal data by any third-party plugin vendors.
Accept
Cancel
flutter_add_ima...
28.8K 4.43
Flutter Snippets
329.7K 4.58
FlutterAssetsGe...
Flutter
Install
11.9M 3.76 flutter.dev
69.0.2 7月 02, 2022
g Flutter applications.
rs an easy and
d and deploy
erformance mobile
and iOS. Installing this
he Dart plugin.
ugin uses Chromium
display content from
the web. JxBrowser complies with LGPL and
offers an option to replace Chromium with
another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
1
Learn Android Studio
Marketplace
Installed
flutter
Search Results (...
Sort By: Relevance
Flutter
Restart IDE
11.9M 3.76
FlutterJsonBean...
269.9K 4.83
FlutterAssetAut...
127.1K 4.43
Flutter Intl
353.7K 4.67
flutter_add_ima...
28.8K 4.43
Flutter Snippets
329.7K 4.58
FlutterAssetsGe...
Install
Restart IDE
11.9M 3.76 flutter.dev
69.0.2 7月 02, 2022
Plugin homepage
Support for developing Flutter applications.
Flutter gives developers an easy and
productive way to build and deploy
cross-platform, high-performance mobile
apps for both Android and iOS. Installing this
plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium
through JxBrowser to display content from
the web. JxBrowser complies with LGPL and
offers an option to replace Chromium with
another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified



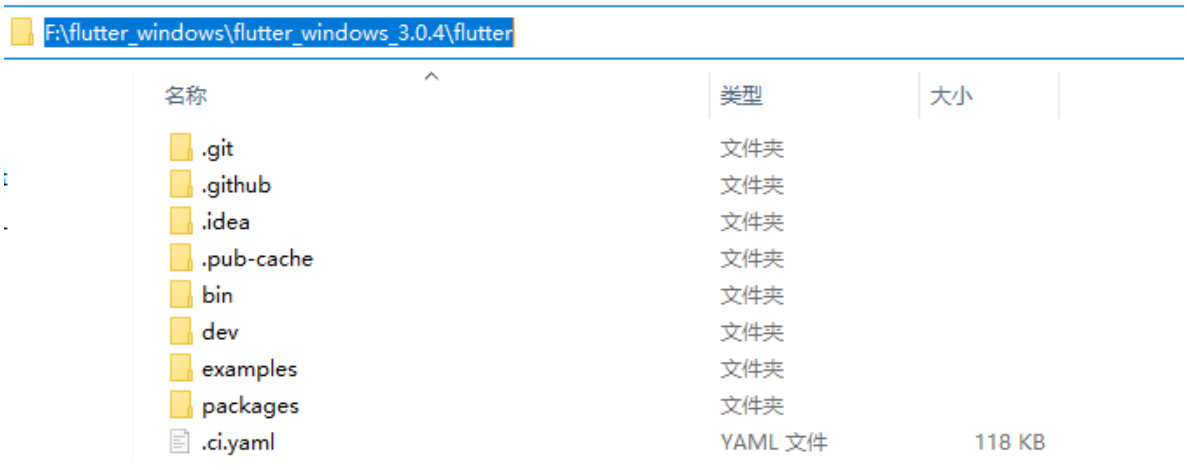
2.3、电脑上面下载配置Flutter Sdk

1、下载Flutter SDK

https://flutter.dev/docs/development/tools/sdk/releases#windows

2、把下载好的Flutter SDK随便减压到你想安装Sdk的目录

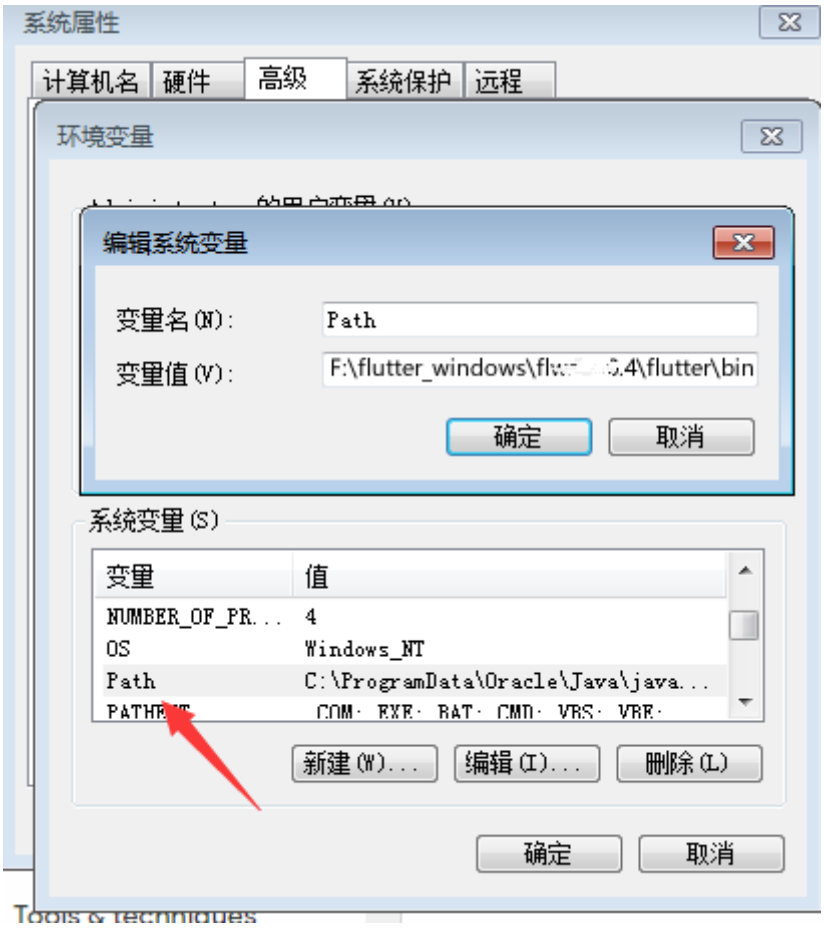
如减压到（F:\flutter_windows\flutter_windows_3.0.4\flutter）



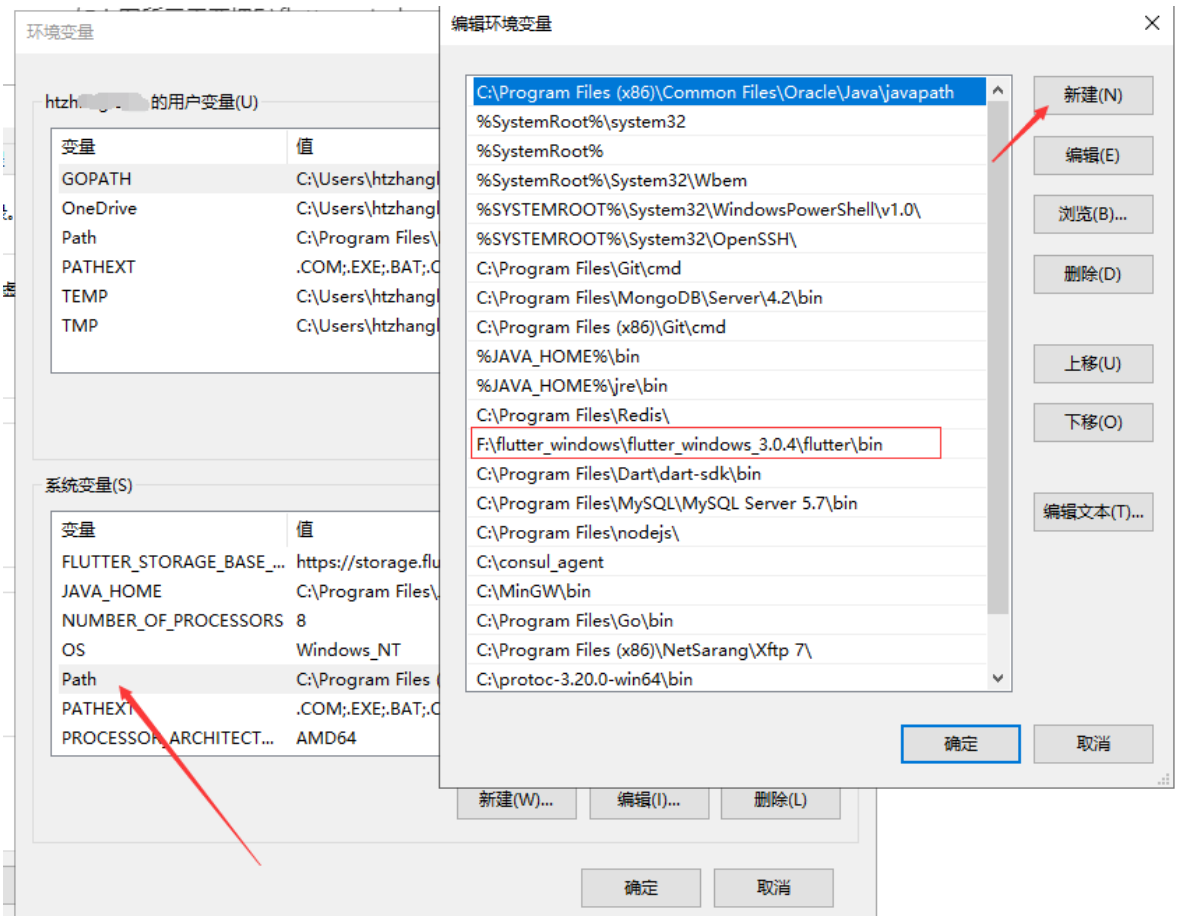
3、把 Flutter安装目录的bin目录配置到环境变量。

如上图所示需要把F:\flutter_windows\flutter_windows_3.0.4\flutter\bin目录配置到path环境变量里面

windows7：



windows10、windows11：



2.4、电脑上配置Flutter国内镜像

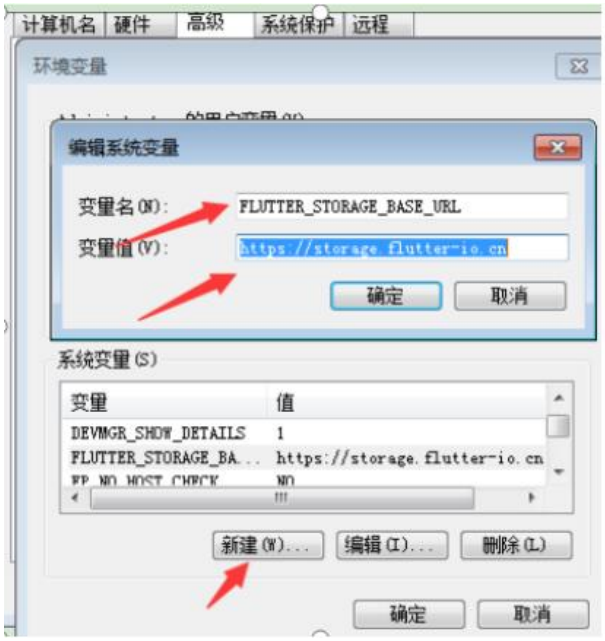
搭建环境过程中要下载很多资源文件，当一些资源下载不了的时候，可能会报各种错误。在国内访问Flutter的时候有可能会受到限制。Flutter官方为我们提供了国内的镜像

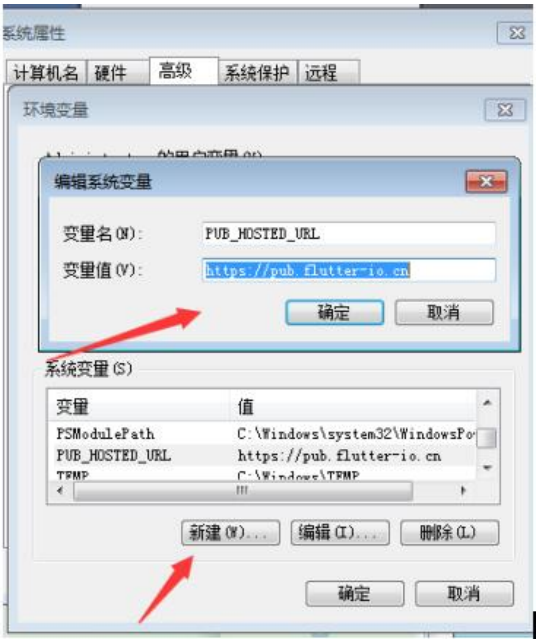
https://flutter.dev/community/china

https://flutter-io.cn/

拉到Flutter中文网最下面有配置方式，把下面两句配置到环境变量里面

```
FLUTTER_STORAGE_BASE_URL: https://storage.flutter-io.cn
PUB_HOSTED_URL: https://pub.flutter-io.cn
```





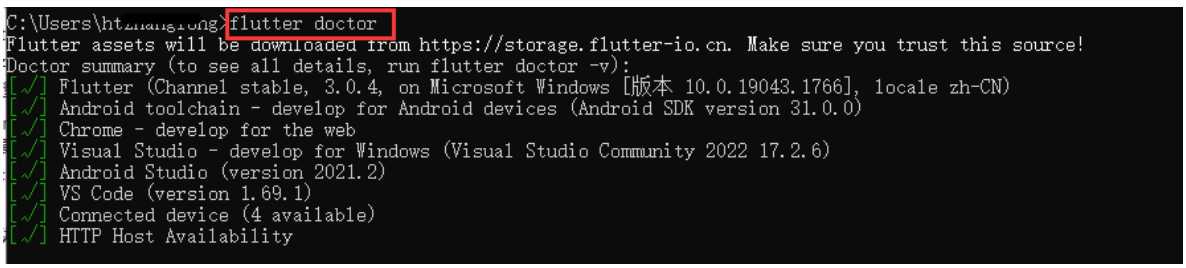
Flutter 社区镜像

```
FLUTTER_STORAGE_BASE_URL: https://storage.flutter-io.cn
PUB_HOSTED_URL: https://pub.flutter-io.cn
```

清华大学 TUNA 协会镜像

```
FLUTTER_STORAGE_BASE_URL: https://mirrors.tuna.tsinghua.edu.cn/flutter
PUB_HOSTED_URL: https://mirrors.tuna.tsinghua.edu.cn/dart-pub
```

2.5、运行 flutter doctor命令检测环境是否配置成功



第一次执行可能会提示下面错误：

1、错误一：`cmdline-tools component is missing`

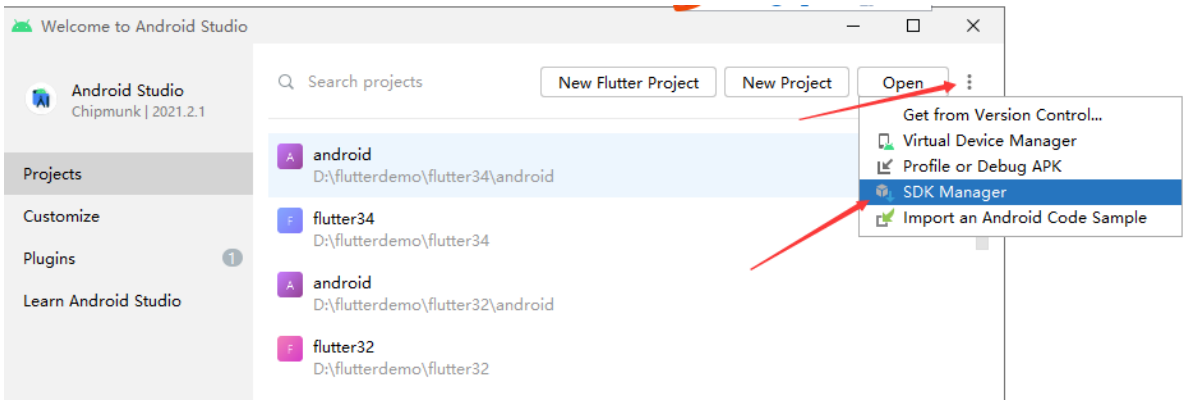
```
C:\Users\MSI>flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel master, 2.4.0-5.0.pre.88, on Microsoft Windows [Version 10.0.18363.1646], locale zh-CN)
[!] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
    X cmdline-tools component is missing
      Run `path/to/sdkmanager --install "cmdline-tools;latest"`
      See https://developer.android.com/studio/command-line for more details.
    X Android license status unknown.
      Run `flutter doctor --android-licenses` to accept the SDK licenses.
      See https://flutter.dev/docs/get-started/install/windows#android-setup for more details.
[√] Chrome - develop for the web
[√] Android Studio (version 4.2.0)
[√] IntelliJ IDEA Ultimate Edition (version 2020.3)
[√] Connected device (2 available)
```

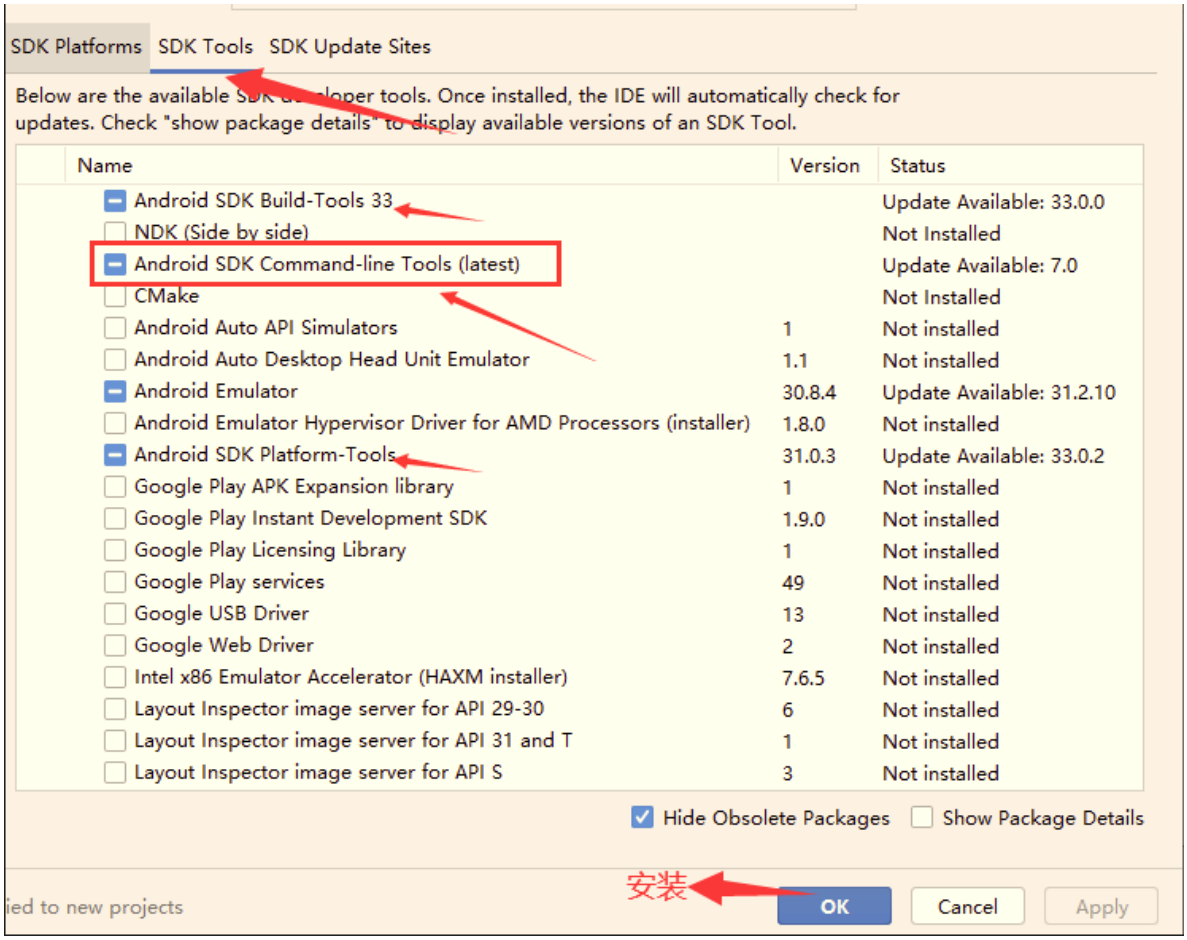
2、错误二：`Visual Studio not installed` 如果只是开发Flutter APP可以忽略此错误信息

```
C:\Users\h[redacted]g>flutter doctor
Flutter assets will be downloaded from https://storage.flutter-io.cn. Make sure you trust this source!
Doctor summary (to see all details, run flutter doctor -v):
[√] Flutter (Channel stable, 3.0.4, on Microsoft Windows [版本 10.0.19043.1766], locale zh-CN)
[√] Android toolchain - develop for Android devices (Android SDK version 31.0.0)
[√] Chrome - develop for the web
[X] Visual Studio - develop for Windows
    X Visual Studio not installed; this is necessary for Windows development.
      Download at https://visualstudio.microsoft.com/downloads/.
      Please install the "Desktop development with C++" workload, including all of its default components
[√] Android Studio (version 2021.2)
[√] VS Code (version 1.69.1)
[√] Connected device (4 available)
[√] HTTP Host Availability

! Doctor found issues in 1 category.
```

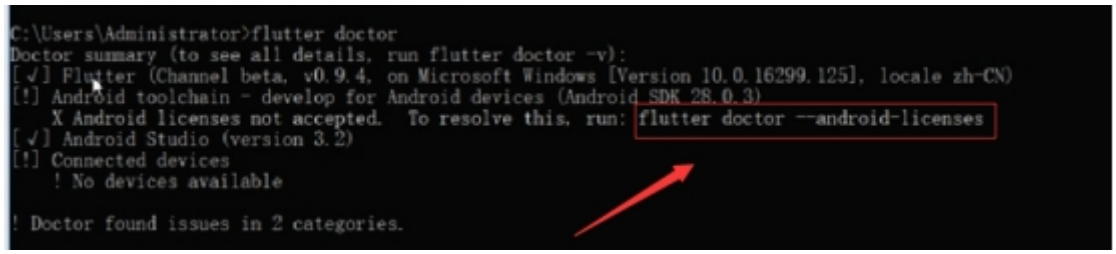
3、错误一的解决方法安装cmdline-tools 以及配置android-licenses：

3.1 安装cmdline-tools





3.2 配置android-licenses



这个时候复制上面红色框框内的命令

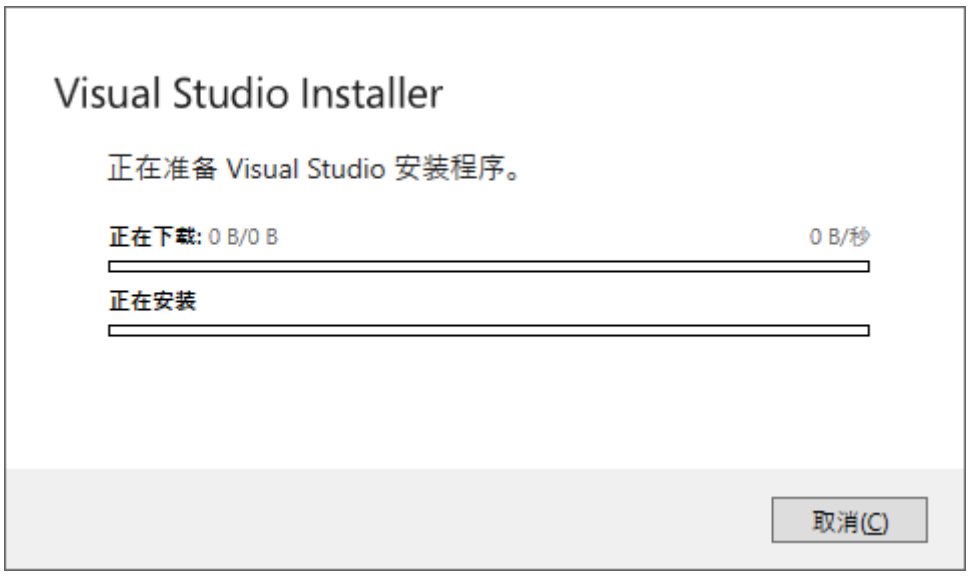
```
flutter doctor --android-licenses
```

注意：提示输入Y/N的地方全部输入Y

4、错误二的解决方法安装Visual Studio：

Visual Studio主要用于flutter 桌面软件开发，如果您只是开发flutter app可以不用安装Visual Studio

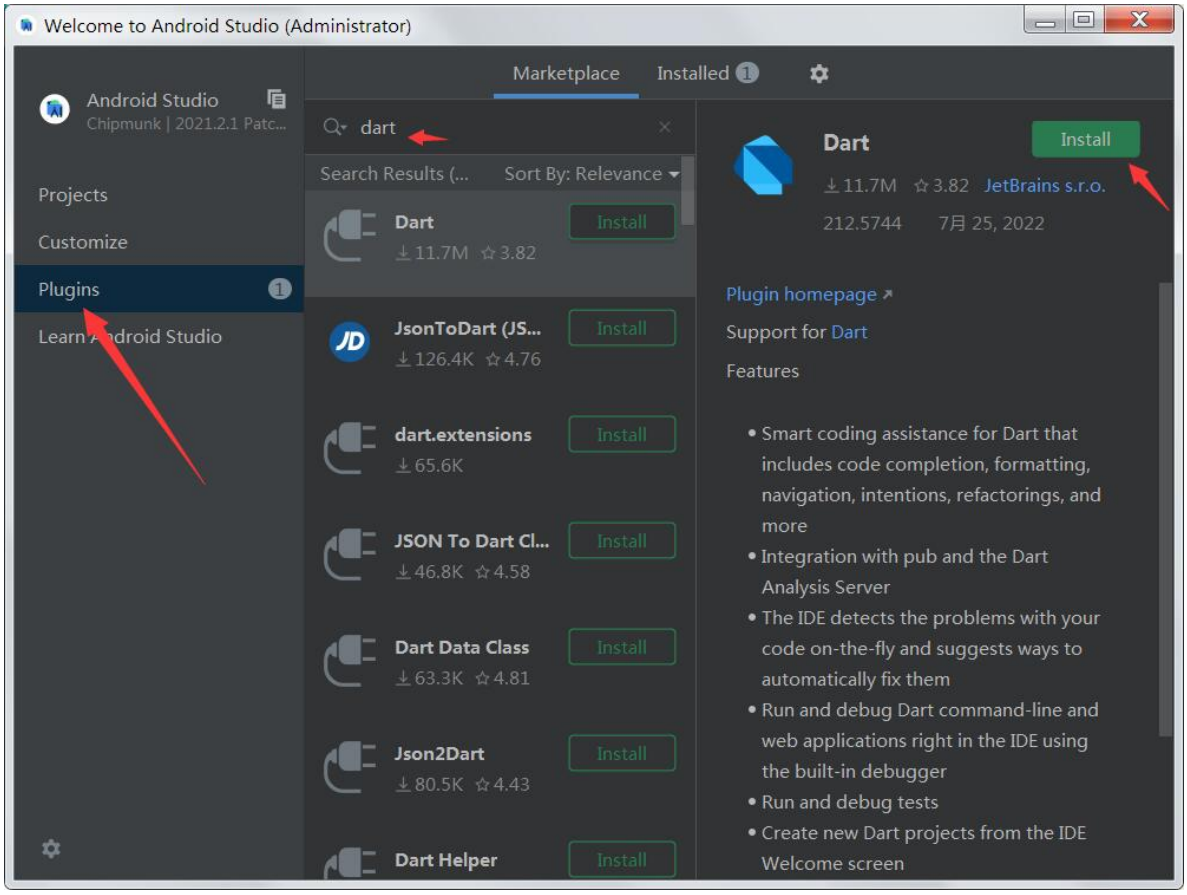
https://visualstudio.microsoft.com/zh-hans/downloads/

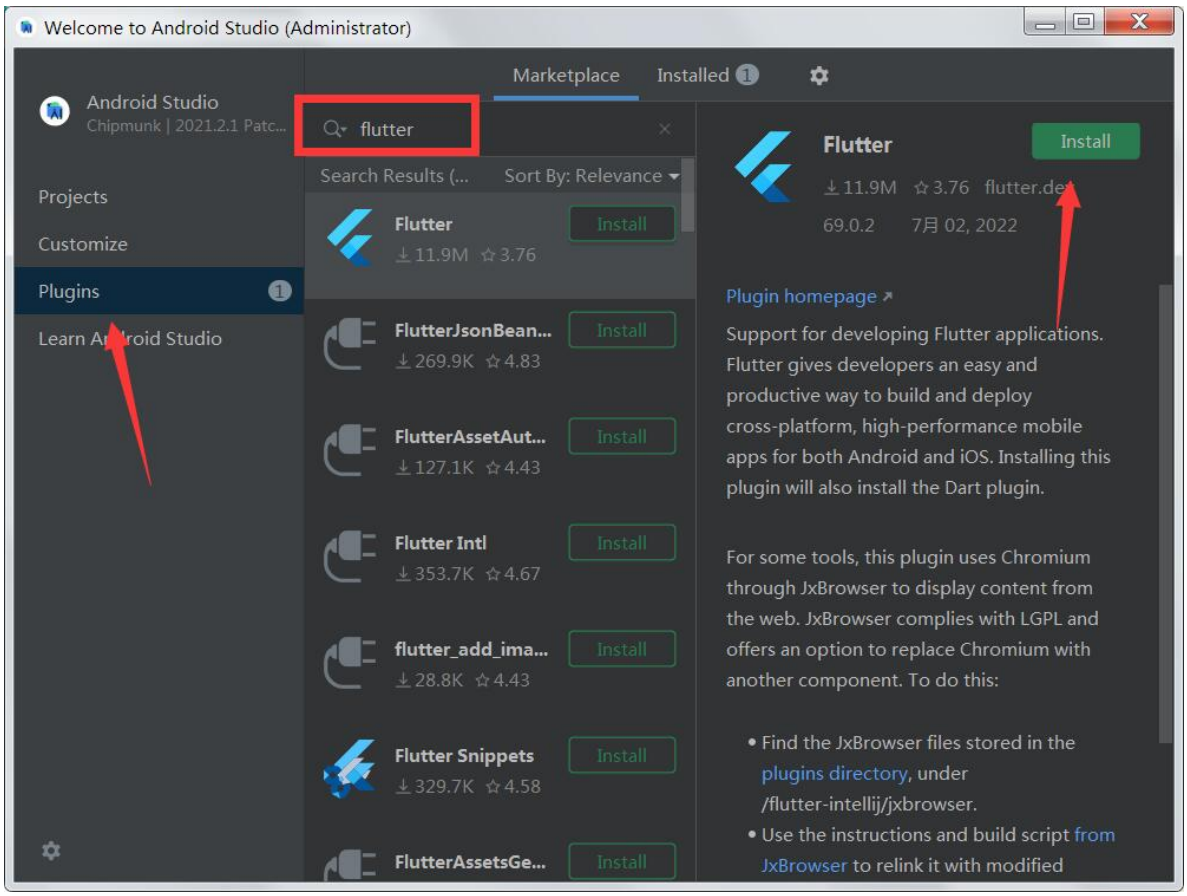


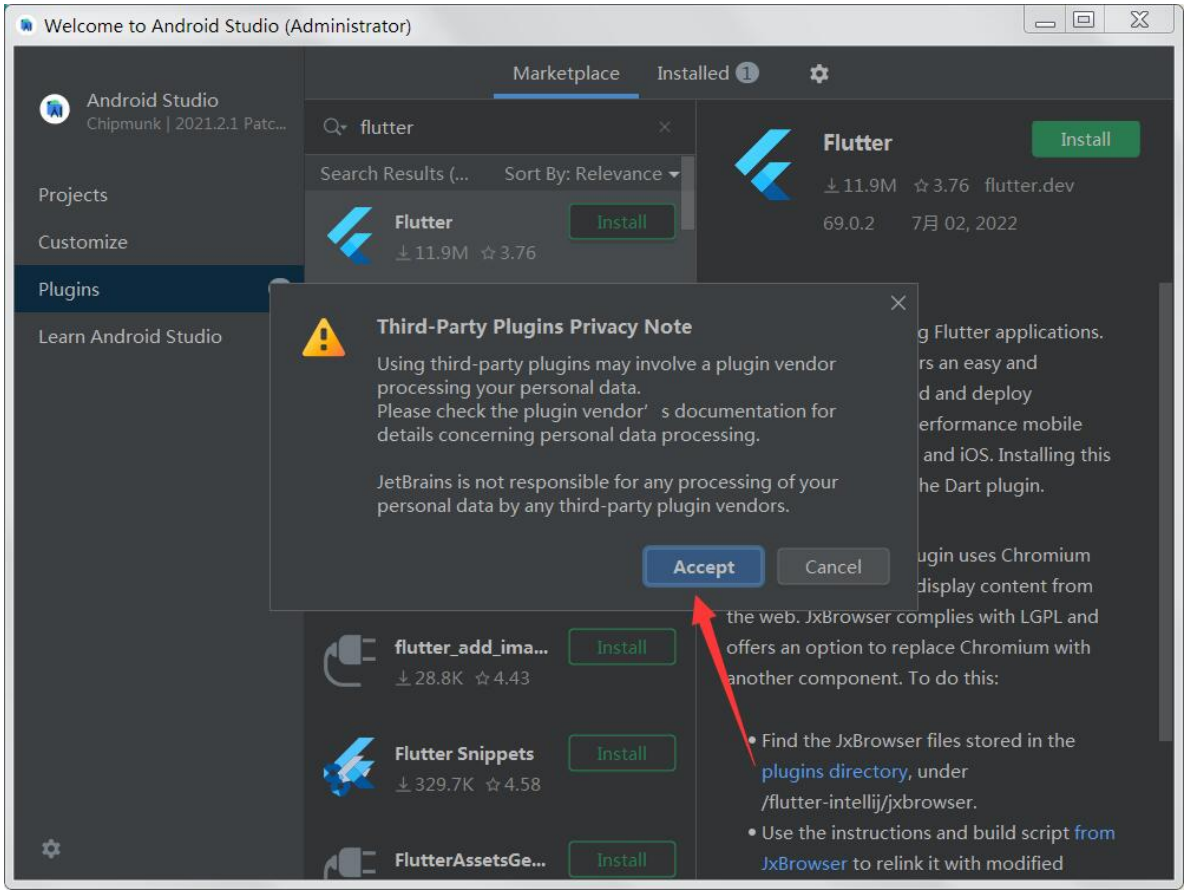
如果安装失败可以修改DNS尝试

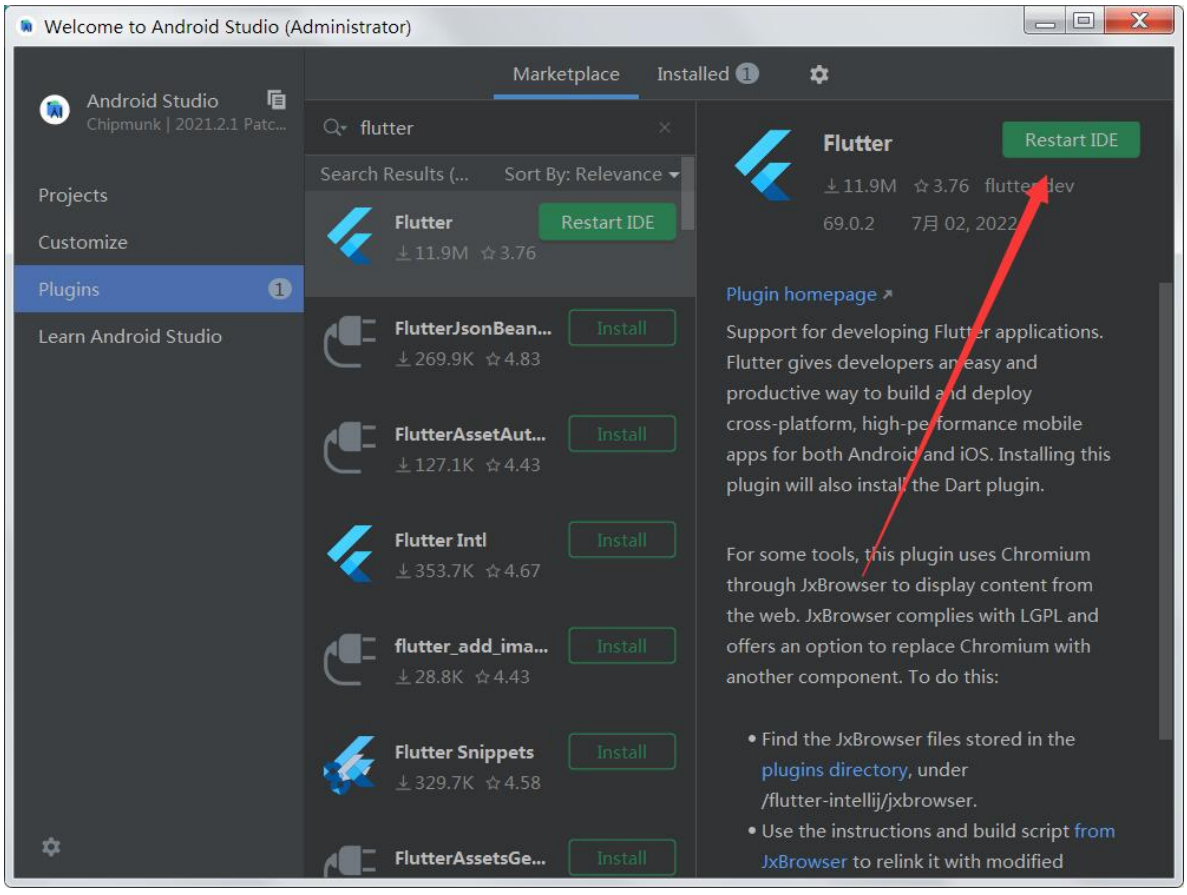
2.6、打开Android Studio 安装Flutter插件

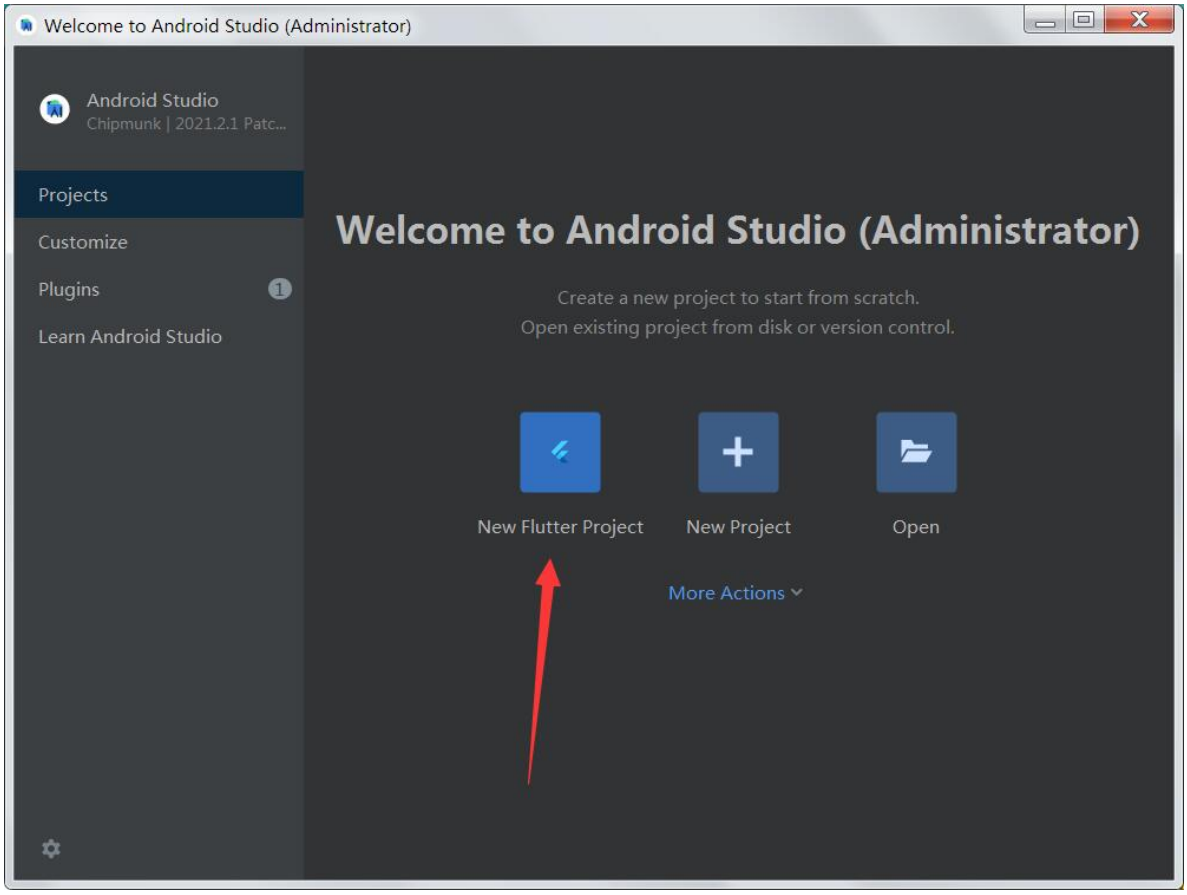
新版Android Studio配置



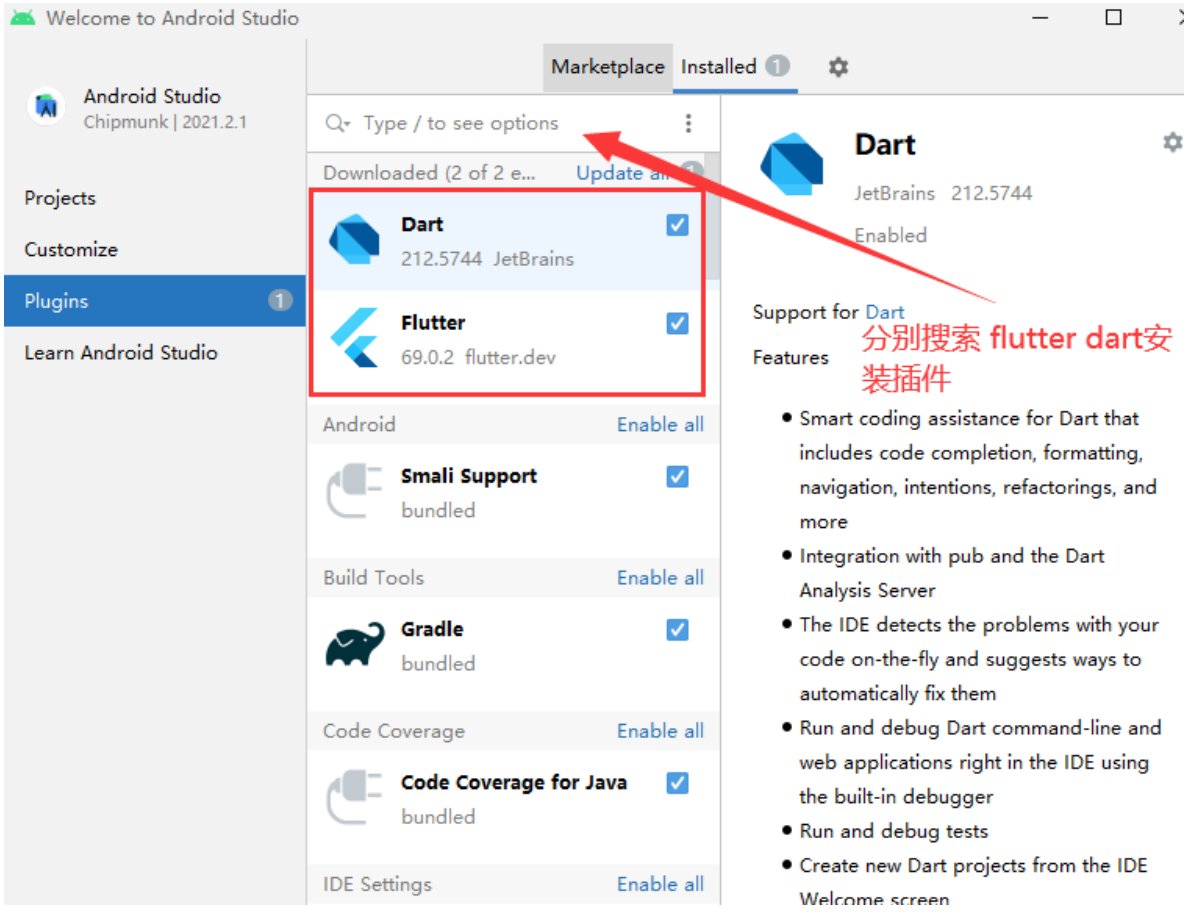


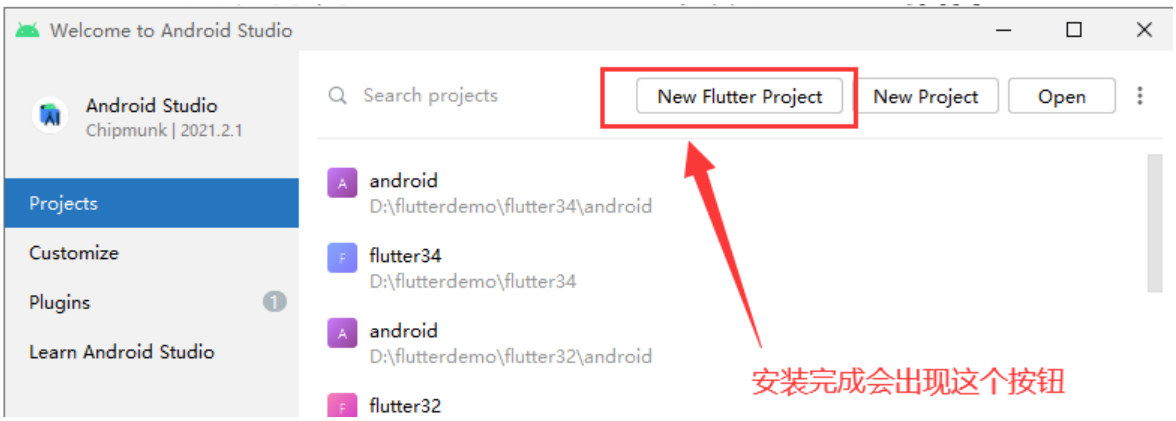

Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins
Learn Android Studio
Marketplace
Installed 1
flutter
Search Results (...
Sort By: Relevance
Flutter
Install
11.9M 3.76
Third-Party Plugins Privacy Note
Using third-party plugins may involve a plugin vendor processing your personal data.
Please check the plugin vendor' s documentation for details concerning personal data processing.
JetBrains is not responsible for any processing of your personal data by any third-party plugin vendors.
Accept
Cancel
flutter_add_ima...
28.8K 4.43
Flutter Snippets
329.7K 4.58
FlutterAssetsGe...
Flutter
Install
69.0.2 7月 02, 2022
g Flutter applications.
rs an easy and
d and deploy
erformance mobile
and iOS. Installing this
he Dart plugin.
ugin uses Chromium
display content from
the web. JxBrowser complies with LGPL and
offers an option to replace Chromium with
another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified


Welcome to Android Studio (Administrator)
Android Studio
Chipmunk | 2021.2.1 Patc...
Projects
Customize
Plugins 1
Learn Android Studio
Marketplace
Installed 1
flutter
Search Results (...
Sort By: Relevance
Flutter
Restart IDE
11.9M 3.76
FlutterJsonBean...
269.9K 4.83
FlutterAssetAut...
127.1K 4.43
Flutter Intl
353.7K 4.67
flutter_add_ima...
28.8K 4.43
Flutter Snippets
329.7K 4.58
FlutterAssetsGe...
Install
Flutter
Restart IDE
69.0.2 7月 02, 2022
Plugin homepage ↗
Support for developing Flutter applications.
Flutter gives developers an easy and
productive way to build and deploy
cross-platform, high-performance mobile
apps for both Android and iOS. Installing this
plugin will also install the Dart plugin.
For some tools, this plugin uses Chromium
through JxBrowser to display content from
the web. JxBrowser complies with LGPL and
offers an option to replace Chromium with
another component. To do this:
• Find the JxBrowser files stored in the plugins directory, under /flutter-intellij/jxbrowser.
• Use the instructions and build script from JxBrowser to relink it with modified

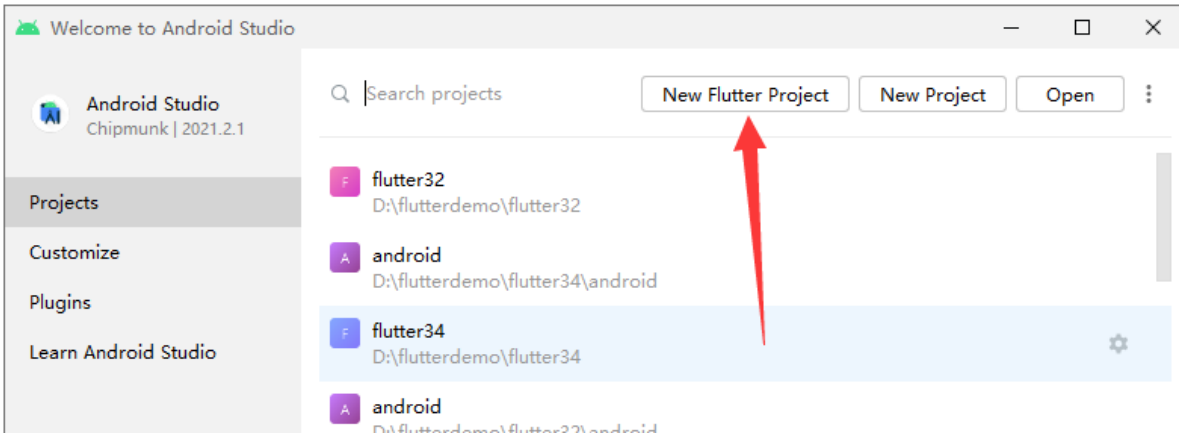


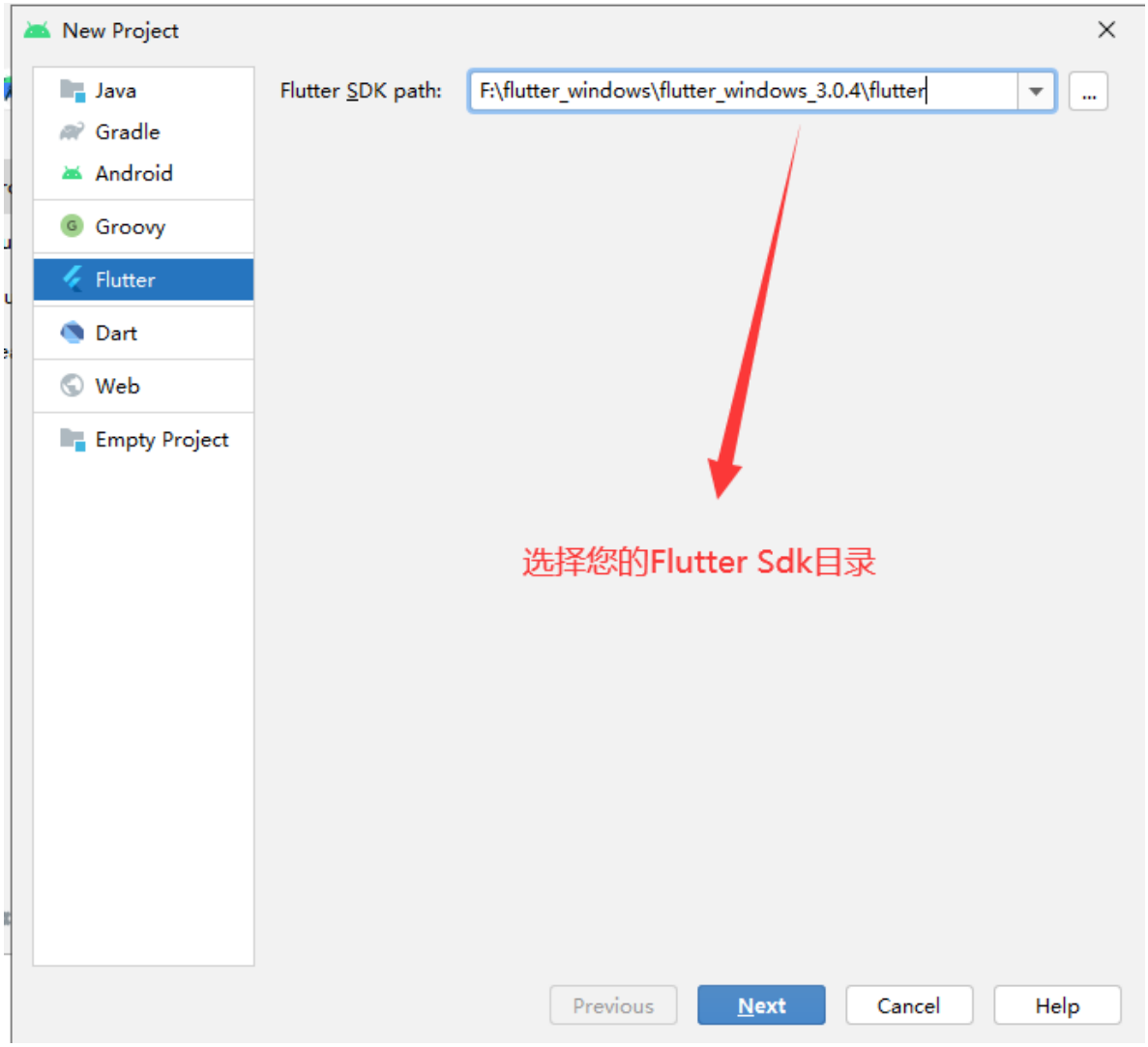
提示：新版本android studio也可能是下面界面

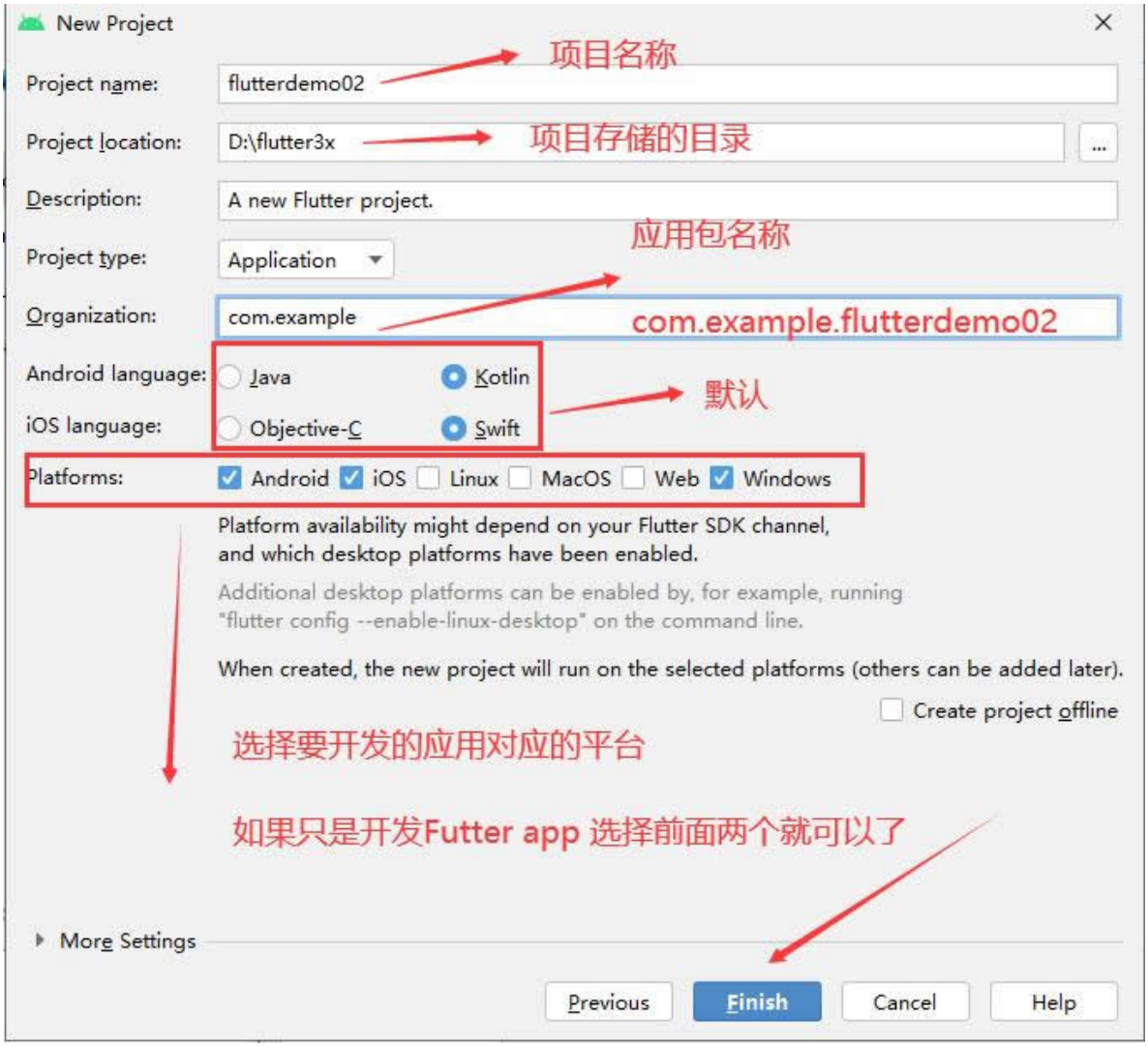


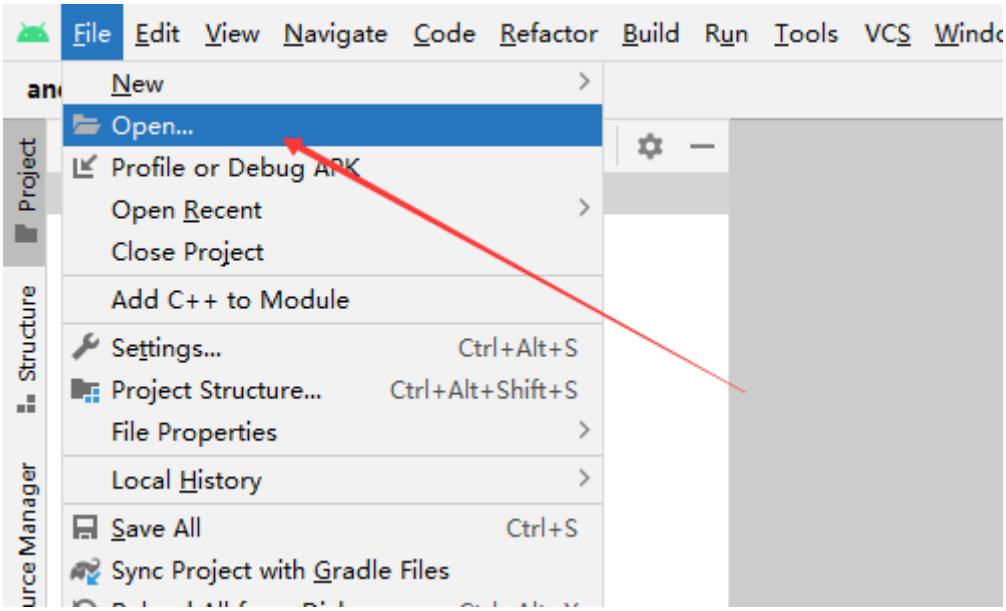


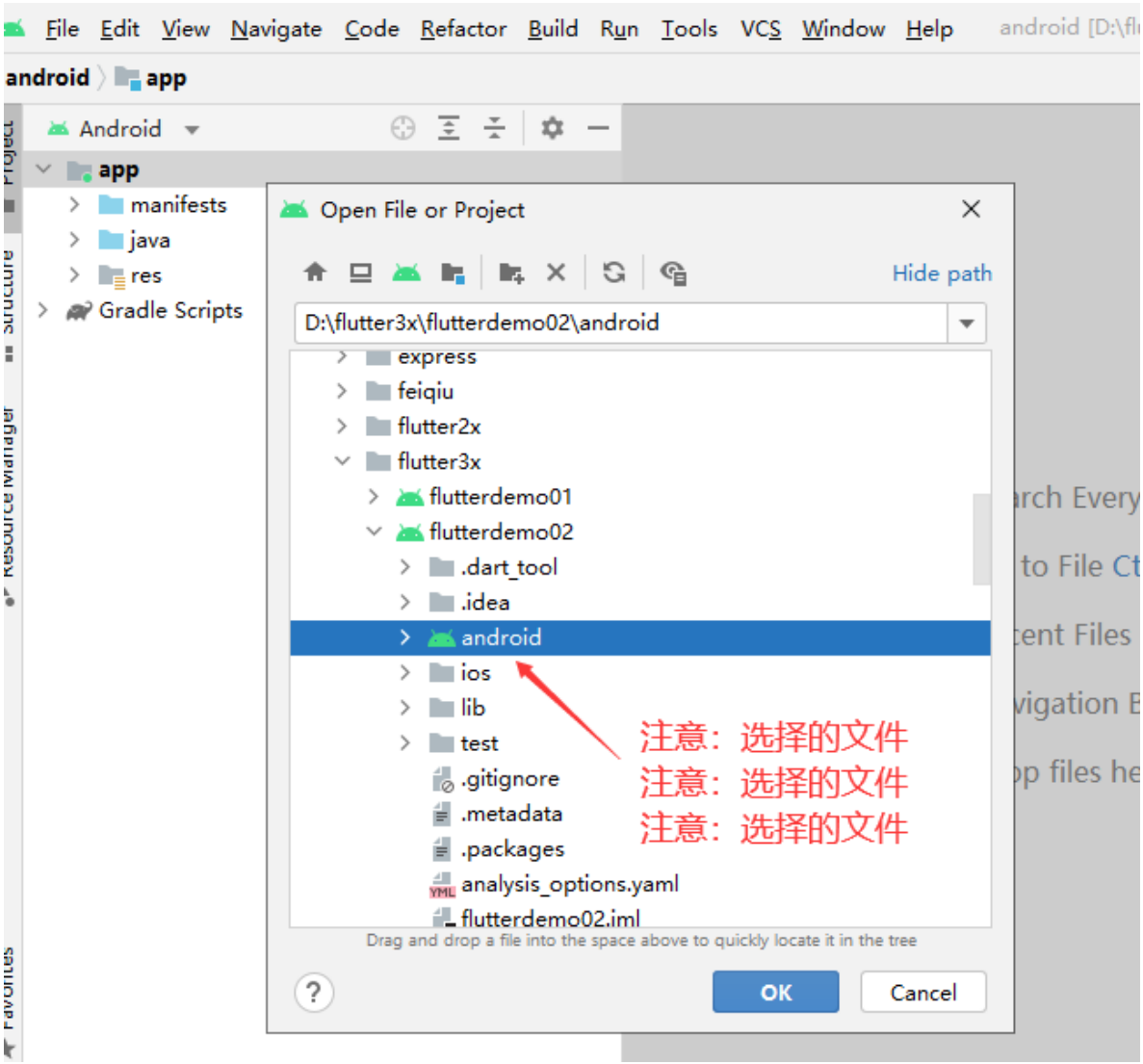
2.7、创建 运行Flutter项目

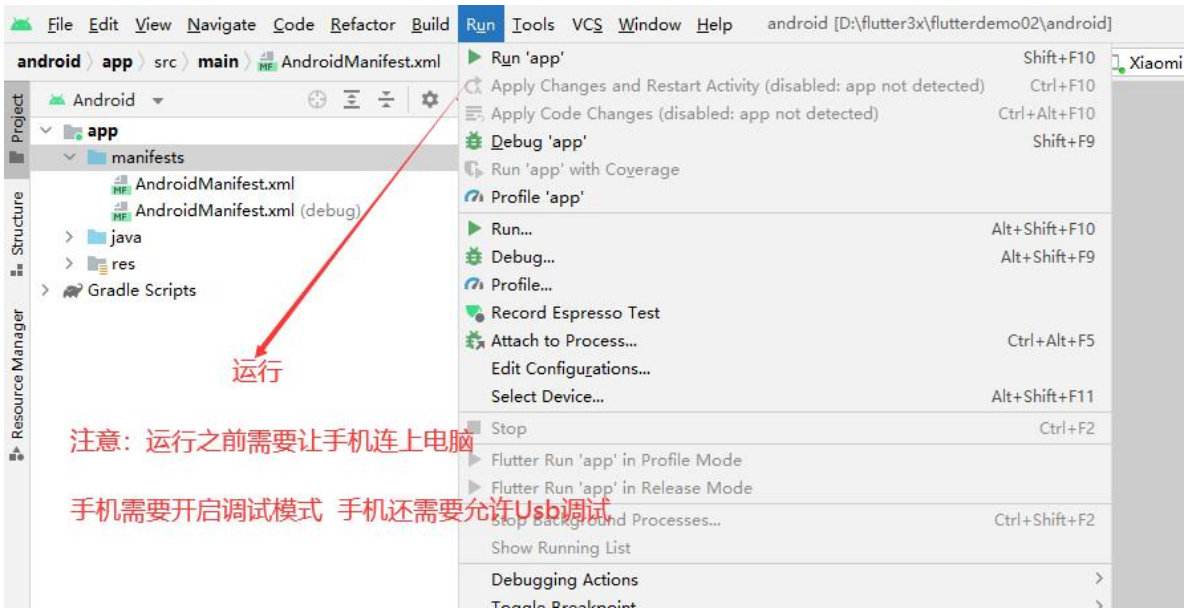



New Project
Java
Gradle
Android
Groovy
Flutter
Dart
Web
Empty Project
Flutter SDK path:
F:\flutter_windows\flutter_windows_3.0.4\flutter
...
选择您的Flutter Sdk目录
Previous
Next
Cancel
Help


New Project
项目名称
Project name:
flutterdemo02
项目存储的目录
Project location:
D:\flutter3x
Description:
A new Flutter project.
Project type:
Application
应用包名称
Organization:
com.example
com.example.flutterdemo02
Android language:
Java
Kotlin
默认
iOS language:
Objective-C
Swift
Platforms:
Android
iOS
Linux
MacOS
Web
Windows
Platform availability might depend on your Flutter SDK channel,
and which desktop platforms have been enabled.
Additional desktop platforms can be enabled by, for example, running
"flutter config --enable-linux-desktop" on the command line.
When created, the new project will run on the selected platforms (others can be added later).
Create project offline
选择要开发的应用对应的平台
如果只是开发Futter app 选择前面两个就可以了
More Settings
Previous
Finish
Cancel
Help


File
Edit
View
Navigate
Code
Refactor
Build
Run
Tools
VCS
New
Open...
Profile or Debug APK
Open Recent
Close Project
Add C++ to Module
Settings...
Ctrl+Alt+S
Project Structure...
Ctrl+Alt+Shift+S
File Properties
Local History
Save All
Ctrl+S
Sync Project with Gradle Files
Project
Structure



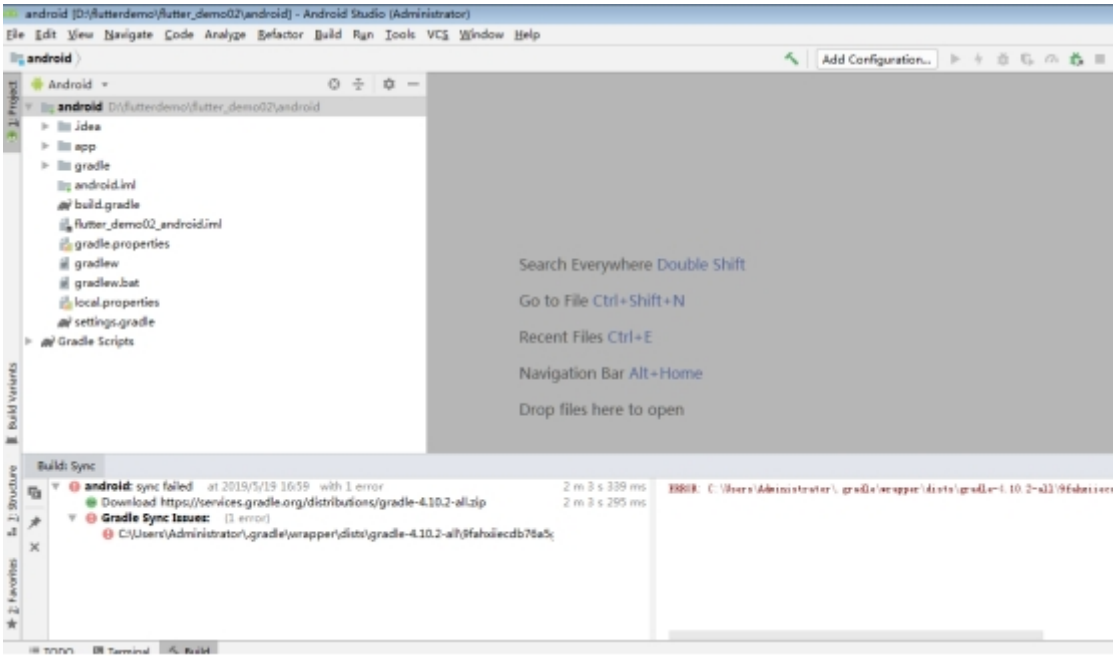


通过 `flutter devices` 可以检测检测可用的设备

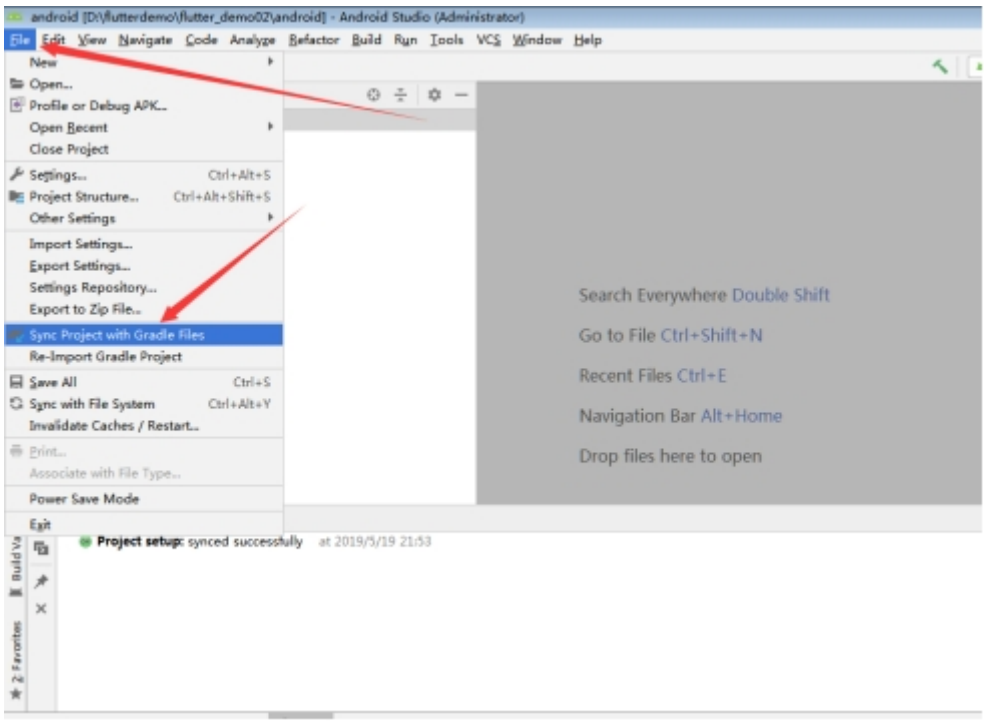
```
C:\Users\itying>flutter devices

4 connected devices:
Redmi K30 (mobile) • beac2700 • android-arm64  • Android 12 (API 31)
Windows (desktop)  • windows  • windows-x64    • Microsoft Windows [版本
10.0.19043.1766]
Chrome (web)       • chrome   • web-javascript • Google Chrome 103.0.5060.114
Edge (web)         • edge     • web-javascript • Microsoft Edge 103.0.1264.49
```

2.8、可能遇到的错误

1、导入项目提示Gradle相关错误



如果报错点击`Flie->Sync Project With Gradle Files` 重新下载Gradle ,这个过程比较慢**10-30分钟**左右。



按照上面方法重试后Gradle 还是失败的解决方法

1、开启手机热点重试

2、谷歌或者百度搜索 “android Gradle编译时下载依赖失败”

2、找不到运行的设备

参考下一讲真机调试，下一讲会详细讲解...

三、Flutter Android真机调试

必备条件：

1、准备一台Android手机

2、手机需要开启调试模式

3、用数据线把手机连上电脑

4、手机要允许电脑进行Usb调试

5、手机对应的sdk版本必须安装

注意：

1、关闭电脑上面的手机助手比如：360手机助手、应用宝等占用adb端口的软件

2、关闭HBuilder之类占用Adb端口的软件

3、数据线一定要可用（可以用360手机助手检测）

四、Flutter虚拟机模拟器调试

必备条件：

1、准备虚拟机模拟器，虚拟机模拟器可以是Android Studio自带的模拟器，也可以是第三方模拟器。

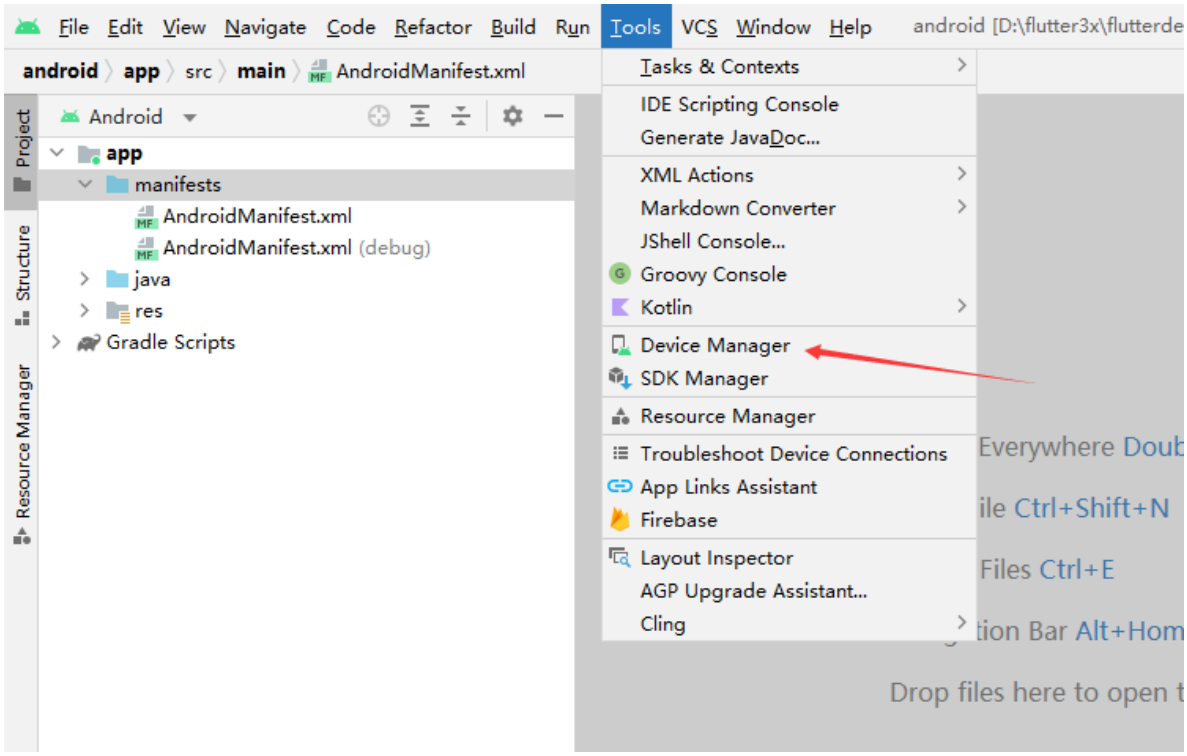
2、模拟器安装好后需要打开

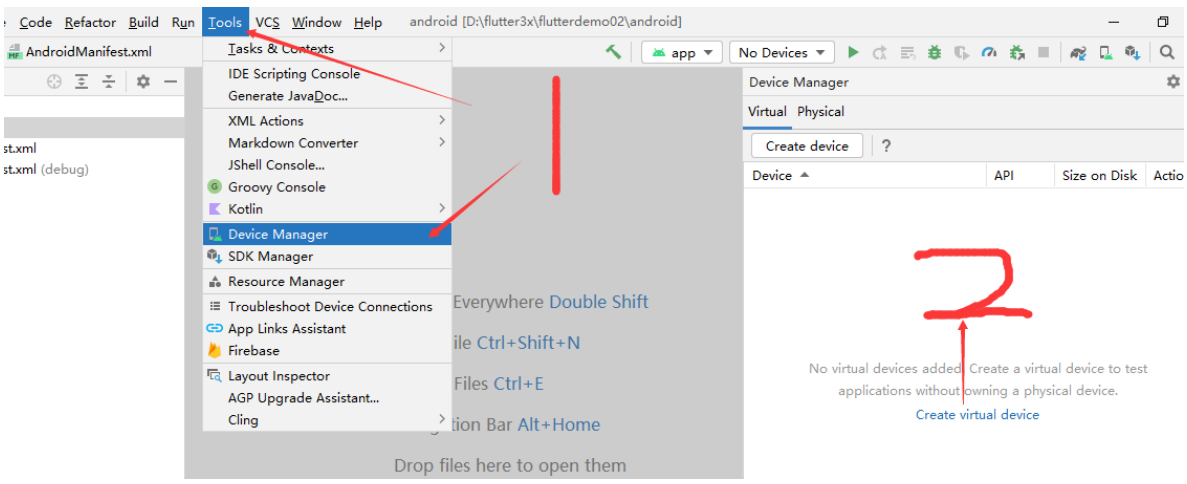
注意：

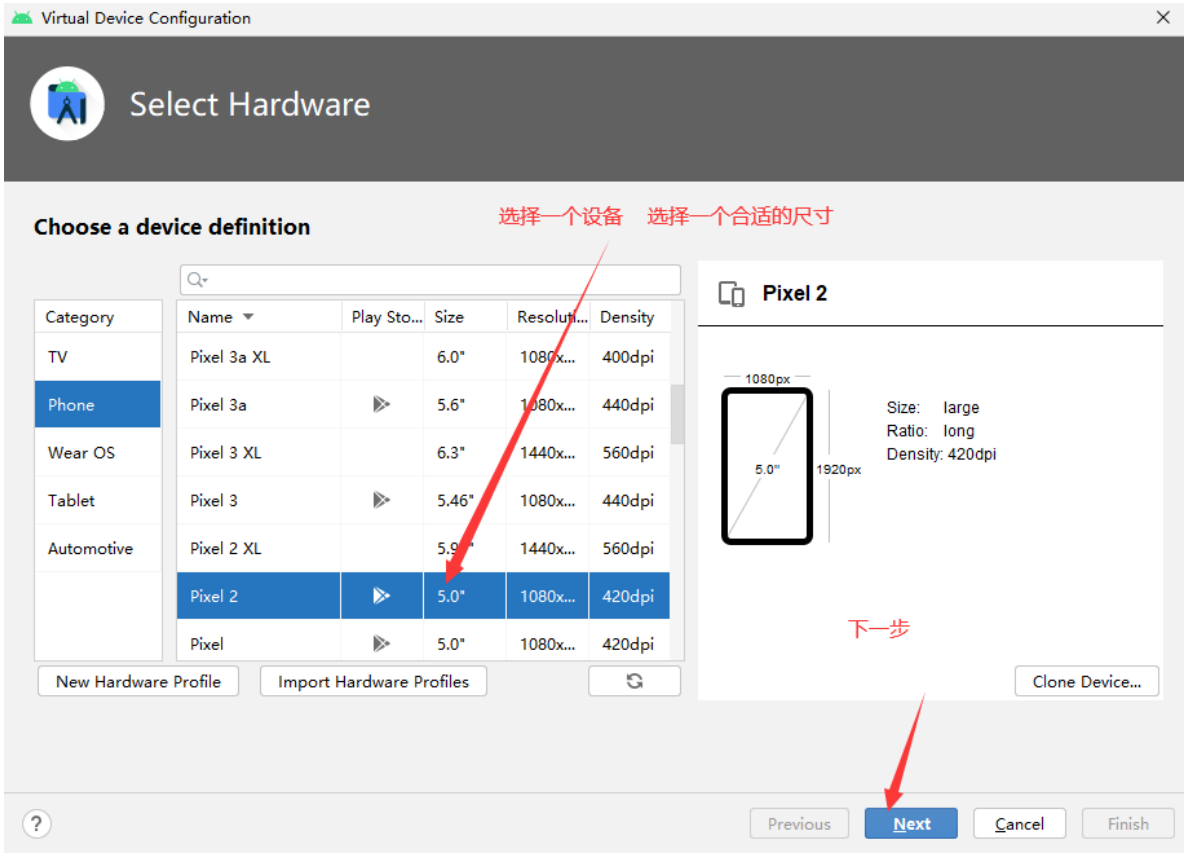
1、关闭电脑上面的手机助手比如：360手机助手、应用宝等占用adb端口的软件

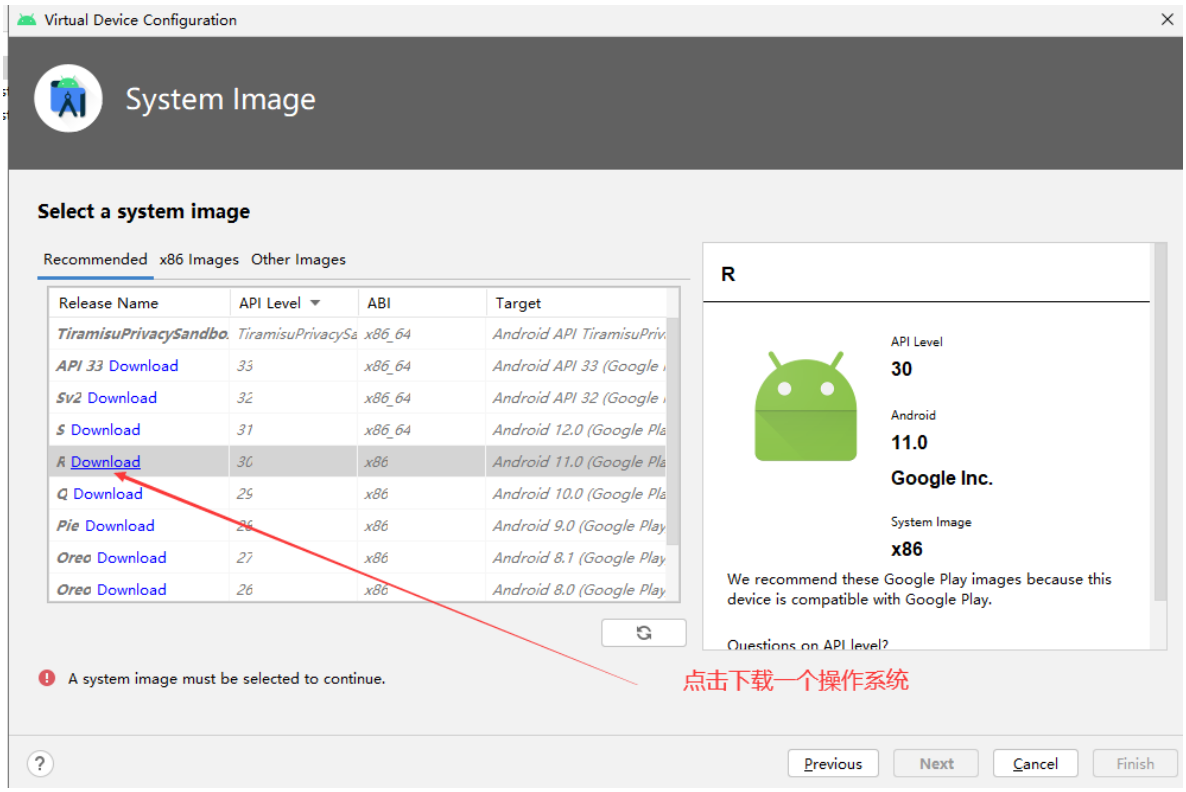
2、关闭HBuilder之类占用Adb端口的软件

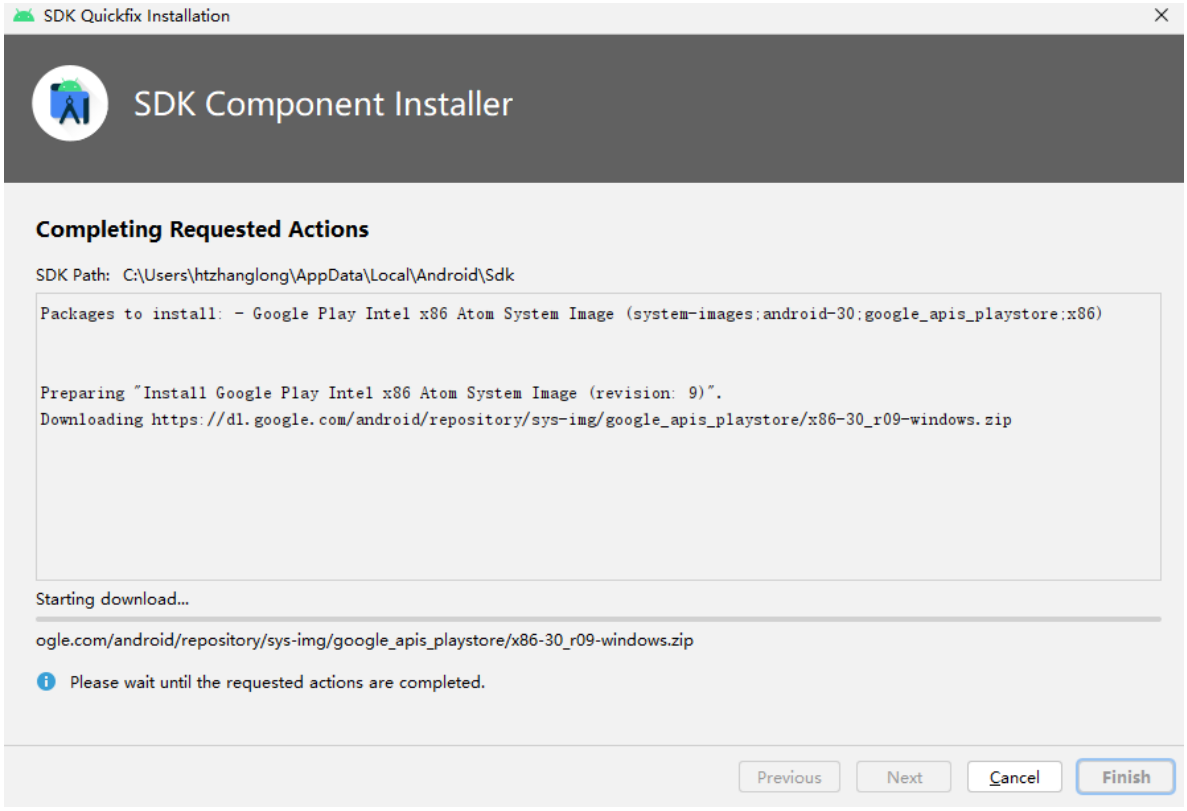
4.1 使用Android Studio自带模拟器

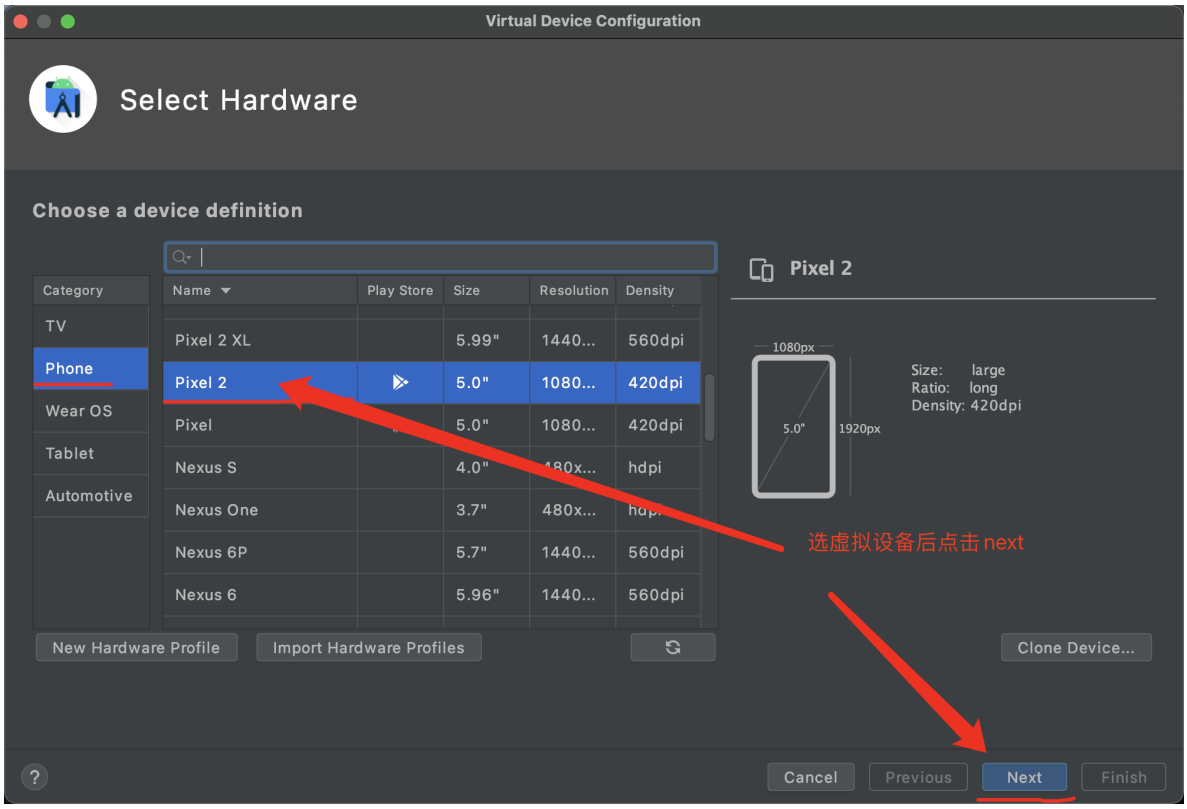


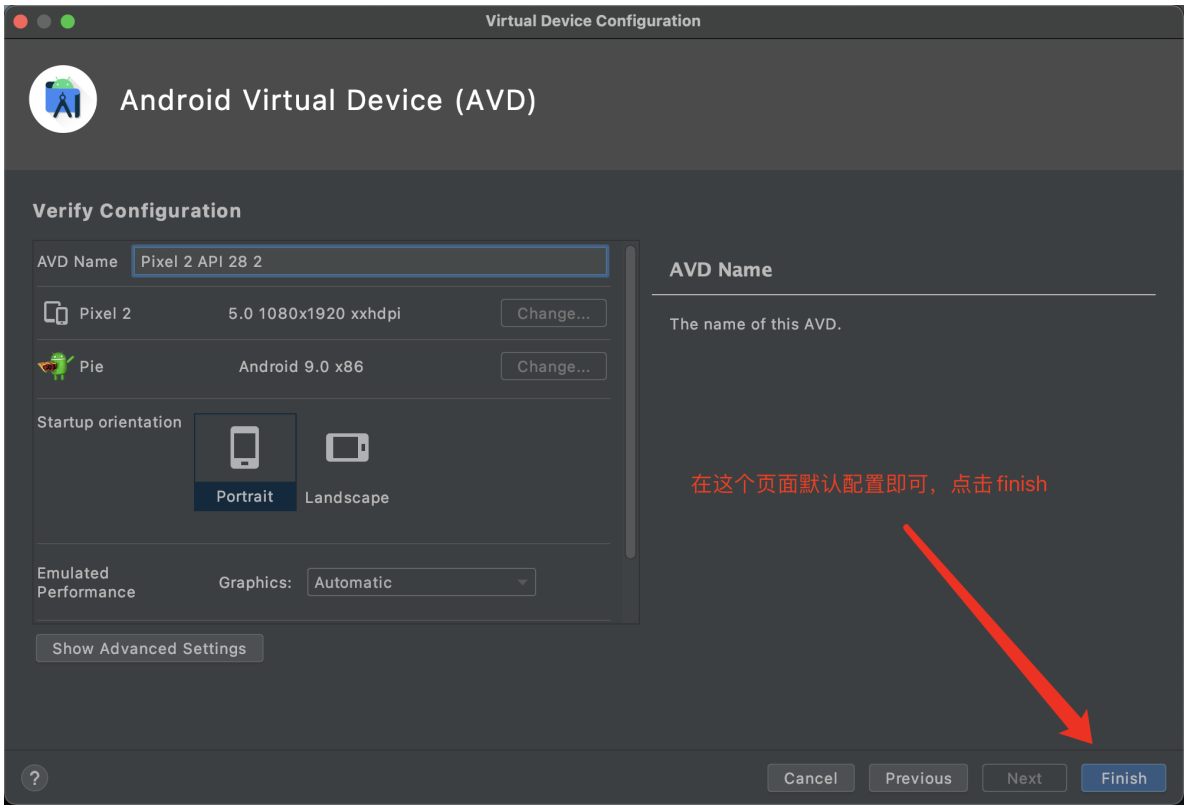


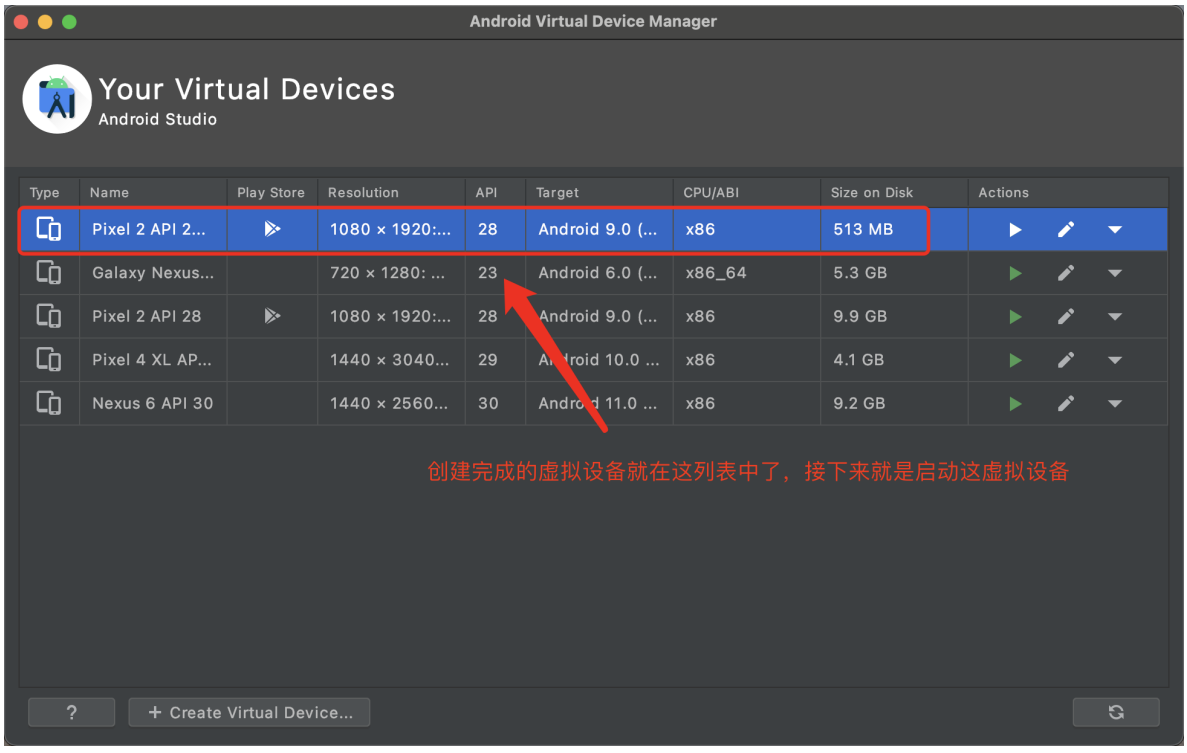

Virtual Device Configuration
Select Hardware
Choose a device definition
选择一个设备
选择一个合适的尺寸
Category
TV
Phone
Wear OS
Tablet
Automotive
Name
Play Sto...
Size
Resoluti...
Density
Pixel 3a XL
6.0"
1080x...
400dpi
Pixel 3a
5.6"
1080x...
440dpi
Pixel 3 XL
6.3"
1440x...
560dpi
Pixel 3
5.46"
1080x...
440dpi
Pixel 2 XL
1440x...
560dpi
Pixel 2
5.0"
1080x...
420dpi
Pixel
5.0"
1080x...
420dpi
New Hardware Profile
Import Hardware Profiles
Pixel 2
1080px
5.0"
1920px
Size: large
Ratio: long
Density: 420dpi
下一步
Clone Device...
Previous
Next
Cancel
Finish

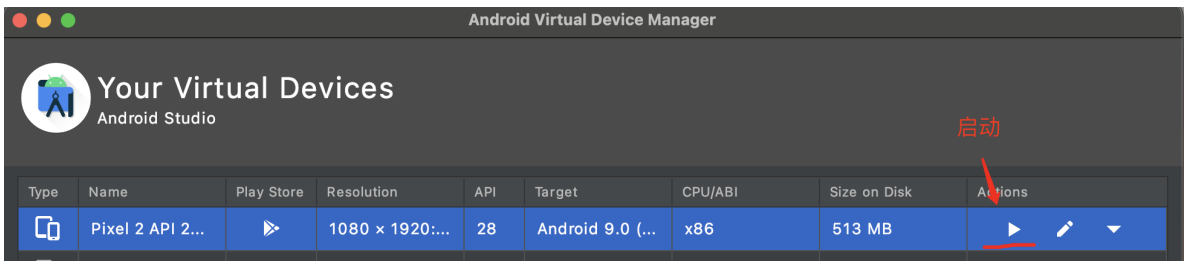

Virtual Device Configuration
System Image
Select a system image
Recommended
x86 Images
Other Images
Release Name
API Level
ABI
Target
TiramisuPrivacySandbo...
API 33 Download
Sv2 Download
S Download
R Download
Q Download
Pie Download
Oreo Download
Oreo Download
A system image must be selected to continue.
点击下载一个操作系统
R
API Level
30
Android
11.0
Google Inc.
System Image
x86
We recommend these Google Play images because this device is compatible with Google Play.
Previous
Next
Cancel
Finish

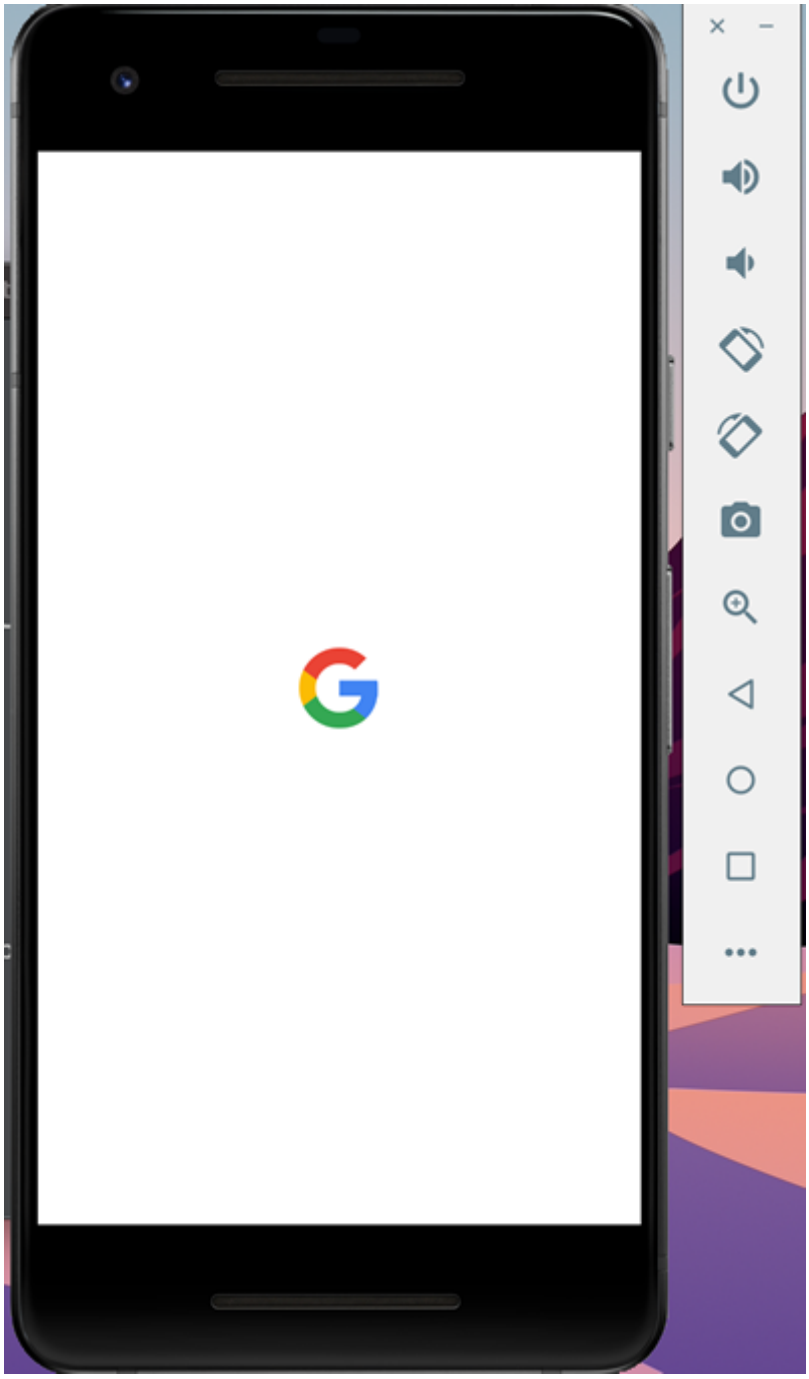

SDK Quickfix Installation
SDK Component Installer
Completing Requested Actions
SDK Path: C:\Users\htzhanglong\AppData\Local\Android\Sdk
Packages to install: - Google Play Intel x86 Atom System Image (system-images;android-30;google_apis_playstore;x86)
Preparing "Install Google Play Intel x86 Atom System Image (revision: 9)".
Downloading https://dl.google.com/android/repository/sys-img/google_apis_playstore/x86-30_r09-windows.zip
Starting download...
ogle.com/android/repository/sys-img/google_apis_playstore/x86-30_r09-windows.zip
Please wait until the requested actions are completed.
Previous
Next
Cancel
Finish


Virtual Device Configuration
Select Hardware
Choose a device definition
Category
TV
Phone
Wear OS
Tablet
Automotive
Name
Play Store
Size
Resolution
Density
Pixel 2 XL 5.99" 1440... 560dpi
Pixel 2 5.0" 1080... 420dpi
Pixel 5.0" 1080... 420dpi
Nexus S 4.0" 480x... hdpi
Nexus One 3.7" 480x...
Nexus 6P 5.7" 1440... 560dpi
Nexus 6 5.96" 1440... 560dpi
Pixel 2
1080px
5.0"
1920px
Size: large
Ratio: long
Density: 420dpi
选虚拟设备后点击 next
New Hardware Profile
Import Hardware Profiles
Clone Device...
Cancel
Previous
Next
Finish


Virtual Device Configuration
Android Virtual Device (AVD)
Verify Configuration
AVD Name
Pixel 2 API 28 2
Pixel 2
5.0 1080x1920 xxhdpi
Change...
Pie
Android 9.0 x86
Change...
Startup orientation
Portrait
Landscape
Emulated Performance
Graphics:
Automatic
Show Advanced Settings
AVD Name
The name of this AVD.
在这个页面默认配置即可，点击 finish
Cancel
Previous
Next
Finish


Android Virtual Device Manager
Your Virtual Devices
Android Studio
Type
Name
Play Store
Resolution
API
Target
CPU/ABI
Size on Disk
Actions
Pixel 2 API 2...
1080 × 1920:...
28
Android 9.0 (...
x86
513 MB
Galaxy Nexus...
720 × 1280: ...
23
Android 6.0 (...
x86_64
5.3 GB
Pixel 2 API 28
1080 × 1920:...
28
Android 9.0 (...
x86
9.9 GB
Pixel 4 XL AP...
1440 × 3040...
29
x86
4.1 GB
Nexus 6 API 30
1440 × 2560...
30
x86
9.2 GB
创建完成的虚拟设备就在这列表中了，接下来就是启动这虚拟设备
+ Create Virtual Device...


Android Virtual Device Manager
Your Virtual Devices
Android Studio
启动
Type
Name
Play Store
Resolution
API
Target
CPU/ABI
Size on Disk
Actions
Pixel 2 API 2...
1080 × 1920:...
28
Android 9.0 (...
x86
513 MB



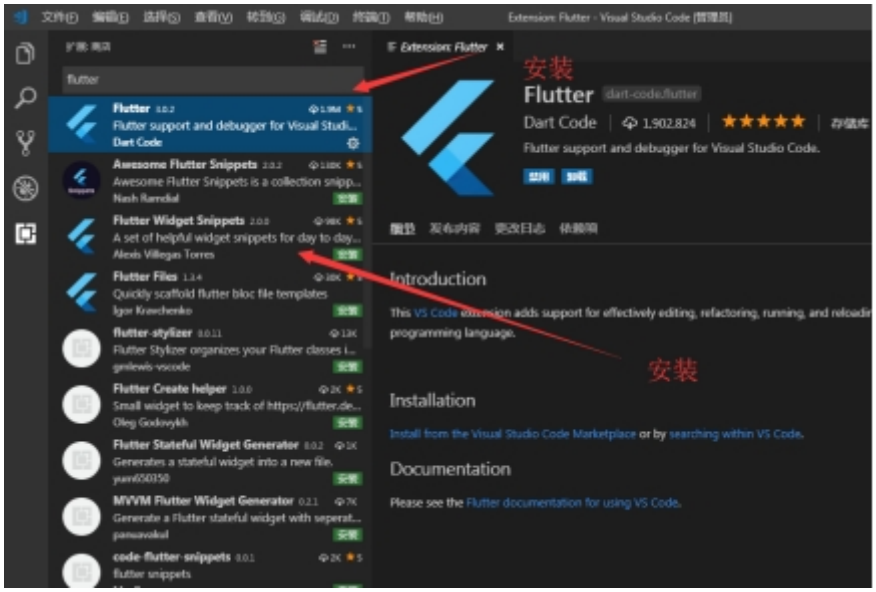
4.2 使用第三方模拟器

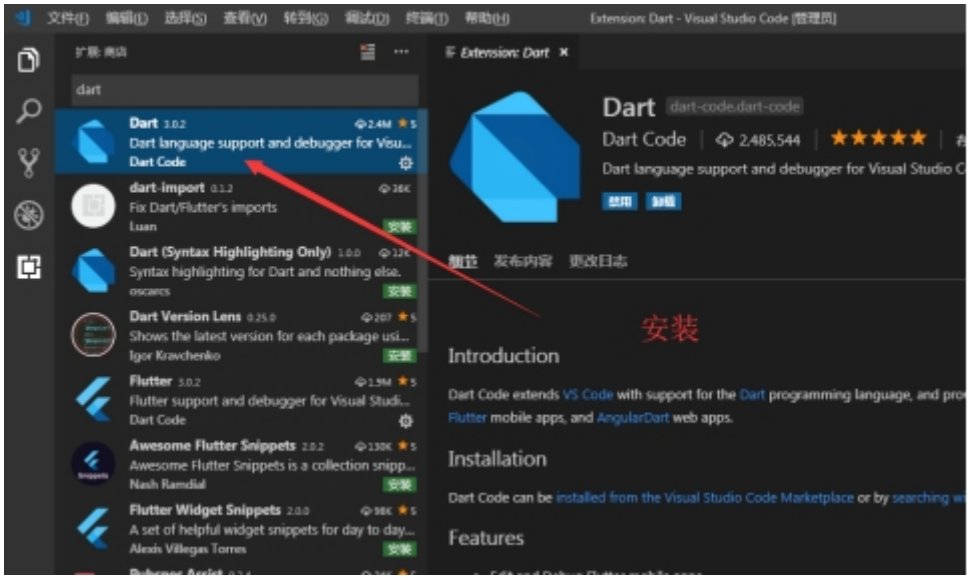
百度或者谷歌里面搜索android 模拟器，然后找一个安装以后参考官方文档配置后就可以使用。

五、Vscode中开发 运行Flutter应用

如果你习惯用Android Studio的话可以直接用Android Studio直接开发Flutter。但是Android Studio比较耗费电脑资源，所以这里我们推荐使用Vscode

5.1、Vscode中安装Flutter插件 Dart插件。





5.2、Vscode中打开flutter项目进行开发

注意：定位到项目目录，不需要定位到项目目录对应的android目录

看教程演示...

5.3、运行Flutter项目、热加载Flutter项目

```
flutter run

flutter run -d all
```

```
Flutter run key commands.
r Hot reload.
R Hot restart.
h List all available interactive commands.
d Detach (terminate "flutter run" but leave application running).
c Clear the screen
q Quit (terminate the application on the device).
```

常用的快捷键

r 键：点击后热加载，也就算是重新加载吧。

R键：热重启项目。

p 键：显示网格，这个可以很好的掌握布局情况，工作中很有用。

o 键：切换android和ios的预览模式。

q 键：退出调试预览模式。

查看设备

```
flutter devices
```

```
$ flutter devices

Flutter assets will be downloaded from https://storage.flutter-io.cn. Make sure
you trust this source!
3 connected devices:
Windows (desktop) • windows • windows-x64    • Microsoft Windows [版本
10.0.19043.1766]
Chrome (web)      • chrome  • web-javascript • Google Chrome 103.0.5060.114
Edge (web)        • edge    • web-javascript • Microsoft Edge 103.0.1264.49
```

运行在所有的设备

```
flutter run -d all
```

指定设备运行

```
flutter run -d chrome
```

六、Mac电脑搭建Flutter Ios环境

6.1、准备工作

1、升级Macos系统为最新系统

2、安装最新的Xcode

3、电脑上面需要安装brew https://brew.sh/

4、安装chrome浏览器（开发web用）

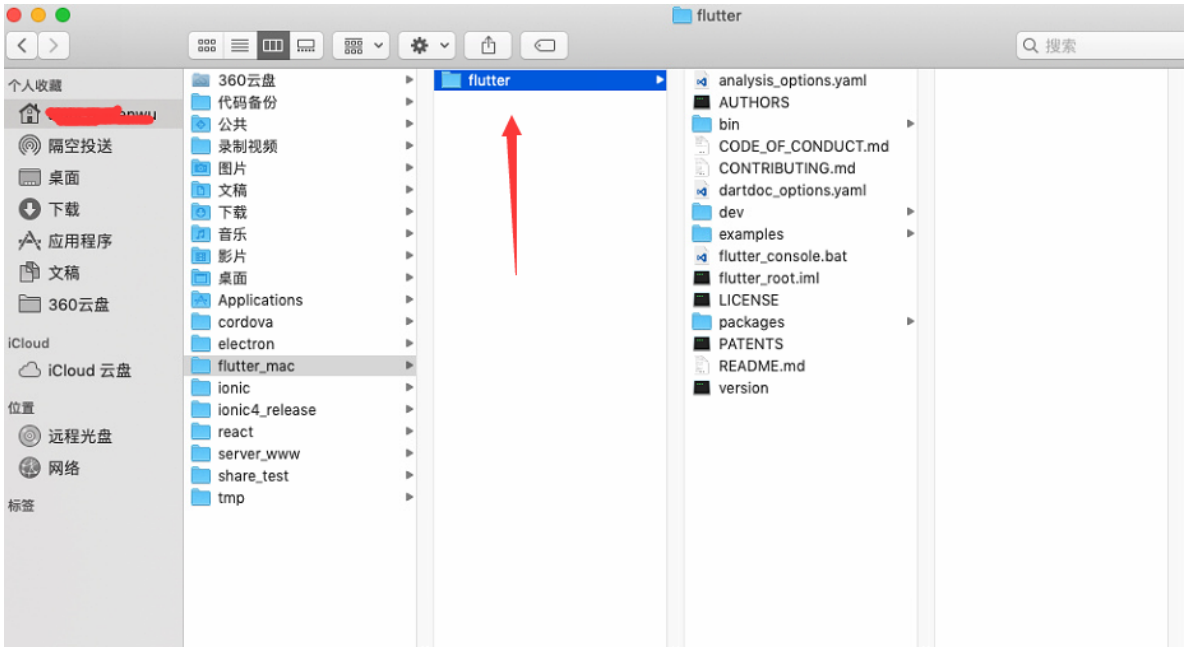
6.2、下载Flutter、配置Flutter环境变量、配置Flutter镜像

6.2.1、下载Flutter SDK

https://flutter.dev/docs/development/tools/sdk/releases?tab=macos

6.2.2、把下载好的Flutter SDK随便减压到你想安装Sdk的目录如

```
/Users/aisheng/flutter_mac/flutte
```



6.2.3、把 Flutter安装目录的bin目录配置到环境变量，然后把Flutter国内镜像也配置到环境变量里面

```
vim ~/.bash_profile
vim ~/.zshrc
```

```
export PATH=/Users/aishengwanwu/flutter_mac/flutter/bin:$PATH

export PUB_HOSTED_URL=https://pub.flutter-io.cn

export FLUTTER_STORAGE_BASE_URL=https://storage.flutter-io.cn
```

让配置环境变量生效

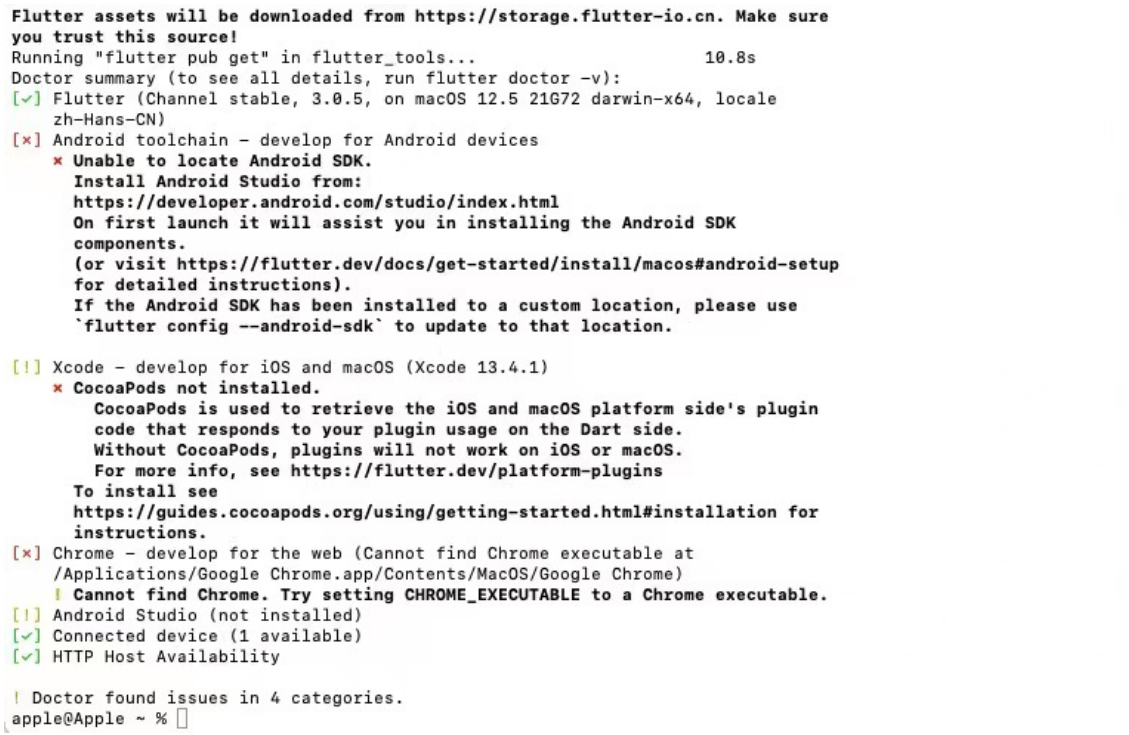
```
source ~/.bash_profile
source ~/.zshrc
```

flutter --version 如果能出来版本说明flutter sdk配置成功。

注意:如果配置完成后输入`flutter --version`告诉你flutter不是内置命令之类的错误的话，可能sdk没有配置成功，也可能sdk下载的时候没有下载全

6.3、运行flutter doctor 命令检测环境

第一次运行 flutter doctor的时候会提示下面错误



相关错误代码如下：

```
    ✗ CocoaPods not installed.
        CocoaPods is used to retrieve the iOS platform side's plugin code that
        responds to your plugin usage on the Dart side.
        Without resolving iOS dependencies with CocoaPods, plugins will not work
        on iOS.
        For more info, see https://flutter.dev/platform-plugins
      To install:
        brew install cocoapods
        pod setup
```

6.4、配置Flutter Xcode Ios环境

6.4.1、如果电脑上面没有安装 brew的话首先第一步需要安装brew

https://brew.sh/

终端执行下面命令

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

```
apple@Apple ~ % /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"
==> Checking for `sudo` access (which may request your password)...
Password:
Sorry, try again.
Password:
Sorry, try again.
Password:
==> This script will install:
/usr/local/bin/brew
/usr/local/share/doc/homebrew
/usr/local/share/man/man1/brew.1
/usr/local/share/zsh/site-functions/_brew
/usr/local/etc/bash_completion.d/brew
/usr/local/Homebrew
==> The following new directories will be created:
/usr/local/bin
/usr/local/etc
/usr/local/include
/usr/local/lib
/usr/local/sbin
/usr/local/share
/usr/local/var
```

```
HEAD is now at 3748bed37 Merge pull request #13680 from Rylan12/fix-arch-dsl
==> Tapping homebrew/core
remote: Enumerating objects: 1246194, done.
Receiving objects:  60% (751943/1246194), 300.15 MiB | 519.00 KiB/s
Receiving objects:  91% (1143621/1246194), 488.68 MiB | 152.00 KiB/s
remote: Total 1246194 (delta 0), reused 0 (delta 0), pack-reused 1246194
Receiving objects: 100% (1246194/1246194), 509.40 MiB | 623.00 KiB/s, done.
Resolving deltas: 100% (858339/858339), done.


From https://github.com/Homebrew/homebrew-core
 * [new branch]              master     -> origin/master
Updating files: 100% (6576/6576), done.
HEAD is now at b90a54785f9 grafana: update 9.0.7 bottle.
Updated 1 tap (homebrew/core).
==> Installation successful!

==> Homebrew has enabled anonymous aggregate formulae and cask analytics.
Read the analytics documentation (and how to opt-out) here:
  https://docs.brew.sh/Analytics
No analytics data has been sent yet (nor will any be during this install run).

==> Homebrew is run entirely by unpaid volunteers. Please consider donating:
  https://github.com/Homebrew/brew#donations

==> Next steps:
- Run brew help to get started
- Further documentation:
    https://docs.brew.sh
```

6.4.2、分别执行下面命令

```
brew install cocoapods
pod setup

sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
sudo xcodebuild -runFirstLaunch
```

```
apple@Apple ~ % brew install cocoapods
==> Downloading https://ghcr.io/v2/homebrew/core/cocoapods/manifests/1.11.3
######################################################################## 100.0%
==> Downloading https://ghcr.io/v2/homebrew/core/cocoapods/blobs/sha256:92ea102a
==> Downloading from https://pkg-containers.githubusercontent.com/ghcr1/blobs/sh
######################################################################## 100.0%
==> Pouring cocoapods--1.11.3.monterey.bottle.tar.gz
🍺  /usr/local/Cellar/cocoapods/1.11.3: 14,135 files, 29.9MB
==> Running `brew cleanup cocoapods`...
Disable this behaviour by setting HOMEBREW_NO_INSTALL_CLEANUP.
Hide these hints with HOMEBREW_NO_ENV_HINTS (see `man brew`).
apple@Apple ~ % pod setup
Setup completed
apple@Apple ~ %
```

```
apple@Apple ~ % flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.0.5, on macOS 12.5 21G72 darwin-x64, locale
    zh-Hans-CN)
[✗] Android toolchain - develop for Android devices
    ✗ Unable to locate Android SDK.
      Install Android Studio from:
      https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK
      components.
      (or visit https://flutter.dev/docs/get-started/install/macos#android-setup
      for detailed instructions).
      If the Android SDK has been installed to a custom location, please use
      `flutter config --android-sdk` to update to that location.

[!] Xcode - develop for iOS and macOS
    ✗ Xcode installation is incomplete; a full installation is necessary for iOS
      development.
      Download at: https://developer.apple.com/xcode/download/
      Or install Xcode via the App Store.
      Once installed, run:
        sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
        sudo xcodebuild -runFirstLaunch
[✗] Chrome - develop for the web (Cannot find Chrome executable at
    /Applications/Google Chrome.app/Contents/MacOS/Google Chrome)
    ! Cannot find Chrome. Try setting CHROME_EXECUTABLE to a Chrome executable.
[!] Android Studio (not installed)
[✓] Connected device (1 available)
[✓] HTTP Host Availability
```

执行：

```
sudo xcode-select --switch /Applications/Xcode.app/Contents/Developer
sudo xcodebuild -runFirstLaunch
```

```
apple@Apple ~ % sudo xcode-select --switch /Applications/Xcode.app/Contents/Deve
loper
Password:
apple@Apple ~ % sudo xcodebuild -runFirstLaunch
apple@Apple ~ %
```

注意：如果运行命令失败请运行 brew doctor并按照说明解决问题。

执行完成上面命令后然后重新运行：`flutter doctor` 如果出来下图表示ios的环境配置完成

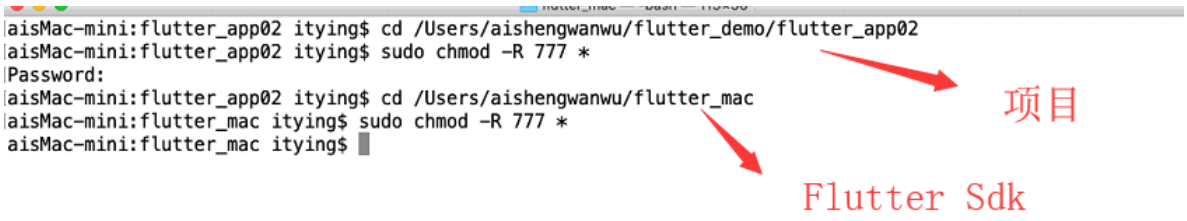
```
[apple@Apple ~ % flutter doctor
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.0.5, on macOS 12.5 21G72 darwin-x64, locale
    zh-Hans-CN)
[✗] Android toolchain - develop for Android devices
    ✗ Unable to locate Android SDK.
      Install Android Studio from:
      https://developer.android.com/studio/index.html
      On first launch it will assist you in installing the Android SDK
      components.
      (or visit https://flutter.dev/docs/get-started/install/macos#android-setup
      for detailed instructions).
      If the Android SDK has been installed to a custom location, please use
      `flutter config --android-sdk` to update to that location.

[✓] Xcode - develop for iOS and macOS (Xcode 13.4.1)
[✓] Chrome - develop for the web
[!] Android Studio (not installed)
[✓] Connected device (2 available)
[✓] HTTP Host Availability

! Doctor found issues in 2 categories.
```

6.5、Mac 上面创建Flutter ios项目

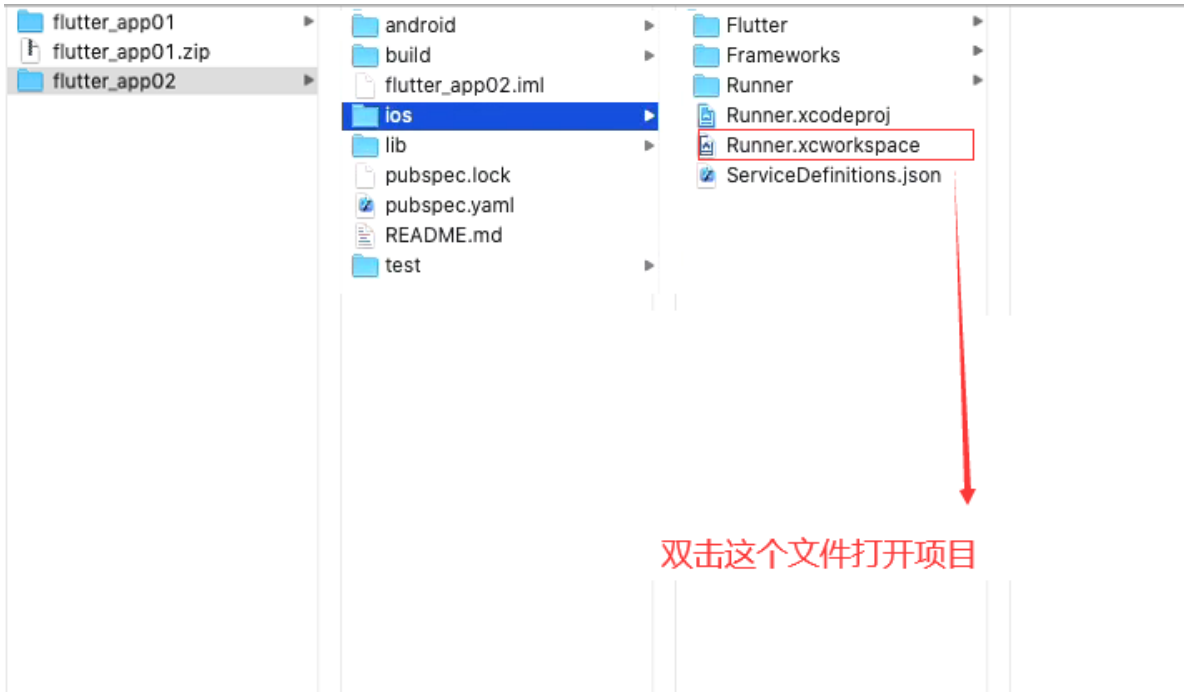
```
sudo flutter create flutterdemo
```

```
aisMac-mini:flutter_demo itying$ sudo flutter create flutter_app02
Password:
   Woah! You appear to be trying to run flutter as root.
   We strongly recommend running the flutter tool without superuser privileges.
  /
📎
Creating project flutter_app02...
  flutter_app02/ios/Runner.xcworkspace/contents.xcworkspacedata (created)
  flutter_app02/ios/Runner/Info.plist (created)
  flutter_app02/ios/Runner/Assets.xcassets/LaunchImage.imageset/LaunchImage@2x.p
  ng (created)
  flutter_app02/ios/Runner/Assets.xcassets/LaunchImage.imageset/LaunchImage@3x.p
  ng (created)
  flutter_app02/ios/Runner/Assets.xcassets/LaunchImage.imageset/README.md
  (created)
  flutter_app02/ios/Runner/Assets.xcassets/LaunchImage.imageset/Contents.json
  (created)
  flutter_app02/ios/Runner/Assets.xcassets/LaunchImage.imageset/LaunchImage.png
  (created)
  flutter_app02/ios/Runner/Assets.xcassets/AppIcon.appiconset/Icon-App-76x76@2x.
  png (created)
  flutter_app02/ios/Runner/Assets.xcassets/AppIcon.appiconset/Icon-App-29x29@1x.
  png (created)
```

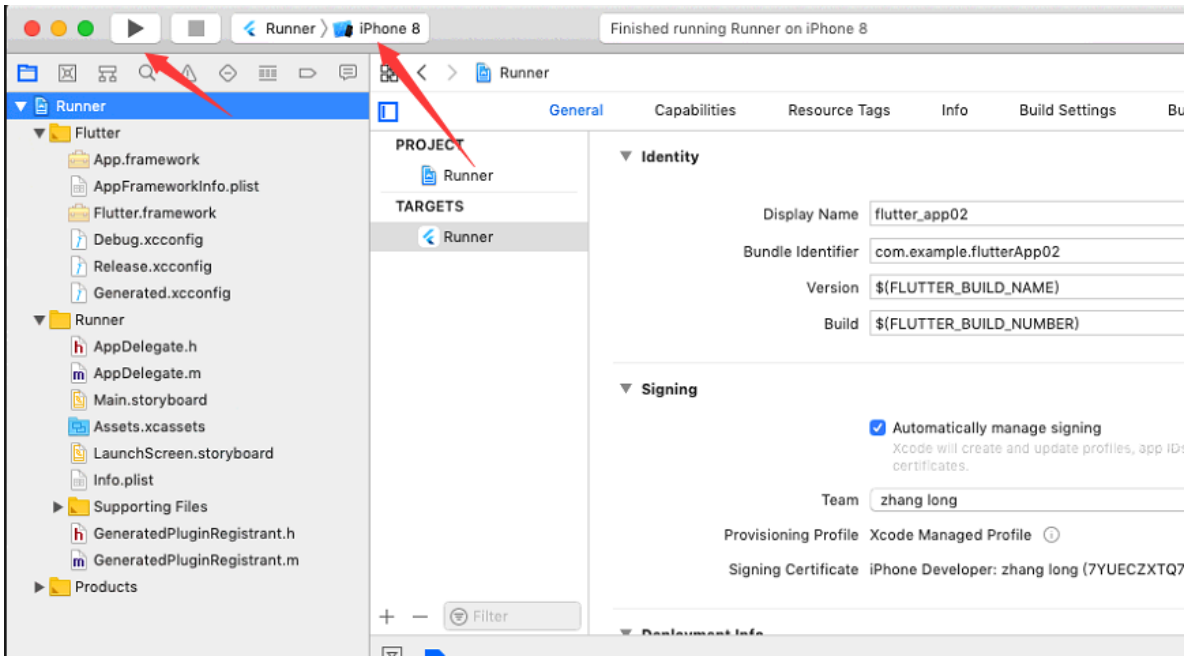
6.6、修改Flutter Sdk目录的权限以及项目目录的权限



6.7、Xcode打开flutter项目 模拟器运行项目

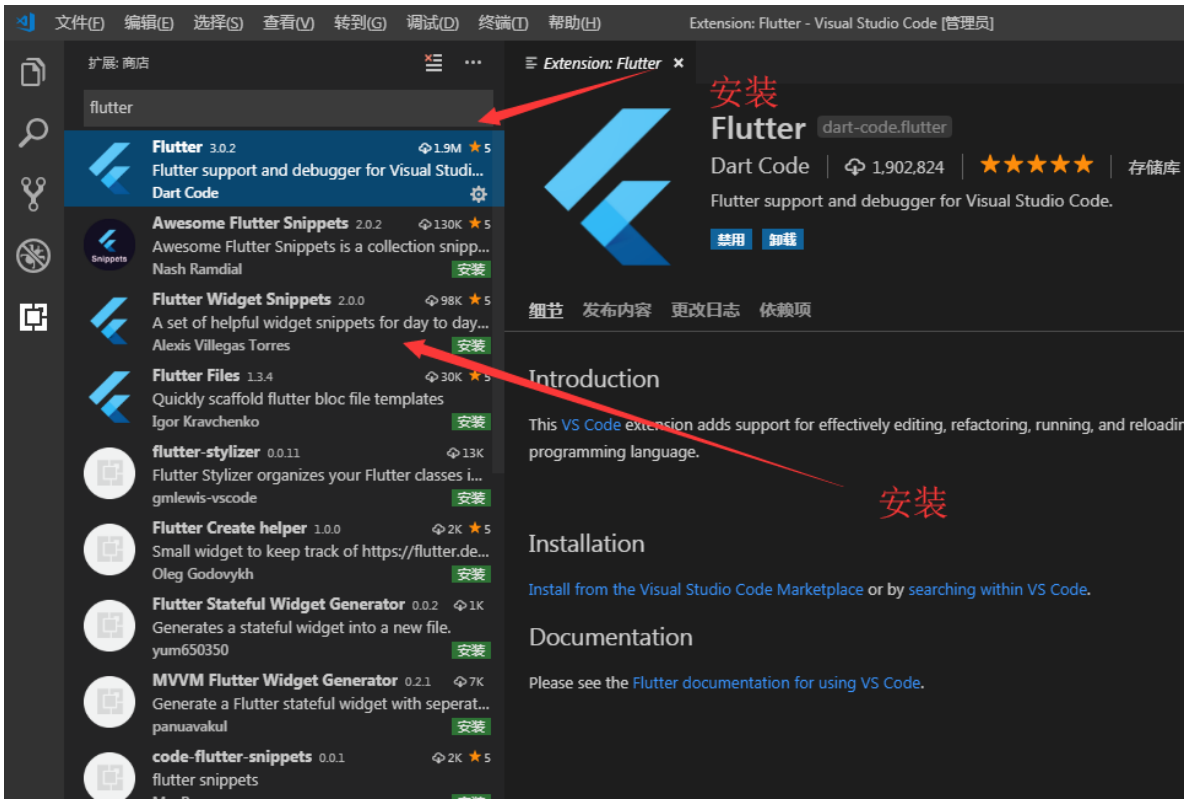


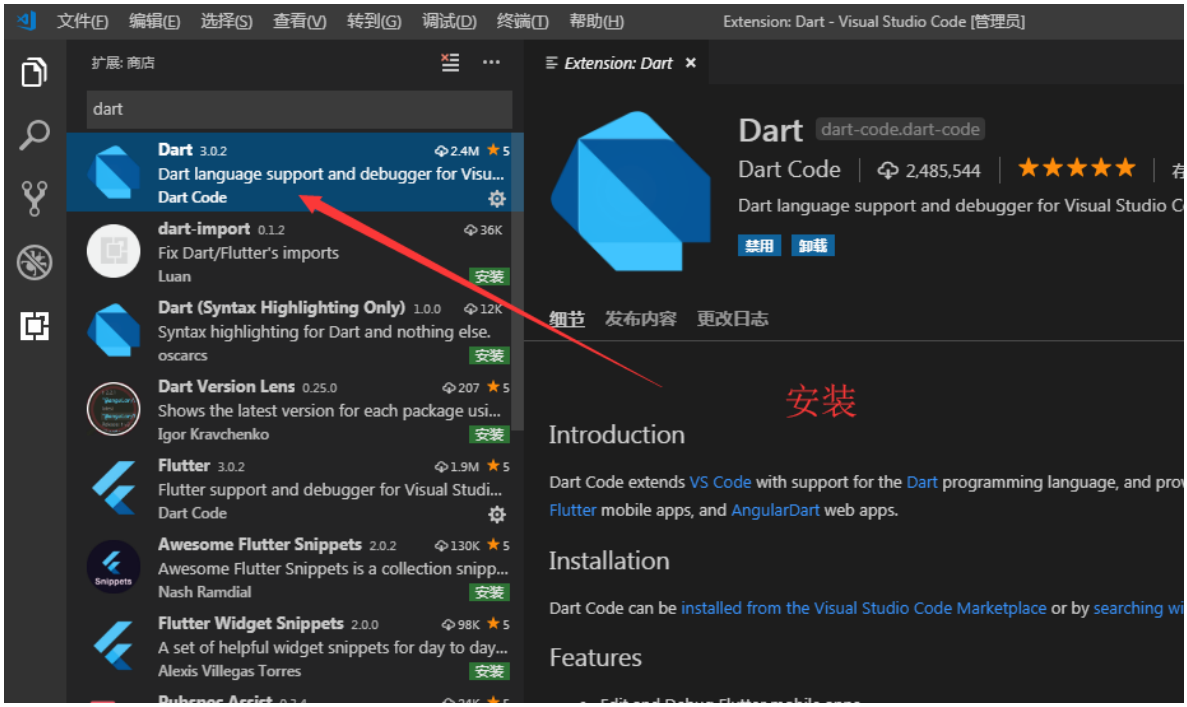
注意：打开项目之前一定要修改权限



6.8、 在Vscode中配置 开发Flutter项目

1、Vscode中安装Flutter插件 Dart插件





2、Vscode中打开flutter项目进行开发

3、运行Flutter项目

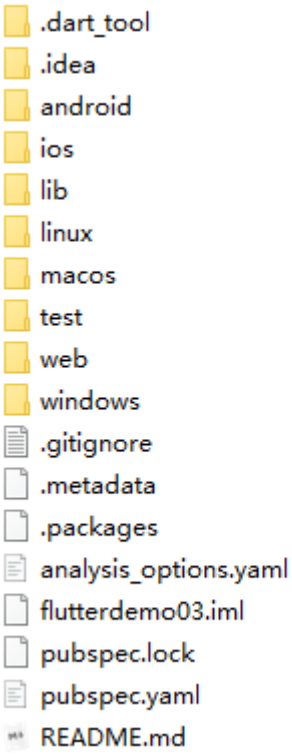
```
flutter run
```

r 键：点击后热加载，也就算是重新加载吧。
p 键：显示网格，这个可以很好的掌握布局情况，工作中很有用。
o 键：切换android和ios的预览模式。
q 键：退出调试预览模式。

七、手把手教您学会写第一个Flutter应用

```
flutter create flutterdemo03
```

7.1、Flutter目录结构介绍



我们着重需要注意一下几个文件夹,其他的暂时不用理会

文件夹	作用
android	android平台相关代码
ios	ios平台相关代码
linux	Linux平台相关的代码
macos	macos平台相关的代码
web	web相关的代码
windows	windows相关的代码
lib	flutter相关代码，我们编写的代码就在这个文件夹
test	用于存放测试代码
pubspec.yaml	配置文件，一般存放一些第三方库的依赖。
analysis_options.yaml	分析dart语法的文件，老项目升级成新项目有警告信息的话可以删掉此文件

7.2、Flutter入口文件、入口方法

每一个flutter项目的lib目录里面都有一个main.dart这个文件就是flutter的入口文件

main.dart里面的

```
void main(){
  runApp(MyApp());
}

也可以简写
void main()=>runApp(MyApp());
```

其中的main方法是dart的入口方法。runApp方法是flutter的入口方法。

MyApp是自定义的一个组件。

7.3、Flutter第一个Demo Center组件的使用

demo1

```
import 'package:flutter/material.dart';
void main() {
  runApp(const Center(
    child: Text(
      "我是一个文本",
      textDirection: TextDirection.ltr,
    ),
  ));
}
```

demo2

给Text组件增加一些装饰

```
import 'package:flutter/material.dart';

void main() {
  runApp(const Center(
    child: Text(
      "我是一个文本",
      textDirection: TextDirection.ltr,
      style: TextStyle(
        fontSize: 40.0,
        // color: Colors.yellow,
        color: Color.fromRGBO(244, 233, 121, 0.5),
      ),
    ),
  ));
}
```

demo3

```
import 'package:flutter/material.dart';
void main() {
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: const Text("你好Flutter")),
      body: const Center(
        child: Text(
          "我是一个文本",
          textDirection: TextDirection.ltr,
          style: TextStyle(
            fontSize: 40.0,
            // color: Colors.yellow,
            color: Color.fromRGBO(244, 233, 121, 0.5),
          ),
        ),
      ),
    ),
  ));
}
```

demo4

```
import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: const Text("你好Flutter")),
      body:const HomeWidget(),
    ),
  ));
}
```

```
class HomeWidget extends StatelessWidget{
  const HomeWidget({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return const Center(
        child: Text(
          "我是一个文本",
          textDirection: TextDirection.ltr,
          style: TextStyle(
            fontSize: 40.0,
            // color: Colors.yellow,
            color: Color.fromRGBO(244, 233, 121, 0.5),
          ),
        ),
      );
  }

}
```

7.4、Flutter把内容单独抽离成一个组件

在Flutter中自定义组件其实就是一个类，这个类需要继承StatelessWidget/StatefulWidget

前期我们都继承StatelessWidget。后期给大家讲StatefulWidget的使用。

StatelessWidget 是无状态组件，状态不可变的widget

StatefulWidget 是有状态组件，持有的状态可能在widget生命周期改变

```
import 'package:flutter/material.dart';

void main(){
  runApp(const MyApp());
}
class MyApp extends StatelessWidget{
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return const Center(
      child: Text(
        "我是一个文本内容",
        textDirection: TextDirection.ltr,
      ),
    );
  }
}
```

```
import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp(
    home: Scaffold(
```

```
      appBar: AppBar(title: const Text("你好Flutter")),
      body:const HomeWidget(),
    ),
  ));
}

class HomeWidget extends StatelessWidget{
  const HomeWidget({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return const Center(
        child: Text(
          "我是一个文本",
          textDirection: TextDirection.ltr,
          style: TextStyle(
            fontSize: 40.0,
            // color: Colors.yellow,
            color: Color.fromRGBO(244, 233, 121, 0.5),
          ),
        ),
      );
  }

}
```

7.5、件用MaterialApp 和 Scaffold两个组件装饰App

1、MaterialApp

MaterialApp是一个方便的Widget，它封装了应用程序实现Material Design所需要的一些Widget。一般作为顶层widget使用。

常用的属性:
home（主页）
title（标题）
color（颜色）
theme（主题）
routes（路由）
...

2、Scaffold

Scaffold是Material Design布局结构的基本实现。此类提供了用于显示drawer、snackbar和底部sheet的API。

Scaffold 有下面几个主要属性:

appBar - 显示在界面顶部的一个 AppBar。
body - 当前界面所显示的主要内容 Widget。
drawer - 抽屉菜单控件。
...

八、Container容器组件

名称	功能
alignment	topCenter：顶部居中对齐topLeft：顶部左对齐topRight：顶部右对齐center：水平垂直居中对齐centerLeft：垂直居中水平居左对齐centerRight：垂直居中水平居右对齐bottomCenter底部居中对齐bottomLeft：底部居左对齐 bottomRight：底部居右对齐
decoration	decoration: BoxDecoration(color: Colors.blue, border: Border.all(color: Colors.red, width: 2.0), borderRadius:BorderRadius.circular((8)),// 圆角， boxShadow: [BoxShadow(color: Colors.blue, offset: Offset(2.0, 2.0), blurRadius: 10.0,)],) //LinearGradient 背景线性渐变 RadialGradient径向渐变 gradient: LinearGradient(colors: [Colors.red, Colors.orange],),
margin	margin属性是表示Container与外部其他组件的距离。 EdgeInsets.all(20.0),
padding	padding就是Container的内边距，指Container边缘与Child之间的距离 padding:EdgeInsets.all(10.0)
transform	让Container容易进行一些旋转之类的transform: Matrix4.rotationZ(0.2)
height	容器高度
width	容器宽度
child	容器子元素

示例代码01：


```
//代码块 importM
import 'package:flutter/material.dart';

void main() {
```

```
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: const Text("你好Flutter")),
      body: const MyApp(),
    ),
  ));
}

// 代码块  statelessW
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        alignment: Alignment.center,
        height: 200,
        width: 200,
        decoration: const BoxDecoration(
          color: Colors.yellow,
        ),
        child: const Text(
          "你好Flutter",
          style:TextStyle(
            fontSize: 20
          ) ,
        ),
      ),
    );
  }
}
```

示例代码02：


```
//代码块 importM
import 'package:flutter/material.dart';
```

```
void main() {
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: const Text("你好Flutter")),
      body: const MyApp(),
    ),
  ));
}

// 代码块  statelessW
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        alignment: Alignment.center,
        height: 200,
        width: 200,
        decoration:  BoxDecoration(
          color: Colors.yellow,
          gradient: const LinearGradient(
            //LinearGradient 背景线性渐变   RadialGradient径向渐变
            colors: [Colors.red, Colors.orange],
          ),
          boxShadow:const [
            //卡片阴影
            BoxShadow(
              color: Colors.blue,
              offset: Offset(2.0, 2.0),
              blurRadius: 10.0,
            )
          ],
          border: Border.all(
            color: Colors.black,
            width: 1
          )
        ),
        transform:Matrix4.rotationZ(.2),
        child: const Text(
          "你好Flutter",
          style: TextStyle(fontSize: 20),
        ),

      ),
    );
  }
}
```

示例代码03： 通过Container创建一个按钮


```
//代码块 importM
import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: const Text("你好Flutter")),
      body: const MyApp(),
    ),
  ));
}

// 代码块  statelessW
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        alignment: Alignment.center,
        height: 40,
        width: 200,
        decoration:  BoxDecoration(
          color: Colors.blue,
          borderRadius: BorderRadius.circular(15)
        ),
        child: const Text(
          "按钮",
          style: TextStyle(fontSize: 20),
        ),
      ),
    );
  }
}
```

示例代码04： padding 和maring

padding 是让容器和里面的元素有相应的间距，margin是让容器和容器外部的其他容器有相应的间距

```
Container(
  margin: EdgeInsets.all(20.0), //容器外补白
  color: Colors.orange,
  child: Text("Hello world!"),
),
Container(
  padding: EdgeInsets.all(20.0), //容器内补白
  color: Colors.orange,
  child: Text("Hello world!"),
),
```

九、Text组件详解

名称	功能
textAlign	文本对齐方式（center居中，left左对齐，right右对齐，justfy两端对齐）
textDirection	文本方向（ltr从左至右，rtl从右至左）
overflow	文字超出屏幕之后的处理方式（clip裁剪，fade渐隐，ellipsis省略号）
textScaleFactor	字体显示倍率
maxLines	文字显示最大行数
style	字体的样式设置

下面是 TextStyle 的参数：

名称	功能
decoration	文字装饰线（none没有线，lineThrough删除线，overline上划线，underline下划线）
decorationColor	文字装饰线颜色
decorationStyle	文字装饰线风格（[dashed,dotted]虚线，double两根线，solid一根实线，wavy波浪线）
wordSpacing	单词间隙（如果是负值，会让单词变得更紧凑
letterSpacing	字母间隙（如果是负值，会让字母变得更紧凑）
fontStyle	文字样式（italic斜体，normal正常体）
fontSize	文字大小
color	文字颜色
fontWeight	字体粗细（bold粗体，normal正常体）

更多参数：https://docs.flutter.io/flutter/painting/TextStyle-class.html

示例代码：

```
//代码块 importM
import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: const Text("你好Flutter")),
      body: const MyApp(),
    ),
  ));
}

// 代码块  statelessW
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        alignment: Alignment.center,
        height: 200,
        width: 200,
        decoration: BoxDecoration(
            color: Colors.yellow,
            gradient: const LinearGradient(
              //LinearGradient 背景线性渐变   RadialGradient径向渐变
              colors: [Colors.red, Colors.orange],
            ),
            boxShadow: const [
              //卡片阴影
              BoxShadow(
                color: Colors.blue,
                offset: Offset(2.0, 2.0),
                blurRadius: 10.0,
              )
            ],
            border: Border.all(color: Colors.black, width: 1)),
        transform: Matrix4.rotationZ(.2),
        child: const Text('各位同学大家好我是主讲老师大地，各位同学大家好我是主讲老师大
地',
            textAlign: TextAlign.left,
            overflow: TextOverflow.ellipsis,
            // overflow:TextOverflow.fade ,
            maxLines: 2,
            textScaleFactor: 1.8,
            style: TextStyle(
                fontSize: 16.0,
                color: Colors.black,
                // color:Color.fromARGB(a, r, g, b)
                fontWeight: FontWeight.w800,
                fontStyle: FontStyle.italic,
                decoration: TextDecoration.lineThrough,
```

```
                decorationColor: Colors.white,
                decorationStyle: TextDecorationStyle.dashed,
                letterSpacing: 5.0)),
      ),
    );
  }
}
```

十、图片组件详解

10.1、图片组件介绍

Flutter 中，我们可以通过`Image`组件来加载并显示图片`Image`的数据源可以是asset、文件、内存以及网络 。

这里我们主要给大家讲两个

Image.asset， 本地图片
Image.network 远程图片

Image组件的常用属性:

名称	类型	说明
alignment	Alignment	图片的对齐方式
color和colorBlendMode		设置图片的背景颜色，通常和colorBlendMode配合一起使用，这样可以是图片颜色和背景色混合。上面的图片就是进行了颜色的混合，绿色背景和图片红色的混合
fit	BoxFit	fit属性用来控制图片的拉伸和挤压，这都是根据父容器来的。 BoxFit.fill:全图显示，图片会被拉伸，并充满父容器。 BoxFit.contain:全图显示，显示原比例，可能会有空隙。 BoxFit.cover：显示可能拉伸，可能裁切，充满（图片要充满整个容器，还不变形）。 BoxFit.fitWidth：宽度充满（横向充满），显示可能拉伸，可能裁切。 BoxFit.fitHeight：高度充满（竖向充满）,显示可能拉伸，可能裁切。 BoxFit.scaleDown：效果和contain差不多，但是此属性不允许显示超过源图片大小，可小不可大。
repeat	平铺	ImageRepeat.repeat : 横向和纵向都进行重复，直到铺满整个画布。ImageRepeat.repeatX: 横向重复，纵向不重复。 ImageRepeat.repeatY：纵向重复，横向不重复。
width		宽度 一般结合ClipOval才能看到效果
height		高度 一般结合ClipOval才能看到效果

更多属性参考：https://api.flutter.dev/flutter/widgets/Image-class.html

10.2、加载远程图片

```
import 'package:flutter/material.dart';

void main(){
  runApp(
    MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text("你好Flutter")),
        body: const MyApp()
      ),
    )
  );
}
/*
图片地址：

https://www.itying.com/images/201906/goods_img/1120_P_1560842352183.png

https://www.itying.com/themes/itying/images/ionic4.png


*/
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
          child:Container(
            width: 150,
            height: 150,
            decoration: const BoxDecoration(
              color: Colors.yellow
            ),
            child: Image.network(
              "https://www.itying.com/themes/itying/images/ionic4.png",
              fit: BoxFit.cover,
            ),
          ) ,
     );
  }
}
```

10.3、Container实现圆形图片

```
import 'package:flutter/material.dart';

void main() {
  runApp(MaterialApp(
    home: Scaffold(
        appBar: AppBar(title: const Text("你好Flutter")), body: const MyApp()),
  ));
}


class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Container(
        width: 150,
        height: 150,
        decoration: BoxDecoration(
            color: Colors.yellow,
            borderRadius: BorderRadius.circular(75),
            image: const DecorationImage(
                image: NetworkImage(
                  "https://www.itying.com/themes/itying/images/ionic4.png",
                ),
                fit: BoxFit.cover)),
      ),
    );
  }
}
```

10.4、ClipOval实现圆形图片

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: ClipOval(
        child: Image.network(
            "https://www.itying.com/themes/itying/images/ionic4.png",
            width: 150.0,
            height: 150.0,
            fit: BoxFit.cover),
      ),
    );
  }
}
```

10.5、CircleAvatar实现圆形图片

radius 元的半径

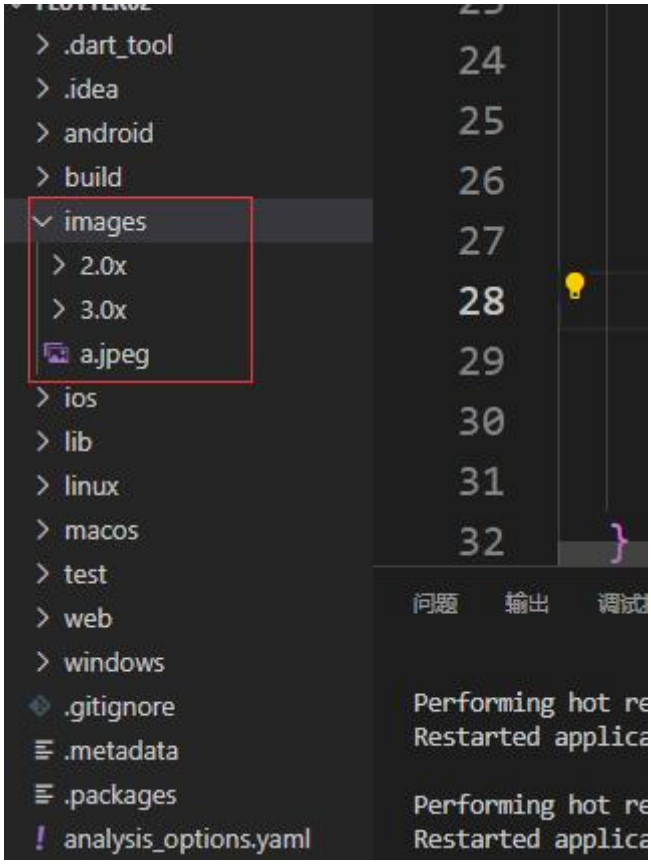
```
const CircleAvatar(
      radius: 200,
      backgroundImage:
NetworkImage("https://www.itying.com/images/flutter/3.png"),
)
```

基本上，CircleAvatar 不提供设置边框的属性。但是，可以将其包裹在具有更大半径和不同背景颜色的不同 CircleAvatar 中，以创建类似于边框的内容。

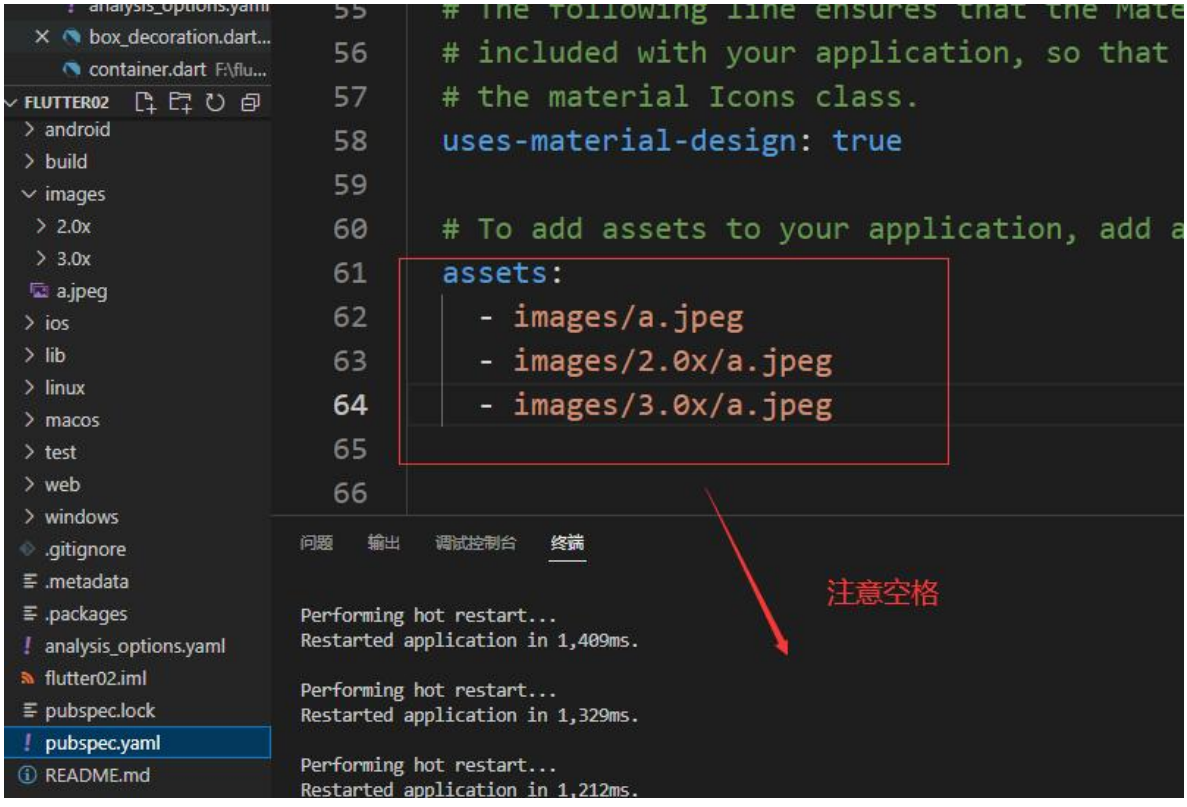
```
return const CircleAvatar(
        radius: 110,
        backgroundColor: Color(0xffFDCF09),
        child: CircleAvatar(
          radius: 100,
          backgroundImage:
NetworkImage("https://www.itying.com/images/flutter/3.png"),
        )
    )
```

10.6、加载本地图片

1、项目根目录新建images文件夹,images中新建2.x 3.x对应的文件



2、然后，打开pubspec.yaml 声明一下添加的图片文件， 注意: 空格



3、使用

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: ClipOval(
        child: Image.asset(
            "images/a.jpeg",
            width: 150.0,
            height: 150.0,
            fit: BoxFit.cover),
      ),
    );
  }
}
```

十一、图标组件

11.1 使用Flutter官方Icons图标

Material Design所有图标可以在其官网查看：https://material.io/tools/icons/

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
          children: const [
            Icon(Icons.search,color: Colors.red,size: 40),
            SizedBox(height: 10),
            Icon(Icons.home),
            SizedBox(height: 10),
            Icon(Icons.category),
            SizedBox(height: 10),
            Icon(Icons.shop),
            SizedBox(height: 10),
          ],
      )
    );
  }
}
```

11.2 Flutter中借助阿里巴巴图标库自定义字体图标

我们也可以使用自定义字体图标。阿里巴巴图标库官网 iconfont.cn上有很多字体图标素材，我们可以选择自己需要的图标打包下载后，会生成一些不同格式的字体文件，在Flutter中，我们使用ttf格式即可。

假设我们项目中需要使用一个书籍图标和微信图标，我们打包下载后导入：

1. 导入字体图标文件；这一步和导入字体文件相同，假设我们的字体图标文件保存在项目根目录下，路径为"fonts/iconfont.ttf"：

```
fonts:
  - family: myIcon  #指定一个字体名
    fonts:
      - asset: fonts/iconfont.ttf
```

也可以配置多个字体文件：

```
  fonts:
    - family: myIcon  #指定一个字体名
      fonts:
        - asset: fonts/iconfont.ttf
    - family: alipayIcon  #指定一个字体名
      fonts:
        - asset: fonts/iconfont2.ttf
```

2、 为了使用方便，我们定义一个`MyIcons`类，功能和`Icons`类一样：将字体文件中的所有图标都定义成静态变量：

```
class MyIcons{
  // book 图标
  static const IconData book = IconData(
      0xe614,
      fontFamily: 'myIcon',
      matchTextDirection: true
  );
  // 微信图标
  static const IconData wechat = IconData(
      0xec7d,
      fontFamily: 'myIcon',
      matchTextDirection: true
  );
}
```

3、 使用

```
Row(
  mainAxisAlignment: MainAxisAlignment.center,
  children: <Widget>[
    Icon(MyIcons.book,color: Colors.purple),
    Icon(MyIcons.wechat,color: Colors.green),
  ],
)
```

十二、Flutter 列表组件

列表布局是我们项目开发中最常用的一种布局方式。Flutter中我们可以通过ListView来定义列表项，支持垂直和水平方向展示。通过一个属性就可以控制列表的显示方向。列表有以下分类：

1、垂直列表
2、垂直图文列表
3、水平列表
4、动态列表

列表组件常用参数：

名称	类型	说明
scrollDirection	Axis	Axis.horizontal水平列表Axis.vertical垂直列表
padding	EdgeInsetsGeometry	内边距
resolve	bool	组件反向排序
children	List	列表元素

12.1、垂直列表

示例1

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return ListView(
      children: const <Widget>[
        ListTile(
          title: Text("我是一个标题"),
        ),
```

```
        ListTile(
          title: Text("我是一个标题"),
        ),
        ListTile(
          title: Text("我是一个标题"),
        ),
        ListTile(
          title: Text("我是一个标题"),
        ),
        ListTile(
          title: Text("我是一个标题"),
        ),
        ListTile(
          title: Text("我是一个标题"),
        ),
        ListTile(
          title: Text("我是一个标题"),
        ),
        ListTile(
          title: Text("我是一个标题"),
        ),
      ],
    );
  }
}
```

示例2


```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return ListView(
      children: const <Widget>[
        ListTile(
          leading: Icon(Icons.assignment, color: Colors.red),
          title: Text("全部订单"),
        ),
        Divider(),
        ListTile(
```

```
          leading: Icon(Icons.payment, color: Colors.green),
          title: Text("待付款"),
        ),
        Divider(),
        ListTile(
          leading: Icon(Icons.local_car_wash, color: Colors.orange),
          title: Text("待收货"),
        ),
        ListTile(
          leading: Icon(Icons.favorite, color: Colors.lightGreen),
          title: Text("我的收藏"),
        ),
        Divider(),
        ListTile(
          leading: Icon(Icons.people, color: Colors.black54),
          title: Text("在线客服"),
        ),
        Divider(),
      ],
    );
  }
}
```

12.2、垂直图文列表

示例1：


```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return ListView(
      children: <Widget>[
        ListTile(
          leading: Image.network("https://www.itying.com/images/flutter/1.png"),
          title: const Text('华北黄淮高温雨今起强势登场'),
          subtitle: const Text("中国天气网讯 21日开始，华北黄淮高温雨今起强势登场"),
        ),
        const Divider(),
        ListTile(
          leading: Image.network("https://www.itying.com/images/flutter/2.png"),
          title: const Text('保监局50天开32罚单 “断供”违规资金为房市降温'),
          subtitle: const Text("中国天气网讯 保监局50天开32罚单 “断供”违规资金为房市降温"),
        ),
        const Divider(),
        ListTile(
            title: const Text('华北黄淮高温雨今起强势登场'),
```

```
            subtitle: const Text("中国天气网讯 21日开始，华北黄淮高温雨今起强势登场"),
            trailing:
                Image.network("https://www.itying.com/images/flutter/3.png")),
        const Divider(),
        ListTile(
          leading: Image.network("https://www.itying.com/images/flutter/4.png"),
          title: const Text('普京现身俄海军节阅兵：乘艇检阅军舰'),
        ),
        const Divider(),
        ListTile(
          leading: Image.network("https://www.itying.com/images/flutter/5.png"),
          title: const Text('鸿星尔克捐1个亿帮助困难残疾群体 网友：企业有担当'),
        ),
        const Divider(),
        ListTile(
          leading: Image.network("https://www.itying.com/images/flutter/6.png"),
          title: const Text('行业冥灯？老罗最好祈祷苹果的AR能成'),
        ),


      ],
    );
  }
}
```

示例2：

我是一个标题

我是一个标题

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return ListView(
      padding: const EdgeInsets.all(10),
      children: <Widget>[
          Image.network("https://www.itying.com/images/flutter/1.png"),
          Container(
            height: 44,
            padding:const EdgeInsets.fromLTRB(0, 10, 0, 10),
            child: const Text(
                '我是一个标题',
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 18,
                ),
            ),
          ),

          Image.network("https://www.itying.com/images/flutter/2.png"),
          Container(
            height: 44,
            padding:const EdgeInsets.fromLTRB(0, 10, 0, 10),
            child: const Text(
```

```
          '我是一个标题',
          textAlign: TextAlign.center,
          style: TextStyle(
            fontSize: 18,
          ),
        ),
      ),

      Image.network("https://www.itying.com/images/flutter/3.png"),
      Container(
        height: 44,
        padding:const EdgeInsets.fromLTRB(0, 10, 0, 10),
        child: const Text(
          '我是一个标题',
          textAlign: TextAlign.center,
          style: TextStyle(
            fontSize: 18,
          ),
        ),
      ),

      Image.network("https://www.itying.com/images/flutter/4.png"),
      Container(
        height: 44,
        padding:const EdgeInsets.fromLTRB(0, 10, 0, 10),
        child: const Text(
          '我是一个标题',
          textAlign: TextAlign.center,
          style: TextStyle(
            fontSize: 18,
          ),
        ),

      ),

      Image.network("https://www.itying.com/images/flutter/1.png"),
      Container(
        height: 44,
        padding:EdgeInsets.fromLTRB(0, 10, 0, 10),
        child: const Text(
          '我是一个标题',
          textAlign: TextAlign.center,
          style: TextStyle(
            fontSize: 18,
          ),
        ),

      ),

      Image.network("https://www.itying.com/images/flutter/2.png"),
      Container(
         height: 44,
        padding:EdgeInsets.fromLTRB(0, 10, 0, 10),
        child: const Text(
          '我是一个标题',
          textAlign: TextAlign.center,
          style: TextStyle(
            fontSize: 18,
```

```
                ),
            ),

          ),

          Image.network("https://www.itying.com/images/flutter/3.png"),
          Container(
             height: 44,
            padding:const EdgeInsets.fromLTRB(0, 10, 0, 10),
            child: const Text(
                '我是一个标题',
                textAlign: TextAlign.center,
                style: TextStyle(
                  fontSize: 18,
                ),
            ),

          ),

          Image.network("https://www.itying.com/images/flutter/1.png"),
          Image.network("https://www.itying.com/images/flutter/2.png"),
          Image.network("https://www.itying.com/images/flutter/3.png"),

      ],
    );
  }
}
```

12.3、水平列表 可以左右滑动


```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 180,
      child: ListView(
        scrollDirection: Axis.horizontal,
        children: <Widget>[
          Container(
```

```
            width: 180.0,
            color: Colors.red,
          ),
          Container(
            width: 180.0,
            color: Colors.orange,
            child: Column(
              children: <Widget>[
                Image.network("https://www.itying.com/images/flutter/1.png"),
                const Text('我是一个文本')
              ],
            ),
          ),
          Container(
            width: 180.0,
            color: Colors.blue,
          ),
          Container(
            width: 180.0,
            color: Colors.deepOrange,
          ),
          Container(
            width: 180.0,
            color: Colors.deepPurpleAccent,
          ),
        ],
      ),
    );
  }
}
```

12.4、ListView动态列表组件 以及循环动态数据

1、for循环实现动态列表

```
import 'package:flutter/material.dart';
import './ityingFont.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primarySwatch: Colors.yellow,
      ),
```

```
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter ICON")),
        body: const MyHomePage(),
      ),
    );
  }
}

class MyHomePage extends StatelessWidget {
  const MyHomePage({Key? key}) : super(key: key);
  List<Widget> _initListView(){
    List<Widget> list=[];
    for (var i = 0; i < 10; i++) {
      list.add(
        const ListTile(
           title: Text("我是一个列表"),
        )
      );
    }
    return list;
  }

  @override
  Widget build(BuildContext context) {
    return ListView(
      children: _initListView(),
    );
  }
}
```

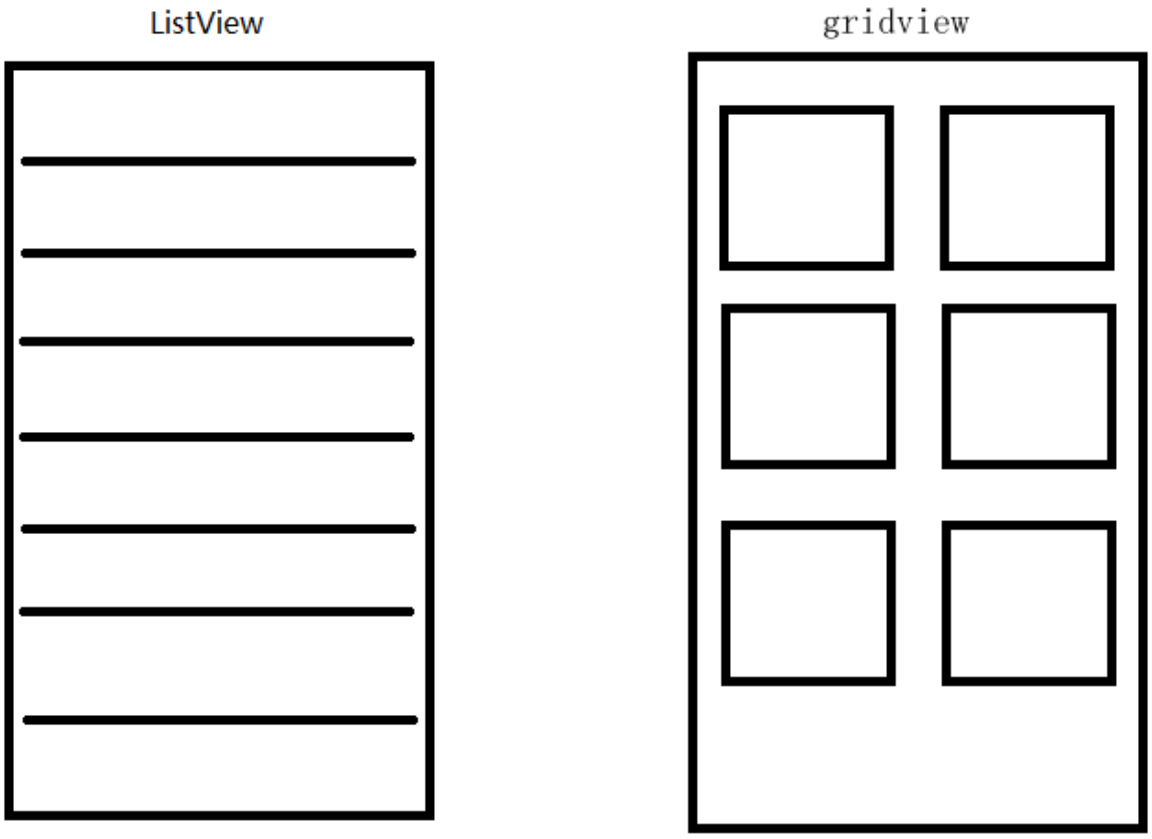
2、ListView.builder实现动态列表

```
import 'package:flutter/material.dart';
import './ityingFont.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      theme: ThemeData(
        primarySwatch: Colors.yellow,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter ICON")),
        body: MyHomePage(),
      ),
    );
  }
```

```
}

class MyHomePage extends StatelessWidget {
  List list = [];
  MyHomePage({Key? key}) : super(key: key) {
    for (var i = 0; i < 10; i++) {
      list.add("我是一个列表--$i");
    }
  }
  @override
  Widget build(BuildContext context) {
    return ListView.builder(
        itemCount: list.length,
        itemBuilder: (context, index) {
          return ListTile(
            title: Text("${list[index]}"),
          );
        });
  }
}
```

十三、Flutter GridView网格布局组件

13.1、GridView网格布局组件介绍



GridView网格布局在实际项目中用的也是非常多的，当我们想让可以滚动的元素使用矩阵方式排列的时候。此时我们可以用网格列表组件GridView实现布局。

GridView创建网格列表主要有下面三种方式

1、可以通过GridView.count 实现网格布局

2、可以通过GridView.extent 实现网格布局

3、通过GridView.builder实现动态网格布局

常用属性：

名称	类型	说明
scrollDirection	Axis	滚动方法
padding	EdgeInsetsGeometry	内边距
resolve	bool	组件反向排序
crossAxisSpacing	double	水平子Widget之间间距
mainAxisSpacing	double	垂直子Widget之间间距
crossAxisCount	int 用在GridView.count	一行的Widget数量
maxCrossAxisExtent	double 用在GridView.extent	横轴子元素的最大长度
childAspectRatio	double	子Widget宽高比例
children		[ ]
gridDelegate	SliverGridDelegateWithFixedCrossAxisCount SliverGridDelegateWithMaxCrossAxisExtent	控制布局主要用在GridView.builder里面

13.2、GridView.count 实现网格布局

GridView.count构造函数内部使用了SliverGridDelegateWithFixedCrossAxisCount，我们通过它可以快速的创建横轴固定数量子元素的GridView

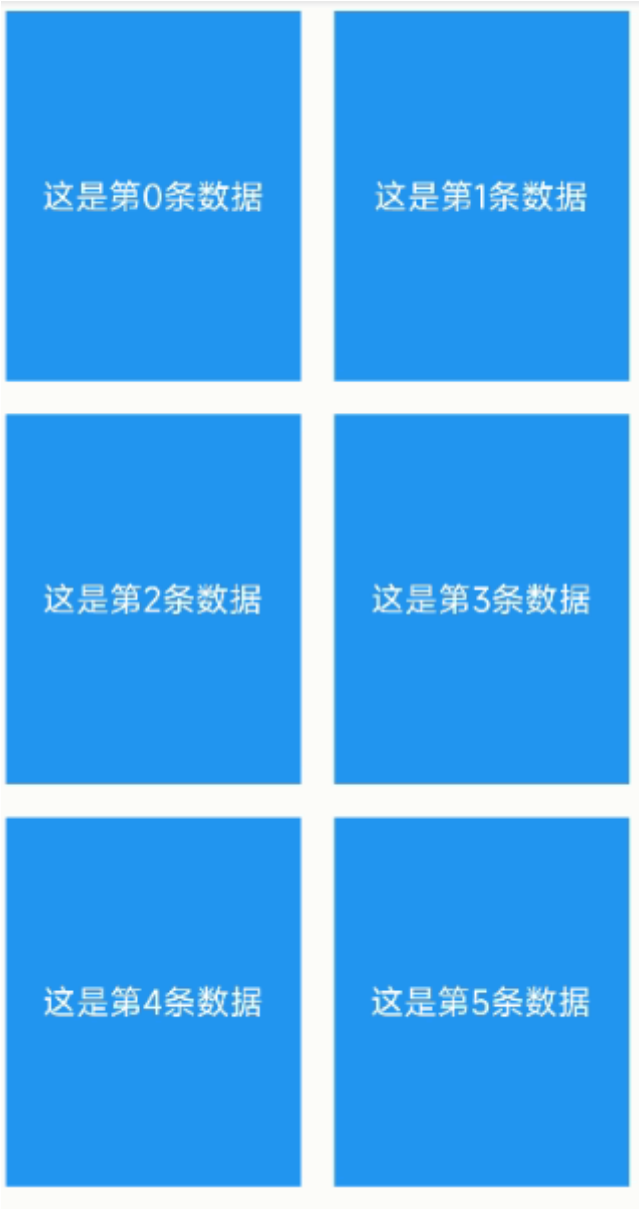
```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GridView.count(
      crossAxisCount: 3,
      childAspectRatio: 1.0,
      children: const <Widget>[
        Icon(Icons.home),
        Icon(Icons.ac_unit),
        Icon(Icons.search),
        Icon(Icons.settings),
        Icon(Icons.airport_shuttle),
        Icon(Icons.all_inclusive),
        Icon(Icons.beach_access),
        Icon(Icons.cake),
        Icon(Icons.circle),
      ],
    );
  }
}
```

13.3、GridView.extent实现网格布局

GridView.extent构造函数内部使用了SliverGridDelegateWithMaxCrossAxisExtent，我们通过它可以快速的创建横轴子元素为固定最大长度的的GridView。

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return GridView.extent(
      maxCrossAxisExtent: 80.0,
      childAspectRatio: 1.0,
      children: const <Widget>[
        Icon(Icons.home),
        Icon(Icons.ac_unit),
        Icon(Icons.search),
        Icon(Icons.settings),
        Icon(Icons.airport_shuttle),
        Icon(Icons.all_inclusive),
        Icon(Icons.beach_access),
        Icon(Icons.cake),
        Icon(Icons.circle),
      ],
    );
  }
}
```

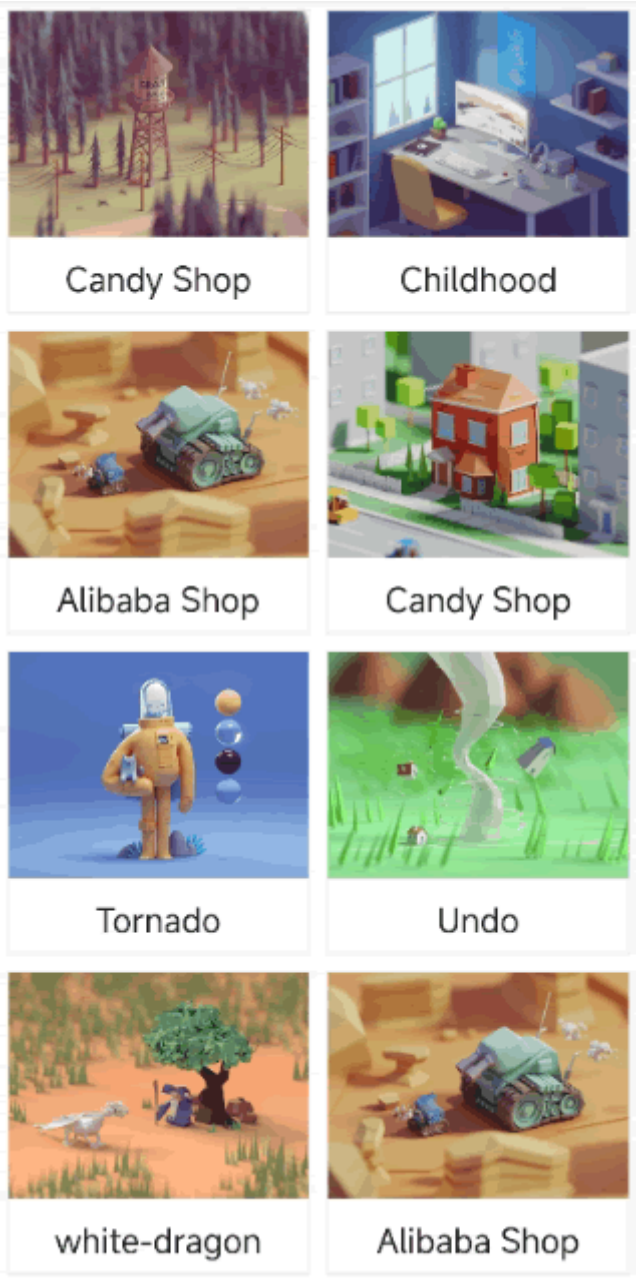
13.4、GridView.count 和 GridView.extent属性详解




```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  List<Widget> _getListData() {
    List<Widget> list = [];
    for (var i = 0; i < 20; i++) {
      list.add(Container(
        alignment: Alignment.center,
        color: Colors.blue,
        child: Text(
          '这是第$i条数据',
          style:  const TextStyle(color: Colors.white, fontSize: 20),
        ),
        // height: 400,  //设置高度没有反应
      ));
    }
    return list;
  }

  @override
  Widget build(BuildContext context) {
    return GridView.count(
      crossAxisSpacing: 20.0, //水平子 Widget 之间间距
      mainAxisSpacing: 20.0, //垂直子 Widget 之间间距
```

```
      padding: const EdgeInsets.all(10),
      crossAxisCount: 2, //一行的 Widget 数量
      childAspectRatio: 0.8, //宽度和高度的比例
      children: _getListData(),
    );
  }
}
```

13.5、GridView.count 实现动态列表




```
import 'package:flutter/material.dart';
import 'res/listData.dart';
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
```

```
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("FLutter App")),
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  List<Widget> _getListData() {
      var tempList=listData.map((value){
          return Container(
            decoration: BoxDecoration(
              border: Border.all(
                color:const Color.fromRGBO(233, 233,233, 0.9),
                width: 1
              )
            ),
            child:Column(
                children: <Widget>[
                  Image.network(value['imageUrl']),
                  const SizedBox(height: 12),
                  Text(
                    value['title'],
                    textAlign: TextAlign.center,
                    style:   const TextStyle(
                      fontSize: 20
                    ),
                  )
                ],
            ),

          );

      });
      // ('xxx','xxx')
      return tempList.toList();
  }


  @override
  Widget build(BuildContext context) {
    return GridView.count(
        crossAxisSpacing:10.0 ,   //水平子 Widget 之间间距
        mainAxisSpacing: 10.0,    //垂直子 Widget 之间间距
        padding: const EdgeInsets.all(10),
```

```
        crossAxisCount: 2,  //一行的 Widget 数量
        // childAspectRatio:0.7,  //宽度和高度的比例
        children: _getListData(),
    );
  }
}
```

13.6、GridView.builder实现动态列表

SliverGridDelegateWithFixedCrossAxisCount

```
import 'package:flutter/material.dart';
import 'res/listData.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("FLutter App")),
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  Widget _getListData(context, index) {
    return Container(
      decoration: BoxDecoration(
          border: Border.all(
              color: const Color.fromRGBO(233, 233, 233, 0.9), width: 1)),
      child: Column(
        children: <Widget>[
          Image.network(listData[index]['imageUrl']),
          const SizedBox(height: 12),
          Text(
            listData[index]['title'],
```

```
            textAlign: TextAlign.center,
            style: const TextStyle(fontSize: 20),
          )
        ],
      ),
      // height: 400,  //设置高度没有反应
    );
  }

  @override
  Widget build(BuildContext context) {
    return GridView.builder(
      //注意
      gridDelegate: const SliverGridDelegateWithFixedCrossAxisCount(
        crossAxisSpacing: 10.0, //水平子 Widget 之间间距
        mainAxisSpacing: 10.0, //垂直子 Widget 之间间距
        crossAxisCount: 2, //一行的 Widget 数量
      ),
      itemCount: listData.length,
      itemBuilder: _getListData,
    );
  }
}
```

SliverGridDelegateWithMaxCrossAxisExtent

```
import 'package:flutter/material.dart';
import 'res/listData.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("FLutter App")),
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatelessWidget {
```

```
  const HomePage({Key? key}) : super(key: key);

  Widget _getListData(context, index) {
    return Container(
      decoration: BoxDecoration(
          border: Border.all(
              color: const Color.fromRGBO(233, 233, 233, 0.9), width: 1)),
      child: Column(
        children: <Widget>[
          Image.network(listData[index]['imageUrl']),
          const SizedBox(height: 12),
          Text(
            listData[index]['title'],
            textAlign: TextAlign.center,
            style: const TextStyle(fontSize: 20),
          )
        ],
      ),
      // height: 400,  //设置高度没有反应
    );
  }

  @override
  Widget build(BuildContext context) {
    return GridView.builder(
      //注意
      gridDelegate: const SliverGridDelegateWithMaxCrossAxisExtent  (
        crossAxisSpacing:10,
        mainAxisSpacing:10,
        maxCrossAxisExtent: 300,
      ),
      itemCount: listData.length,
      itemBuilder: _getListData,
    );
  }
}
```

十四、Flutter Paddiing组件

在html中常见的布局标签都有padding属性，但是Flutter中很多Widget是没有padding属性。这个时候我们可以用Padding组件处理容器与子元素之间的间距。

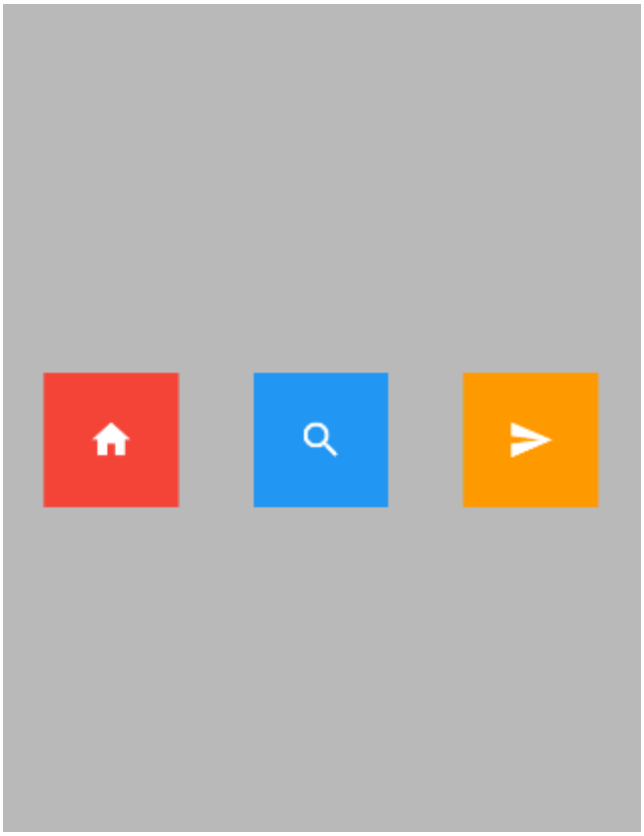
属性	说明
padding	padding值, EdgeInsetss设置填充的值
child	子组件

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return GridView.count(
      // padding: const EdgeInsets.all(10),
      crossAxisCount: 2,
      childAspectRatio: 1,
      children: [
        Padding(
          padding: const EdgeInsets.all(10),
          child: Image.network('https://www.itying.com/images/flutter/1.png',
              fit: BoxFit.cover),
        ),
         Padding(
          padding: const EdgeInsets.all(10),
          child: Image.network('https://www.itying.com/images/flutter/2.png',
              fit: BoxFit.cover),
        ),
         Padding(
          padding: const EdgeInsets.all(10),
          child: Image.network('https://www.itying.com/images/flutter/3.png',
              fit: BoxFit.cover),
        ),
         Padding(
          padding: const EdgeInsets.all(10),
          child: Image.network('https://www.itying.com/images/flutter/4.png',
              fit: BoxFit.cover),
        ),
         Padding(
          padding: const EdgeInsets.all(10),
          child: Image.network('https://www.itying.com/images/flutter/5.png',
              fit: BoxFit.cover),
        ),
         Padding(
          padding: const EdgeInsets.all(10),
          child: Image.network('https://www.itying.com/images/flutter/6.png',
              fit: BoxFit.cover),
        ),

      ],
    );
  }
}
```

十四、线性布局（Row和Column）

14.1、Row 水平布局组件

属性	说明
mainAxisAlignment	主轴的排序方式
crossAxisAlignment	次轴的排序方式
children	组件子元素

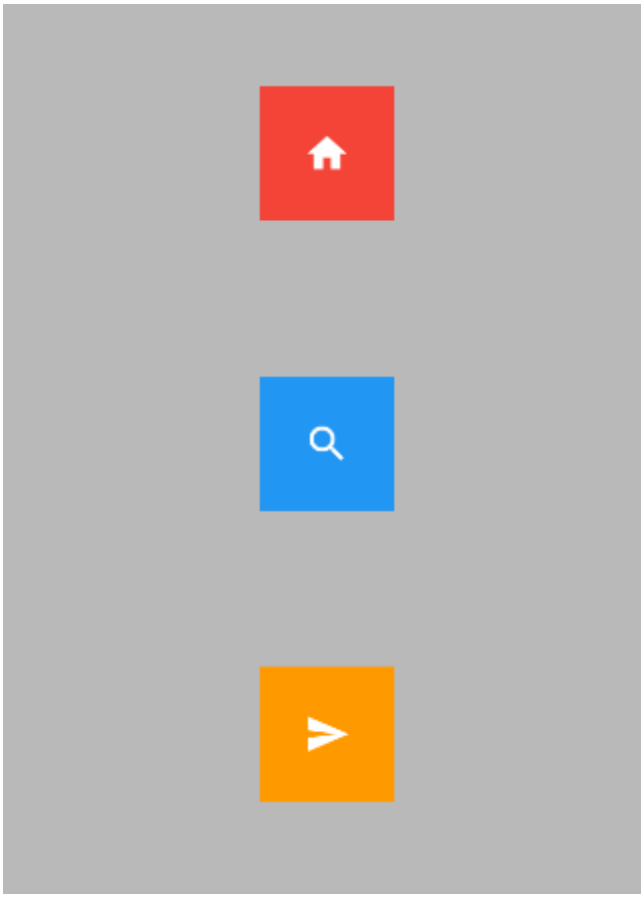


```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      height: double.infinity,
      width: double.infinity,
      color: Colors.black26,
      child: Row(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          IconContainer(Icons.home, color: Colors.red),
          IconContainer(Icons.search, color: Colors.blue),
          IconContainer(Icons.send, color: Colors.orange),
        ],
      ),
    );
  }
}
```

```
class IconContainer extends StatelessWidget {
  Color color;
  double size;
  IconData icon;
  IconContainer(this.icon,
      {Key? key, this.color = Colors.red, this.size = 32.0})
      : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 100.0,
      width: 100.0,
      color: color,
      child: Center(child: Icon(icon, size: size, color: Colors.white)),
    );
  }
}
```

14.2、Column垂直布局组件

属性	说明
mainAxisAlignment	主轴的排序方式
crossAxisAlignment	次轴的排序方式
children	组件子元素



```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Container(
      height: double.infinity,
      width: double.infinity,
      color: Colors.black26,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          IconContainer(Icons.home, color: Colors.red),
          IconContainer(Icons.search, color: Colors.blue),
          IconContainer(Icons.send, color: Colors.orange),
        ],
      ),
    );
  }
}

class IconContainer extends StatelessWidget {
  Color color;
  double size;
  IconData icon;
  IconContainer(this.icon,
      {Key? key, this.color = Colors.red, this.size = 32.0})
```

```
      : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 100.0,
      width: 100.0,
      color: color,
      child: Center(child: Icon(icon, size: size, color: Colors.white)),
    );
  }
}
```

14.3、double.infinity 和double.maxFinite

double.infinity 和double.maxFinite可以让当前元素的width或者height达到父元素的尺寸

底层代码

```
  static const double nan = 0.0 / 0.0;
  static const double infinity = 1.0 / 0.0;
  static const double negativeInfinity = -infinity;
  static const double minPositive = 5e-324;
  static const double maxFinite = 1.7976931348623157e+308;
```

如下可以让Container铺满整个屏幕

```
 Widget build(BuildContext context) {
    return Container(
      height: double.infinity,
      width: double.infinity,
      color: Colors.black26,
      child: Column(
        crossAxisAlignment: CrossAxisAlignment.center,
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          IconContainer(Icons.home, color: Colors.red),
          IconContainer(Icons.search, color: Colors.blue),
          IconContainer(Icons.send, color: Colors.orange),
        ],
      ),
    );
  }
```

如下可以让Container的宽度和高度等于父元素的宽度高度

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
```

```
  Widget build(BuildContext context) {
    return Container(
      height: 400,
      width: 600,
      color: Colors.red,
      child: Container(
        height: double.maxFinite,
        width: double.infinity,
        color: Colors.black26,
        child: Column(
          crossAxisAlignment: CrossAxisAlignment.center,
          mainAxisAlignment: MainAxisAlignment.spaceEvenly,
          children: [
            IconContainer(Icons.home, color: Colors.red),
            IconContainer(Icons.search, color: Colors.blue),
            IconContainer(Icons.send, color: Colors.orange),
          ],
        ),
      ),
    );
  }
}
```

十五、 弹性布局（Flex Expanded）

`Flex`组件可以沿着水平或垂直方向排列子组件，如果你知道主轴方向，使用`Row`或`Column`会方便一些，**因为`Row`和`Column`都继承自`Flex`**，参数基本相同，所以能使用Flex的地方基本上都可以使用`Row`或`Column`。`Flex`本身功能是很强大的，它也可以和`Expanded`组件配合实现弹性布局 。

15.1 水平弹性布局

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
```

```
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Flex(
      direction: Axis.horizontal,
      children: [
        Expanded(flex: 2, child: IconContainer(Icons.home, color: Colors.red)),
        Expanded(
          flex: 1,
          child: IconContainer(Icons.search, color: Colors.orange),
        )
      ],
    );
  }
}

class IconContainer extends StatelessWidget {
  Color color;
  double size;
  IconData icon;
  IconContainer(this.icon,
      {Key? key, this.color = Colors.red, this.size = 32.0})
      : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 100.0,
      width: 100.0,
      color: color,
      child: Center(child: Icon(icon, size: size, color: Colors.white)),
    );
  }
}
```

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
```

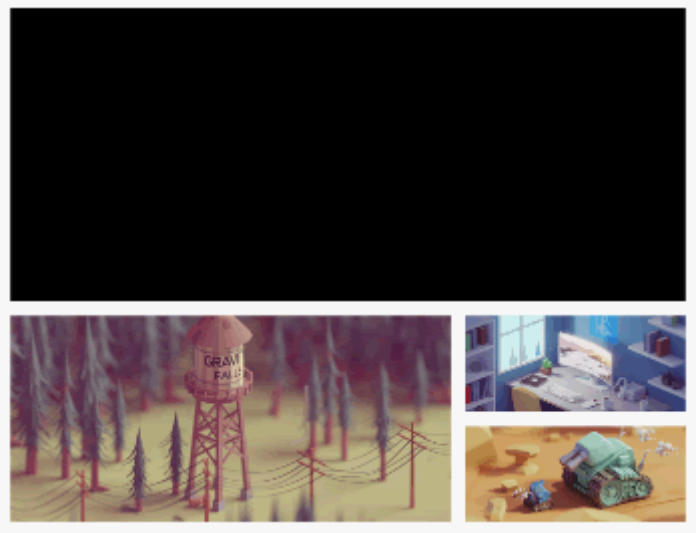
```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Row(
      children: [
        Expanded(flex: 2, child: IconContainer(Icons.home, color: Colors.red)),
        Expanded(
          flex: 1,
          child: IconContainer(Icons.search, color: Colors.orange),
        )
      ],
    );
  }
}

class IconContainer extends StatelessWidget {
  Color color;
  double size;
  IconData icon;
  IconContainer(this.icon,
      {Key? key, this.color = Colors.red, this.size = 32.0})
      : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 100.0,
      width: 100.0,
      color: color,
      child: Center(child: Icon(icon, size: size, color: Colors.white)),
    );
  }
}
```

15.1 垂直弹性布局

```
import 'package:flutter/material.dart';

void main() {
```

```
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Column(
      children: [
        Expanded(flex: 2, child: IconContainer(Icons.home, color: Colors.red)),
        Expanded(
          flex: 1,
          child: IconContainer(Icons.search, color: Colors.orange),
        )
      ],
    );
  }
}

class IconContainer extends StatelessWidget {
  Color color;
  double size;
  IconData icon;
  IconContainer(this.icon,
      {Key? key, this.color = Colors.red, this.size = 32.0})
      : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 100.0,
      width: 100.0,
      color: color,
      child: Center(child: Icon(icon, size: size, color: Colors.white)),
    );
  }
}
```

15.2、使用Row或Column结合Expanded实现下面示例



```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return ListView(
      children: [
        Container(
```

```
          width: double.infinity,
          height: 200,
          color: Colors.black,
        ),
        const SizedBox(height: 10),
        Row(
          children: [
            Expanded(
              flex: 2,
              child: SizedBox(
                height: 180,
                child: Image.network(
                    "https://www.itying.com/images/flutter/2.png",
                    fit: BoxFit.cover),
              ),
            ),
            const SizedBox(width: 10),
            Expanded(
                flex: 1,
                child: SizedBox(
                   height: 180,
                  child: Column(
                    children: [
                      Expanded(
                        flex: 1,
                        child: SizedBox(
                          width:double.infinity ,
                          child: Image.network(
                            "https://www.itying.com/images/flutter/3.png",
                            fit: BoxFit.cover),
                        ),
                      ),
                      const SizedBox(height: 10),
                      Expanded(
                        flex: 2,
                        child: SizedBox(
                          width:double.infinity ,
                          child: Image.network(
                            "https://www.itying.com/images/flutter/4.png",
                            fit: BoxFit.cover),
                        ),
                      )
                    ],
                  ),
                ))
          ],
        )
      ],
    );
  }
}
```

十六、层叠布局（Stack、Align、Positioned）

16.1、Flutter Stack组件

Stack表示堆的意思，我们可以用Stack或者Stack结合Align或者Stack结合 Positiond来实现页面的定位布局

属性	说明
alignment	配置所有子元素的显示位置
children	子组件

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Stack(
        alignment: Alignment.topLeft,
        children: <Widget>[
          Container(
            height: 400,
            width: 300,
            color: Colors.red,
          ),
          const Text('我是一个文本',
              style: TextStyle(fontSize: 40, color: Colors.white))
        ],
      ),
    );
  }
}
```

16.2、Flutter Stack Align

`Align` 组件可以调整子组件的位置，Stack组件中结合Align组件也可以控制每个子元素的显示位置

属性	说明
alignment	配置所有子元素的显示位置
child	子组件

2.1 Align结合Container的使用

我们先来看一个简单的例子：

`FlutterLogo` 是Flutter SDK 提供的一个组件，内容就是 Flutter 的 log

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Container(
      height: 120.0,
      width: 120.0,
      color: Colors.blue.shade50,
      child: const Align(
        alignment: Alignment.topRight,
        child: FlutterLogo(
          size: 60,
        ),
      ),
    );
  }
}
```

2.2 Align结合Alignment 参数

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Container(
        height: 120.0,
        width: 120.0,
        color: Colors.blue.shade50,
        child: const Align(
          alignment: Alignment(2, 0.0),
          child: FlutterLogo(
            size: 60,
          ),
        ));
  }
}
```

`Alignment` Widget会以**矩形的中心点作为坐标原点**，即`Alignment(0.0, 0.0)`。`x`、`y`的值从-1到1分别代表矩形左边到右边的距离和顶部到底边的距离，因此2个水平（或垂直）单位则等于矩形的宽（或高），如`Alignment(-1.0, -1.0)`代表矩形的左侧顶点，而`Alignment(1.0, 1.0)`代表右侧底部终点，而`Alignment(1.0, -1.0)`则正是右侧顶点，即`Alignment.topRight`。为了使用方便，矩形的原点、四个顶点，以及四条边的终点在`Alignment`类中都已经定义为了静态常量。

`Alignment`可以通过其**坐标转换公式**将其坐标转为子元素的具体偏移坐标：

```
(Alignment.x*childWidth/2+childWidth/2, Alignment.y*childHeight/2+childHeight/2)
```

其中`childWidth`为子元素的宽度，`childHeight`为子元素高度。

现在我们再看看上面的示例，我们将`Alignment(2, 0.0)`带入上面公式，`(2*120/2+120/2, 0*120/2+120/2)`，可得`FlutterLogo`的实际偏移坐标正是（180,60）。下面再看一个例子：

2.3 Align结合Alignment 参数

`Center`继承自`Align`，它比`Align`只少了一个`alignment`参数；由于`Align`的构造函数中`alignment`值为`Alignment.center`，所以，我们可以认为`Center`组件其实是对齐方式确定

2.4 Align结合Stack组件

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Center(
      child:  Container(
            height: 400,
            width: 300,
            color: Colors.red,
            child: Stack(
              // alignment: Alignment.center,
              children: const <Widget>[
                Align(
                  alignment: Alignment(1,-0.2),
                  child: Icon(Icons.home,size: 40,color: Colors.white),
                ),
```

```
                Align(
                  alignment: Alignment.center,
                  child: Icon(Icons.search,size: 30,color: Colors.white),
                ),
                Align(
                  alignment: Alignment.bottomRight,
                  child: Icon(Icons.settings_applications,size: 30,color:
Colors.white),
                )
              ],
            ),
      ),
    );
  }
}
```

16.3、Flutter Stack Positioned

Stack组件中结合Positioned组件也可以控制每个子元素的显示位置

属性	说明
top	子元素距离顶部的距离
bottom	子元素距离底部的距离
left	子元素距离左侧距离
right	子元素距离右侧距离
child	子组件
width	组件的高度 （注意：宽度和高度必须是固定值，没法使用double.infinity）
height	子组件的高度

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return  Center(
      child:  Container(
            height: 400,
            width: 300,
            color: Colors.red,
            child: Stack(
              // alignment: Alignment.center,
              children: const <Widget>[
                Positioned(
                 left: 10,
                  child: Icon(Icons.home,size: 40,color: Colors.white),
```

```
                ),
                Positioned(
                 bottom: 0,
                 left: 100,
                  child: Icon(Icons.search,size: 30,color: Colors.white),
                ),
                Positioned(
                  right: 0,
                  child: Icon(Icons.settings_applications,size: 30,color:
Colors.white),
                )
              ],
            ),
      ),
    );
  }
}
```

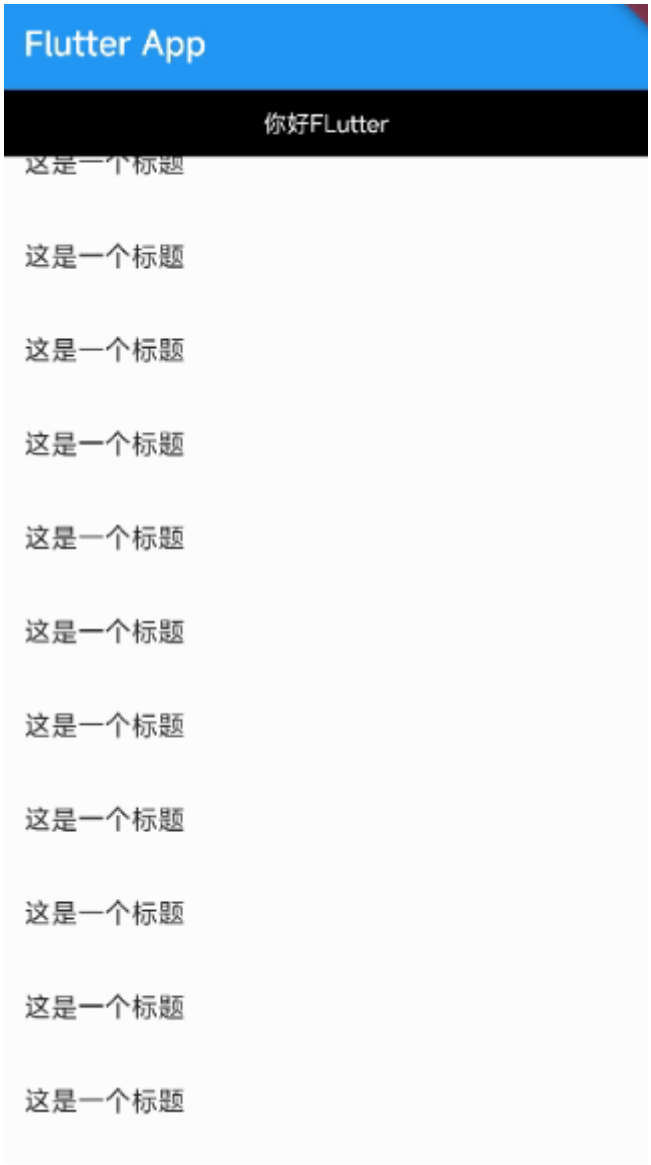
16.4、FlutterMediaQuery获取屏幕宽度和高度

```
final size =MediaQuery.of(context).size;
```

组件的build方法中可以通过，=MediaQuery.of(context).size;

```
Widget build(BuildContext context) {
    final size =MediaQuery.of(context).size;
    final width =size.width;
    final height =size.height;
}
```

16.5、Flutter Stack Positioned固定导航案例




```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    final size = MediaQuery.of(context).size;

    return Stack(
      children: [
        ListView(
          padding: const EdgeInsets.only(top: 45),
          children: const [
            ListTile(
              title: Text("这是一个标题 "),
            ),
            ListTile(
              title: Text("这是一个标题"),
            ),
            ListTile(
              title: Text("这是一个标题"),
            ),
            ListTile(
```

```
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
        ),
        ListTile(
          title: Text("这是一个标题"),
```

```
            ),
          ],
        ),
        Positioned(
            top: 0,
            left: 0,
            height: 40,
            width: size.width,
            child: Container(
              alignment: Alignment.center,
              color: Colors.black,
              child: const Text("你好FLutter",style: TextStyle(color:
Colors.white),),
            ))
      ],
    );
  }
}
```

十七、Flutter AspectRatio

AspectRatio的作用是根据设置调整子元素child的宽高比。

AspectRatio首先会在布局限制条件允许的范围内尽可能的扩展，widget的高度是由宽度和比率决定的，类似于BoxFit中的contain，按照固定比率去尽量占满区域。

如果在满足所有限制条件过后无法找到一个可行的尺寸，AspectRatio最终将会去优先适应布局限制条件，而忽略所设置的比率。

属性	说明
aspectRatio	宽高比，最终可能不会根据这个值去布局，具体则要看综合因素，外层是否允许按照这种比率进行布局，这只是一个参考值
child	子组件

```
class HomePage extends StatelessWidget {
  const HomePage({Key? key}) : super(key: key);
  @override
 Widget build(BuildContext context) {
    // TODO: implement build
    return Container(

        width: 200,
        color: Colors.yellow,
        child: AspectRatio(
          aspectRatio: 2.0/1.0,
          child: Container(
```

```
            color: Colors.red,
          ),
        ),

    );
  }
}
```

```
class LayoutDemo extends StatelessWidget {
 @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return AspectRatio(
        aspectRatio: 3.0/1.0,
        child: Container(
          color: Colors.red,
        ),
    );
  }
}
```

十八、Flutter Card组件

Card是卡片组件块，内容可以由大多数类型的Widget构成，Card具有圆角和阴影，这让它看起来有立体感。

属性	说明
margin	外边距
child	子组件
elevation	阴影值的深度
color	背景颜色
shadowColor	阴影颜色
margin	外边距
clipBehavior	clipBehavior 内容溢出的剪切方式 Clip.none不剪切 Clip.hardEdge裁剪但不应用抗锯齿 Clip.antiAlias裁剪而且抗锯齿 Clip.antiAliasWithSaveLayer带有抗锯齿的剪辑，并在剪辑之后立即保存saveLayer
Shape	Card的阴影效果，默认的阴影效果为圆角的长方形边。 shape: const RoundedRectangleBorder(borderRadius: BorderRadius.all(Radius.circular(10))),

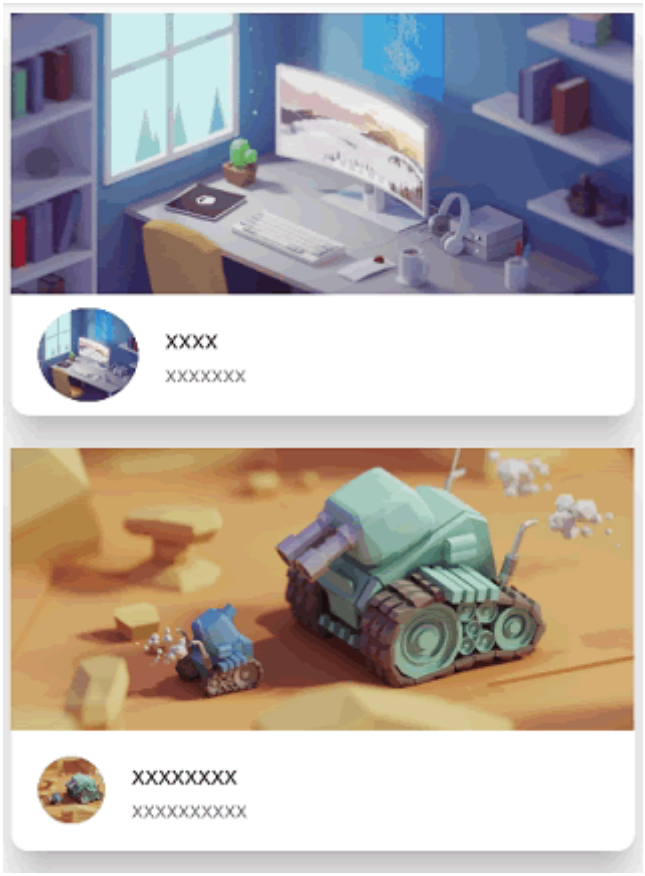
18.1 Card实现一个通讯录的卡片


```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const LayoutDemo(),
      ),
    );
  }
}

class LayoutDemo extends StatelessWidget {
  const LayoutDemo({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return ListView(
      children: <Widget>[
        Card(
          shape: const RoundedRectangleBorder(
```

```
            borderRadius: BorderRadius.all(Radius.circular(10))),
          elevation: 20,
          margin: const EdgeInsets.all(10),
          child: Column(
            children: const <Widget>[
              ListTile(
                title: Text("张三", style: TextStyle(fontSize: 28)),
                subtitle: Text("高级软件工程师"),
              ),
              Divider(),
              ListTile(
                title: Text("电话：1213214142"),
              ),
              ListTile(title: Text("地址：北京市海淀区"))
            ],
          ),
        ),
        Card(
          shape: const RoundedRectangleBorder(
              borderRadius: BorderRadius.all(Radius.circular(10))),
          // color: Colors.red,
          elevation: 20,
          margin: const EdgeInsets.all(10),
          child: Column(
            children: const <Widget>[
              ListTile(
                title: Text("李四", style: TextStyle(fontSize: 28)),
                subtitle: Text("高级软件工程师"),
              ),
              Divider(),
              ListTile(
                title: Text("电话：1213214142"),
              ),
              ListTile(title: Text("地址：北京市海淀区"))
            ],
          ),
        )
      ],
    );
  }
}
```

SSS

18.2 Card实现一个图文列表卡片




```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const LayoutDemo(),
      ),
    );
  }
}

class LayoutDemo extends StatelessWidget {
  const LayoutDemo({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // TODO: implement build
    return ListView(
```

```
      children: <Widget>[
         Card(
          elevation:20,
          shape: RoundedRectangleBorder(
            borderRadius:BorderRadius.circular(10),
          ),
          margin: const EdgeInsets.all(10),
          child:Column(
              children: <Widget>[
                  AspectRatio(
                    aspectRatio: 20/9,
                    child:
Image.network("https://www.itying.com/images/flutter/2.png",fit: BoxFit.cover,),
                  ),
                  ListTile(
                    leading: ClipOval(
                      child:
Image.network("https://www.itying.com/images/flutter/2.png",fit:
BoxFit.cover,height:60,width: 60),
                    ),
                    title:const Text("xxxx"),
                    subtitle:const Text("xxxxxxx"),

                  )

              ],
          ),

        ),
         Card(
          elevation:20,
           shape: RoundedRectangleBorder(
            borderRadius:BorderRadius.circular(10),
          ),
          margin: const EdgeInsets.all(10),
          child:Column(
              children: <Widget>[
                  AspectRatio(
                    aspectRatio: 20/9,
                    child:
Image.network("https://www.itying.com/images/flutter/3.png",fit: BoxFit.cover,),
                  ),
                  const ListTile(
                    leading: CircleAvatar(

backgroundImage:NetworkImage('https://www.itying.com/images/flutter/3.png')
                    ),
                    title: Text("xxxxxxxx"),
                    subtitle: Text("xxxxxxxxxx"),

                  )
              ],
          ),

        )
      ],
    );
  }
```

```
}
```

18.3 CircleAvatar实现一个圆形图片

radius 元的半径

```
const CircleAvatar(
      radius: 200,
      backgroundImage:
NetworkImage("https://www.itying.com/images/flutter/3.png"),
)
```

基本上，CircleAvatar 不提供设置边框的属性。但是，可以将其包裹在具有更大半径和不同背景颜色的不同 CircleAvatar 中，以创建类似于边框的内容。

```
return const CircleAvatar(
        radius: 110,
        backgroundColor: Color(0xffFDCF09),
        child: CircleAvatar(
          radius: 100,
          backgroundImage:
NetworkImage("https://www.itying.com/images/flutter/3.png"),
        )
    )
```

十九、Flutter 按钮组件

19.1 按钮组件的属性

属性	说明
onPressed	必填参数，按下按钮时触发的回调，接收一个方法，传null表示按钮禁用，会显示禁用相关样式
child	子组件
style	通过ButtonStyle装饰

ButtonStylee里面的常用的参数

属性名称	值类型	属性值
foregroundColor	Color	文本颜色
backgroundColor	Color	按钮的颜色
shadowColor	Color	阴影颜色
elevation	double	阴影的范围，值越大阴影范围越大
padding		内边距
shape		设置按钮的形状 shape: MaterialStateProperty.all(RoundedRectangleBorder(borderRadius: BorderRadius.circular(10)))
side	设置边框	MaterialStateProperty.all(BorderSide(width:1,color: Colors.red))

19.2 ElevatedButton

`ElevatedButton` 即"凸起"按钮，它默认带有阴影和灰色背景。按下后，阴影会变大

使用`ElevatedButton`非常简单，如：

```
ElevatedButton(
          onPressed: () {},
          child: const Text("普通按钮")
)
```

19.3 TextButton

`TextButton`即文本按钮，默认背景透明并不带阴影。按下后，会有背景色

```
TextButton(
  child: Text("文本按钮"),
  onPressed: () {},
)
```

19.4 OutlinedButton

`OutlineButton`默认有一个边框，不带阴影且背景透明。按下后，边框颜色会变亮、同时出现背景和阴影


```
OutlinedButton(
  child: Text("边框按钮"),
  onPressed: () {},
)
```

19.5 IconButton

`IconButton`是一个可点击的Icon，不包括文字，默认没有背景，点击后会出现背景

```
IconButton(
  icon: Icon(Icons.thumb_up),
  onPressed: () {},
)
```

19.6 带图标的按钮

`ElevatedButton`、`TextButton`、`OutlineButton`都有一个`icon`构造函数，通过它可以轻松创建带图标的按钮


```
ElevatedButton.icon(
  icon: Icon(Icons.send),
  label: Text("发送"),
  onPressed: _onPressed,
),
OutlineButton.icon(
  icon: Icon(Icons.add),
  label: Text("添加"),
  onPressed: _onPressed,
),
TextButton.icon(
  icon: Icon(Icons.info),
  label: Text("详情"),
  onPressed: _onPressed,
),
```

19.7 修改按钮的宽度高度

```
SizedBox(
                  height: 80,
                  width: 200,
                  child: ElevatedButton(
                    style:ButtonStyle(
                      backgroundColor:MaterialStateProperty.all(Colors.red),
                      foregroundColor: MaterialStateProperty.all(Colors.black)
                    ) ,
                    onPressed: () {

                    },
                    child: const Text('宽度高度'),
                  ),
)
```

19.8 自适应按钮

```
Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Expanded(
                child: Container(
                  height: 60,
                  margin: const EdgeInsets.all(10),
                  child: ElevatedButton(
                    child: const Text('自适应按钮'),
                    onPressed: () {
                      print("自适应按钮");
                    },
                  ),
                ),
              )
            ],
),
```

19.9 配置圆形圆角按钮

圆角按钮

```
ElevatedButton(
                    style: ButtonStyle(
                      backgroundColor:MaterialStateProperty.all(Colors.blue),
                      foregroundColor: MaterialStateProperty.all(Colors.white),

                      elevation: MaterialStateProperty.all(20),
                      shape: MaterialStateProperty.all(
                        RoundedRectangleBorder(
                          borderRadius: BorderRadius.circular(10))
                        ),
                    ),
                     onPressed: () {
                       print("圆角按钮");
                     },
                     child: const Text('圆角')
)
```

圆形按钮

```
 Container(
                height: 80,
                child: ElevatedButton(
                    style: ButtonStyle(
                        backgroundColor: MaterialStateProperty.all(Colors.blue),
```

```
                        foregroundColor:
                            MaterialStateProperty.all(Colors.white),
                        elevation: MaterialStateProperty.all(20),
                        shape: MaterialStateProperty.all(
                          CircleBorder(side: BorderSide(color: Colors.white)),
                        )),
                    onPressed: () {
                      print("圆形按钮");
                    },
                    child: const Text('圆形按钮')),
)
```

19.10 修改OutlinedButton边框

```
Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: <Widget>[
              Expanded(
                child: Container(
                  margin: EdgeInsets.all(20),
                  height: 50,
                  child: OutlinedButton(
                      style: ButtonStyle(
                          foregroundColor:
                              MaterialStateProperty.all(Colors.black),
                          side: MaterialStateProperty.all(
                              const BorderSide(width: 1, color: Colors.red))),
                      onPressed: () {},
                      child: const Text("注册 配置边框")),
                ),
              )
            ],
)
```

二十、Flutter Wrap组件

Wrap可以实现流布局，单行的Wrap跟Row表现几乎一致，单列的Wrap则跟Column表现几乎一致。但Row与Column都是单行单列的，Wrap则突破了这个限制，mainAxis上空间不足时，则向crossAxis上去扩展显示。

属性	说明
direction	主轴的方向，默认水平
alignment	主轴的对其方式
spacing	主轴方向上的间距
textDirection	文本方向
verticalDirection	定义了children摆放顺序，默认是down，见Flex相关属性介绍。
runAlignment	run的对齐方式。run可以理解为新的行或者列，如果是水平方向布局的话，run可以理解为新的一行
runSpacing	run的间距

20.1、自定义一个按钮组件

```
class LayoutDemo extends StatelessWidget {
  const LayoutDemo({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Button("第一集",onPressed: (){});
  }
}

class Button extends StatelessWidget {

  String text;
```

```
  void Function()? onPressed;

  Button(this.text,{Key? key,required this.onPressed}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed:onPressed,
      style: ButtonStyle(
        backgroundColor: MaterialStateProperty.all(const Color.fromARGB(255,
236, 233, 233)),
        foregroundColor: MaterialStateProperty.all(Colors.black45),
      ),
      child: Text(text),
    );
  }
}
```

20.2、Wrap组件的使用

```
class LayoutDemo extends StatelessWidget {
  const LayoutDemo({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding:const EdgeInsets.all(3),
      child: Wrap(
        spacing: 5,
        runSpacing: 5,
        // direction: Axis.vertical,
        // alignment:WrapAlignment.start,
        // runAlignment: WrapAlignment.center,
        children: <Widget>[
          Button("第1集1111", onPressed: () {}),
          Button("第2集", onPressed: () {}),
          Button("第3集", onPressed: () {}),
          Button("第4集", onPressed: () {}),
          Button("第5集", onPressed: () {}),
          Button("第6集", onPressed: () {}),
          Button("第7集", onPressed: () {}),
          Button("第8集", onPressed: () {}),
          Button("第9集", onPressed: () {}),
          Button("第10集", onPressed: () {}),
          Button("第11集", onPressed: () {}),
          Button("第12集", onPressed: () {}),
          Button("第13集", onPressed: () {}),
          Button("第14集", onPressed: () {}),
          Button("第15集", onPressed: () {}),
          Button("第16集", onPressed: () {}),
          Button("第17集", onPressed: () {}),
          Button("第18集", onPressed: () {}),

        ],
      ),
```

```
    );
  }
}

class Button extends StatelessWidget {
  String text;
  void Function()? onPressed;

  Button(this.text, {Key? key, required this.onPressed}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: onPressed,
      style: ButtonStyle(
        backgroundColor:
            MaterialStateProperty.all(const Color.fromARGB(255, 236, 233, 233)),
        foregroundColor: MaterialStateProperty.all(Colors.black45),
      ),
      child: Text(text),
    );
  }
}
```

20.3、Wrap组件搜索页面布局


```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
```

```
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const LayoutDemo(),
      ),
    );
  }
}

class LayoutDemo extends StatelessWidget {
  const LayoutDemo({Key? key}) : super(key: key);
  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(10),
      child: ListView(children: [
        Row(
          children: [
            Text(
              "热搜",
              style: Theme.of(context).textTheme.headline6,
            )
          ],
        ),
        const Divider(),
        Wrap(
          spacing: 10,
          runSpacing: 12,
          children: [
            Button("女装", onPressed: () {}),
            Button("笔记本", onPressed: () {}),
            Button("玩具", onPressed: () {}),
            Button("文学", onPressed: () {}),
            Button("女装", onPressed: () {}),
            Button("时尚", onPressed: () {}),
            Button("女装", onPressed: () {}),
            Button("女装", onPressed: () {}),
          ],
        ),
        const SizedBox(height: 10),
        Row(
          children: [
            Text(
              "历史记录",
              style: Theme.of(context).textTheme.headline6,
            )
          ],
        ),
        const Divider(),
        Column(
          children: const [
            ListTile(
              title: Text("女装"),
            ),
            Divider(),
            ListTile(
              title: Text("时尚"),
            ),
```

```
            Divider(),
          ],
        ),
        const SizedBox(height: 40),
        Padding(
          padding: const EdgeInsets.all(20),
          child: OutlinedButton.icon(
              onPressed: () {},
              style: ButtonStyle(
                foregroundColor: MaterialStateProperty.all(Colors.black38)
              ),
              icon: const Icon(Icons.delete),
              label: const Text("清空历史记录")),


        )
      ]),
    );
  }
}

class Button extends StatelessWidget {
  String text;
  void Function()? onPressed;

  Button(this.text, {Key? key, required this.onPressed}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return ElevatedButton(
      onPressed: onPressed,
      style: ButtonStyle(
        backgroundColor:
            MaterialStateProperty.all(const Color.fromARGB(255, 236, 233, 233)),
        foregroundColor: MaterialStateProperty.all(Colors.black45),
      ),
      child: Text(text),
    );
  }
}
```

二十一、Flutter StatelessWidget 、StatefulWidget

在Flutter中自定义组件其实就是一个类，这个类需要继承StatelessWidget/StatefulWidget。

StatelessWidget是无状态组件，状态不可变的widget
StatefulWidget是有状态组件，持有的状态可能在widget生命周期改变。

通俗的讲：如果我们想改变页面中的数据的话这个时候就需要用到StatefulWidget

21.1 StatefulWidget实现一个计数器的功能

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  int countNum=0;
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Column(
        children: [
          const SizedBox(height: 40),
          Text("$countNum",style: Theme.of(context).textTheme.titleLarge),
           const SizedBox(height: 100,),
           ElevatedButton(onPressed: (){
            setState(() {
              countNum++;
            });

           }, child: const Text("增加"))
        ],
      ),
    );
```

```
  }
}
```

21.2 StatefulWidget实现一个动态列表

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const HomePage(),
      ),
    );
  }
}

class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  List<String> list = [];
  @override
  Widget build(BuildContext context) {
    return ListView(
      children: [
        Column(
          children: list.map((value) {
            return ListTile(
              title: Text(value),
            );
          }).toList(),
        ),
```

```
        const SizedBox(
          height: 40,
        ),
        Padding(
          padding: const EdgeInsets.all(40),
          child: ElevatedButton(
              onPressed: () {
                setState(() {
                  list.add("新增一条数据");
                });
              },
              child: const Text("增加")),
        )
      ],
    );
  }
}
```

二十二、Scaffold属性 BottomNavigationBar 自定义底部导航

22.1、BottomNavigationBar 组件介绍

BottomNavigationBar 是底部导航条，可以让我们定义底部Tab切换，bottomNavigationBar是Scaffold组件的参数。

BottomNavigationBar 常见的属性

属性名	说明
items	List 底部导航条按钮集合
iconSize	icon
currentIndex	默认选中第几个
onTap	选中变化回调函数
fixedColor	选中的颜色
type	BottomNavigationBarType.fixed BottomNavigationBarType.shifting

22.2、BottomNavigationBar 自定义底部导航


```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: const Text("Flutter App")),
        body: const Center(
          child: Text("我是一个文本"),
        ),
        bottomNavigationBar: BottomNavigationBar(
          items: const [
          BottomNavigationBarItem(
```

```
            icon: Icon(Icons.home),
            label: "首页"
          ),
           BottomNavigationBarItem(
            icon: Icon(Icons.category),
            label: "分类"
          ),
           BottomNavigationBarItem(
            icon: Icon(Icons.settings),
            label: "设置"
          ),
        ]),
        ),
      );
  }
}
```

22.3、BottomNavigationBar 底部菜单选中

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const Tabs(),
    );
  }
}

class Tabs extends StatefulWidget {
  const Tabs({super.key});

  @override
  State<Tabs> createState() => _TabsState();
}

class _TabsState extends State<Tabs> {
  int _currentIndex = 0;
  @override
  Widget build(BuildContext context) {
    return Scaffold(
```

```
      appBar: AppBar(title: const Text("Flutter App")),
      body: const Center(
        child: Text("我是一个文本"),
      ),
      bottomNavigationBar: BottomNavigationBar(
          currentIndex: _currentIndex,
          onTap: (v) {
            setState(() {

              _currentIndex = v;
            });
          },
          items: const [
            BottomNavigationBarItem(icon: Icon(Icons.home), label: "首页"),
            BottomNavigationBarItem(icon: Icon(Icons.category), label: "分类"),
            BottomNavigationBarItem(icon: Icon(Icons.settings), label: "设置"),
          ]),
    );
  }
}
```

22.4、BottomNavigationBar 自定义底部导航实现页面切换

看教程演示...

二十三、Scaffold属性 FloatingActionButton实现类似闲鱼App底部导航凸起按钮

23.1、FloatingActionButton详解

FloatingActionButton简称FAB ,可以实现浮动按钮，也可以实现类似闲鱼app的底部凸起导航

属性名称	属性值
child	子视图，一般为Icon，不推荐使用文字
tooltip	FAB被长按时显示，也是无障碍功能
backgroundColor	背景颜色
elevation	未点击的时候的阴影
hignlightElevation	点击时阴影值，默认12.0
onPressed	点击事件回调
shape	可以定义FAB的形状等
mini	是否是mini类型默认false

23.2、实现类似闲鱼App底部导航凸起按钮

```
import 'package:flutter/material.dart';
import './tabs/home.dart';
import './tabs/category.dart';
import './tabs/message.dart';
import './tabs/setting.dart';
import './tabs/user.dart';

class Tabs extends StatefulWidget {
  const Tabs({super.key});

  @override
  State<Tabs> createState() => _TabsState();
}

class _TabsState extends State<Tabs> {
  int _currentIndex = 0;
  final List<Widget> _pages = const [
    HomePage(),
    CategoryPage(),
    MessagePage(),
    SettingPage(),
    UserPage()
  ];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(title: const Text("Flutter App")),
      body: _pages[_currentIndex],
      bottomNavigationBar: BottomNavigationBar(
          fixedColor: Colors.red, //选中的颜色
          // iconSize:35,           //底部菜单大小
          currentIndex: _currentIndex, //第几个菜单选中
          type: BottomNavigationBarType.fixed, //如果底部有4个或者4个以上的菜单的时候就需要配置这个参数
          onTap: (index) {
            //点击菜单触发的方法
            //注意
            setState(() {
              _currentIndex = index;
            });
          },
          items: const [
            BottomNavigationBarItem(icon: Icon(Icons.home), label: "首页"),
            BottomNavigationBarItem(icon: Icon(Icons.category), label: "分类"),
            BottomNavigationBarItem(icon: Icon(Icons.message), label: "消息"),
            BottomNavigationBarItem(icon: Icon(Icons.settings), label: "设置"),
            BottomNavigationBarItem(icon: Icon(Icons.people), label: "用户")
          ]),
      floatingActionButton: Container(
        height: 60,
        width: 60,
        padding: const EdgeInsets.all(4),
        margin: const EdgeInsets.only(top: 4),

        decoration: BoxDecoration(
          color: Colors.white,
          borderRadius: BorderRadius.circular(30)
        ),
```

```
        child: FloatingActionButton(
          backgroundColor: _currentIndex==2?Colors.red:Colors.blue,
          onPressed: () {
            setState(() {
              _currentIndex=2;
            });
          },
          child: const Icon(Icons.add),
        ),
      ),
      floatingActionButtonLocation: FloatingActionButtonLocation.centerDocked,
    );
  }
}
```

二十四、Scaffold属性 抽屉菜单Drawer

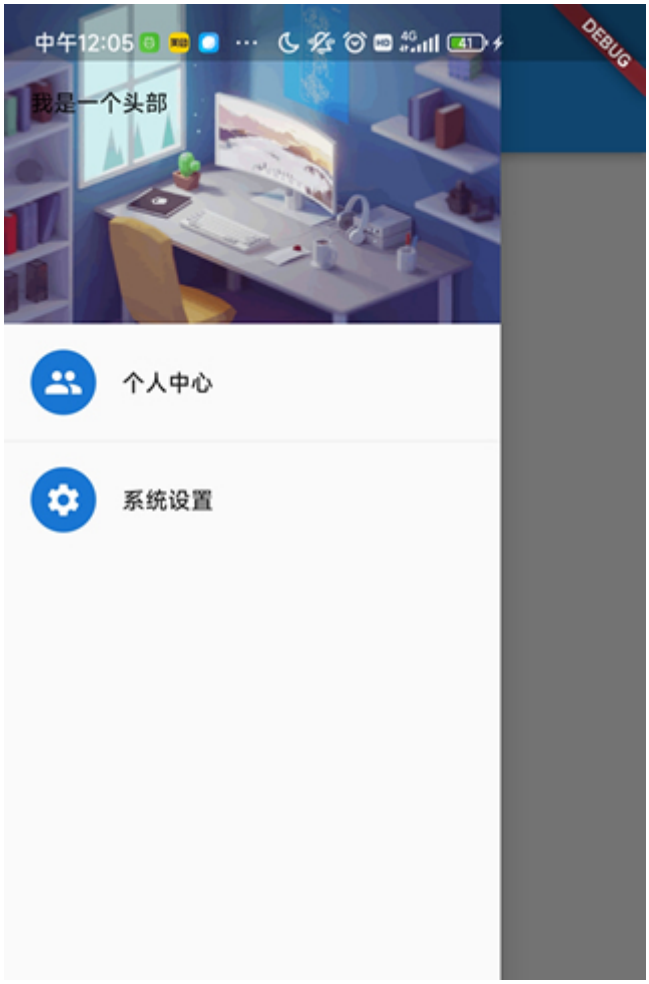
在Scaffold组件里面传入drawer参数可以定义左侧边栏，传入endDrawer可以定义右侧边栏。侧边栏默认是隐藏的，我们可以通过手指滑动显示侧边栏，也可以通过点击按钮显示侧边栏。

```
return Scaffold(
        appBar: AppBar(
          title: Text("Flutter App"),
        ),

        drawer: Drawer(
          child: Text('左侧边栏'),
        ),
        endDrawer:  Drawer(
          child: Text('右侧侧边栏'),
        ),
);
```

24.1、Flutter DrawerHeader

常见属性:

属性	描述
decoration	设置顶部背景颜色
child	配置子元素
padding	内边距
margin	外边距

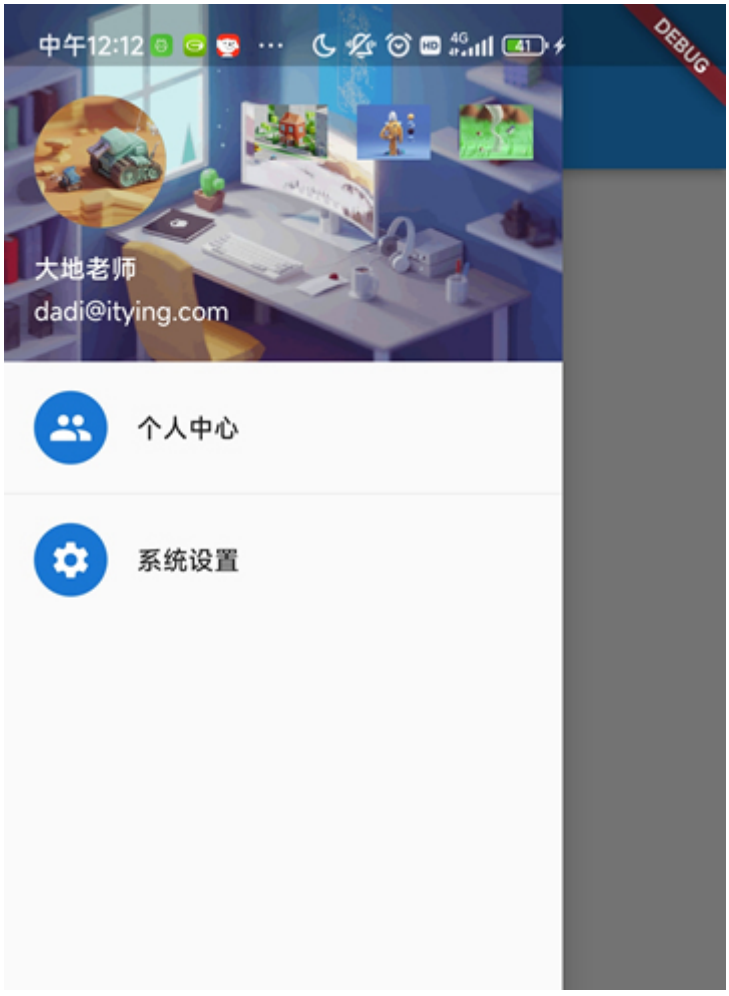



```
drawer: Drawer(
          child: Column(
        children: <Widget>[
          DrawerHeader(
            decoration: const BoxDecoration(
                color: Colors.yellow,
                image: DecorationImage(
                    image: NetworkImage(
                        "https://www.itying.com/images/flutter/2.png"),
                    fit: BoxFit.cover)),
            child: ListView(
              children: <Widget>[Text('我是一个头部')],
            ),
          ),
          const ListTile(
            title: Text("个人中心"),
            leading: CircleAvatar(child: Icon(Icons.people)),
```

```
          ),
          const Divider(),
          const ListTile(
            title: Text("系统设置"),
            leading: CircleAvatar(child: Icon(Icons.settings)),
          )
        ],
))
```

24.2、Flutter UserAccountsDrawerHeader

属性	描述
decoration	设置顶部背景颜色
accountName	账户名称
accountEmail	账户邮箱
currentAccountPicture	用户头像
otherAccountsPictures	用来设置当前账户其他账户头像
margin	

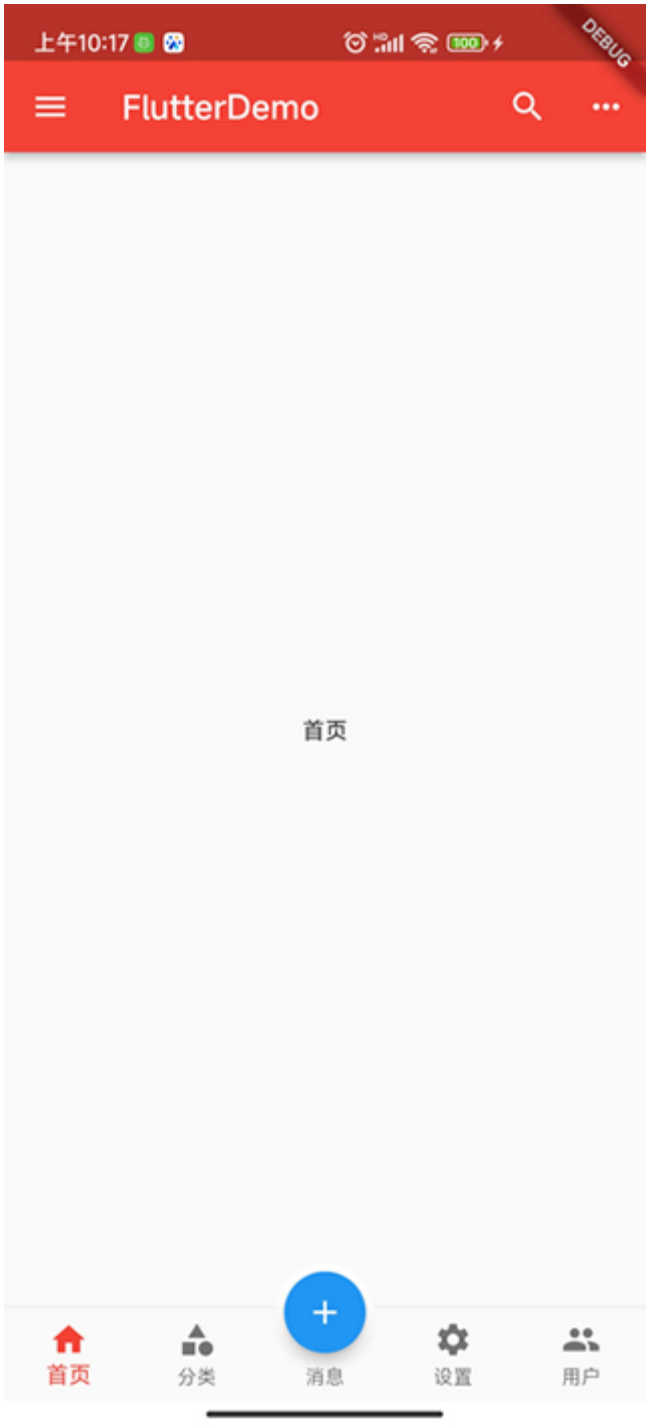


```
drawer: Drawer(
          child: Column(
            children: <Widget>[
              UserAccountsDrawerHeader(
                  accountName:const Text("大地老师") ,
                  accountEmail:const Text("dadi@itying.com") ,
                  currentAccountPicture: const CircleAvatar(
                    backgroundImage:
NetworkImage("https://www.itying.com/images/flutter/3.png"),

                  ),
                  decoration: const BoxDecoration(
                    color: Colors.yellow,
                    image:DecorationImage(
                      image:
NetworkImage("https://www.itying.com/images/flutter/2.png"),
                      fit:BoxFit.cover
                    )

                  ),
                  otherAccountsPictures: <Widget>[

Image.network("https://www.itying.com/images/flutter/4.png"),

Image.network("https://www.itying.com/images/flutter/5.png"),
                    Image.network("https://www.itying.com/images/flutter/6.png")
                  ],

              ),
              const ListTile(
                title: Text("个人中心"),
                leading: CircleAvatar(
                  child: Icon(Icons.people)
                ),
              ),
              const Divider(),
              const ListTile(
                title: Text("系统设置"),
                leading: CircleAvatar(
                  child: Icon(Icons.settings)
                ),
              )
            ],
          )
        )
```

二十五、Flutter AppBar TabBar TabBarView

25.1、AppBar自定义顶部按钮图标、颜色

属性	描述
leading	在标题前面显示的一个控件，在首页通常显示应用的 logo；在其他界面通常显示为返回按钮
title	标题，通常显示为当前界面的标题文字，可以放组件
actions	通常使用 IconButton 来表示，可以放按钮组
bottom	通常放tabBar，标题下面显示一个 Tab 导航栏
backgroundColor	导航背景颜色
iconTheme	图标样式
centerTitle	标题是否居中显示




```
AppBar(
            backgroundColor:Colors.red,
            leading:IconButton(
                icon: const Icon(Icons.menu),
                onPressed: (){
                  print('menu Pressed');
                }
            ),
            title: const Text('FlutterDemo'),
            actions: <Widget>[
              IconButton(
                icon: const Icon(Icons.search),
                onPressed: (){
                  print('Search Pressed');
                }
              ) ,
              IconButton(
```

```
                icon: const Icon(Icons.more_horiz),
                onPressed: (){
                  print('more_horiz Pressed');
                }
              )
            ],
)
```

25.2、Flutter AppBar结合TabBar实现顶部Tab切换

TabBar常见属性：

属性	描述
tabs	显示的标签内容，一般使用Tab对象,也可以是其他的Widget
controller	TabController对象
isScrollable	是否可滚动
indicatorColor	指示器颜色
indicatorWeight	指示器高度
indicatorPadding	底部指示器的Padding
indicator	指示器decoration，例如边框等
indicatorSize	指示器大小计算方式，TabBarIndicatorSize.label跟文字等宽,TabBarIndicatorSize.tab跟每个tab等宽
labelColor	选中label颜色
labelStyle	选中label的Style
labelPadding	每个label的padding值
unselectedLabelColor	未选中label颜色
unselectedLabelStyle	未选中label的Style

25.3 Tabbar TabBarView实现类似头条顶部导航

1、混入SingleTickerProviderStateMixin

```
class _HomePageState extends State<HomePage> with
SingleTickerProviderStateMixin{}
```

2、定义TabController

```
late TabController _tabController;

void initState() {
   super.initState();
   _tabController = TabController(length: 8, vsync: this);

   _tabController.addListener(() {
     if (_tabController.animation!.value == _tabController.index) {
       print(_tabController.index); //获取点击或滑动页面的索引值
     }
   });
 }
```

3、配置TabBar和TabBarView

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const HomePage(),
    );
  }
}

class HomePage extends StatefulWidget {
  const HomePage({Key? key}) : super(key: key);

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> with SingleTickerProviderStateMixin
{
  late TabController _tabController;

  @override
  void initState() {
    // TODO: implement initState
    super.initState();
    _tabController=TabController(length: 3, vsync: this);
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text("Flutter App"),
          bottom: TabBar(
            controller: _tabController,
            tabs: const [
            Tab(child:Text("热门")),
            Tab(child:Text("推荐")),
            Tab(child:Text("视频"))
          ],),
        ),
        body: TabBarView(
```

```
          controller: _tabController,
          children: const [Text("热门"), Text("推荐"), Text("视频")]));
  }
}
```

25.4 BottomNavigationBar 的页面中使用Tabbar

```
import 'package:flutter/material.dart';
import '../../tools/KeepAliveWrapper.dart';

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage>
    with SingleTickerProviderStateMixin {
  late TabController _tabController;

  @override
  void dispose() {   //生命周期函数
    // TODO: implement dispose
    super.dispose();
    _tabController.dispose();
  }

  @override
 void initState() {
    super.initState();
    _tabController = TabController(length: 8, vsync: this);
    _tabController.addListener(() {
      if (_tabController.animation!.value == _tabController.index) {
        print(_tabController.index); //获取点击或滑动页面的索引值
      }
    });
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: PreferredSize(
          preferredSize: const Size.fromHeight(40),
          child: AppBar(
              backgroundColor: const Color.fromARGB(255, 253, 247, 247),
              elevation: 10,
              title: SizedBox(
                height: 30,
```

```
                child: TabBar(
                    // isScrollable: true, //如果多个按钮的话可以滑动
                    indicatorColor: Colors.red,
                    labelColor: Colors.red,
                    unselectedLabelColor: Colors.white,
                    indicatorSize: TabBarIndicatorSize.label,
                    controller: _tabController, //注意
                    tabs: const [
                      Tab(
                          child: Text("热销",
                              style: TextStyle(color: Colors.black))),
                      Tab(
                          child:
                              Text("推荐", style: TextStyle(color:
Colors.black)))
                    ]),
              ))),
      body: TabBarView(controller: _tabController, //注意
          children: [
            KeepAliveWrapper(
                child: ListView(
              children: const <Widget>[
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("第一个tab")),
                ListTile(title: Text("最后一个tab")),
              ],
            )),
            KeepAliveWrapper(
                child: ListView(
              children: const <Widget>[
                ListTile(title: Text("第二个tab")),
                ListTile(title: Text("第二个tab")),
                ListTile(title: Text("第二个tab"))
              ],
            ))
          ]),
    );
  }
}
```

25.5 preferredSize组件

PreferredSize可以改变appBar的高度

```
Scaffold(
        appBar: PreferredSize(
          preferredSize: Size.fromHeight(50),
          child:  AppBar(
            ....
          )
        ),
        body: Test(),
      )
```

25.6 自定义KeepAliveWrapper 缓存页面

AutomaticKeepAliveClientMixin 可以快速的实现页面缓存功能，但是通过混入的方式实现不是很优雅， 所以我们有必要对AutomaticKeepAliveClientMixin 混入进行封装

```
import 'package:flutter/material.dart';

class KeepAliveWrapper extends StatefulWidget {
  const KeepAliveWrapper(
      {Key? key, @required this.child, this.keepAlive = true})
      : super(key: key);

  final Widget? child;
  final bool keepAlive;

  @override
  State<KeepAliveWrapper> createState() => _KeepAliveWrapperState();
}

class _KeepAliveWrapperState extends State<KeepAliveWrapper>
    with AutomaticKeepAliveClientMixin {
  @override
  Widget build(BuildContext context) {
    return widget.child!;
  }

  @override
  bool get wantKeepAlive => widget.keepAlive;

  @override
  void didUpdateWidget(covariant KeepAliveWrapper oldWidget) {
    if (oldWidget.keepAlive != widget.keepAlive) {
      // keepAlive 状态需要更新，实现在 AutomaticKeepAliveClientMixin 中
      updateKeepAlive();
    }
```

```
    super.didUpdateWidget(oldWidget);
  }
}
```

25.7 监听TabController改变事件

```
void initState() {

  super.initState();

  _tabController = TabController(length: 8, vsync: this);

  //监听_tabController的改变事件

  _tabController.addListener(() {

   if (_tabController.animation!.value==_tabController.index){

     print(_tabController.index);    //获取点击或滑动页面的索引值

   }

  });

 }
```

25.8 MaterialApp 去掉debug图标

```
 return MaterialApp(
      debugShowCheckedModeBanner:false ,  //去掉debug图标
      home:Tabs(),
      ...
 );
```

二十六、Flutter中的路由

26.1 、Flutter 路由介绍

Flutter中的路由通俗的讲就是页面跳转。在Flutter中通过Navigator组件管理路由导航。

并提供了管理堆栈的方法。如：Navigator.push和Navigator.pop

Flutter中给我们提供了两种配置路由跳转的方式：1、基本路由 2、命名路由

26.2 、Flutter 中的普通路由使用

比如我们现在想从HomePage组件跳转到SearchPage组件。

1、需要在HomPage中引入SearchPage.dart

```
import '../SearchPage.dart';
```

2、在HomePage中通过下面方法跳转

```
 Center(
        child: ElevatedButton(onPressed: (){
            Navigator.of(context).push(
              MaterialPageRoute(builder: (context){
                return const SearchPage();
              })
            );
        }, child: const Text("跳转到搜索页面")),
)
```

26.3、Flutter 中的普通路由跳转传值

跳转传值和调用组件传值的实现方法是一样的

1、定义一个SearchPage接收传值

```
import 'package:flutter/material.dart';
class SearchPage extends StatefulWidget {
  final String title;
  const SearchPage({super.key,this.title="Search Page"});

  @override
  State<SearchPage> createState() => _SearchPageState();
}

class _SearchPageState extends State<SearchPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text(widget.title),
        centerTitle: true,
      ),
      body: const Center(
        child: Text("组件居中"),
      ),
    );
  }
}
```

2、**跳转页面实现传值**

```
 Center(
        child: ElevatedButton(onPressed: (){
            Navigator.of(context).push(
              MaterialPageRoute(builder: (context){
                return const SearchPage(title: "搜索页面",);
              })
            );
        }, child: const Text("跳转到搜索页面")),
)
```

26.4、Flutter 中的命名路由

1、main.dart中配置路由

```
import 'package:flutter/material.dart';
import './pages/tabs.dart';
import './pages/search.dart';
import './pages/form.dart';
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner:false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      // home:const Tabs() ,
       routes: {
        '/':(contxt)=>const Tabs(),
        '/search':(contxt) => const SearchPage(),
        '/form': (context) => const FormPage(),
      },
    );
  }
}
```

2、跳转路由

```
ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/form');
              },
              child: const Text("跳转到form页面")
            )
```

26.5、Flutter 中的命名路由传值

官方文档：https://flutter.dev/docs/cookbook/navigation/navigate-with-arguments

26.5.1、配置onGenerateRoute

```
import 'package:flutter/material.dart';
import './pages/tabs.dart';
import './pages/search.dart';
import './pages/form.dart';

void main() {
  runApp(MyApp());
}

class MyApp extends StatelessWidget {
  //1、定义Map类型的routes
  Map routes = {
    '/': (contxt) => const Tabs(),
    '/search': (contxt) => const SearchPage(),
    '/form': (context, {arguments}) => FormPage(arguments: arguments),
  };
  MyApp({Key? key}) : super(key: key);
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
       initialRoute: '/',
      //2、调用onGenerateRoute处理
      onGenerateRoute: (RouteSettings settings) {
        // 统一处理
        final String? name = settings.name;
        final Function? pageContentBuilder = routes[name];
        if (pageContentBuilder != null) {
          if (settings.arguments != null) {
            final Route route = MaterialPageRoute(
                builder: (context) =>
                    pageContentBuilder(context, arguments: settings.arguments));
            return route;
```

```
          } else {
            final Route route = MaterialPageRoute(
                builder: (context) => pageContentBuilder(context));
            return route;
          }
        }
        return null;
      },
    );
  }
}
```

26.5.2、定义页面接收arguments传参

```
import 'package:flutter/material.dart';

class FormPage extends StatefulWidget {
  final Map arguments;
  const FormPage({super.key,required this.arguments});

  @override
  State<FormPage> createState() => _FormPageState();
}

class _FormPageState extends State<FormPage> {
  @override
  void initState() {
    // TODO: implement initState

    print(widget.arguments);
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: const Text("我是一个Form表单演示页面"),
      ),
    );
  }
}
```

26.5.3、跳转页面实现传参

```
ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/form',arguments: {
                   "title": "搜索页面",
                });
              },
             child: const Text("form")
)
```

26.6、Flutter 中的命名路由单独抽离到一个文件

26.6.1、新建routers/routers.dart 配置路由

```
import 'package:flutter/material.dart';
import '../pages/tabs.dart';
import '../pages/search.dart';
import '../pages/form.dart';

final Map<String,Function> routes = {
  '/': (contxt) => const Tabs(),
  '/search': (contxt) => const SearchPage(),
  '/form': (context, {arguments}) => FormPage(arguments: arguments),
};

var onGenerateRoute = (RouteSettings settings) {
  // 统一处理
  final String? name = settings.name;
  final Function? pageContentBuilder = routes[name];
  if (pageContentBuilder != null) {
    if (settings.arguments != null) {
      final Route route = MaterialPageRoute(
          builder: (context) =>
              pageContentBuilder(context, arguments: settings.arguments));
      return route;
    } else {
      final Route route =
          MaterialPageRoute(builder: (context) => pageContentBuilder(context));
      return route;
    }
  }
  return null;
};
```

26.6.2、修改main.dart

```
import 'package:flutter/material.dart';
import './routers/routers.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);
  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
```

```
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
       initialRoute: '/',
      onGenerateRoute: onGenerateRoute,
    );
  }
}
```

26.6.3、实现页面跳转传值

```
ElevatedButton(
              onPressed: () {
                Navigator.pushNamed(context, '/form',arguments: {
                   "title": "搜索页面",
                });
              },
             child: const Text("form")
)
```

26.7、Flutter 返回上一级路由

```
Navigator.of(context).pop();
```

26.8、Flutter 中替换路由

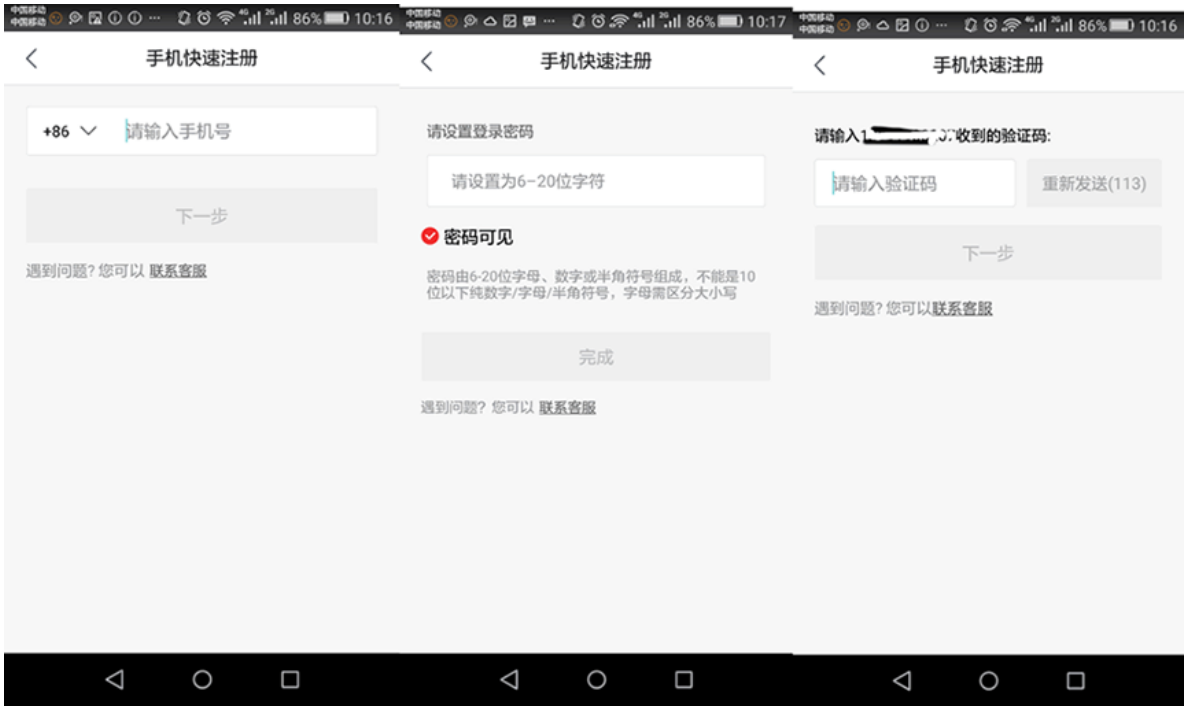
比如我们从用户中心页面跳转到了registerFirst页面，然后从registerFirst页面通过pushReplacementNamed跳转到了registerSecond页面。这个时候当我们点击registerSecond的返回按钮的时候它会直接返回到用户中心。

```
Navigator.of(context).pushReplacementNamed('/registerSecond');
```

26.9、Flutter 返回到根路由

比如我们从用户中心跳转到registerFirst页面，然后从registerFirst页面跳转到registerSecond页面，然后从registerSecond跳转到了registerThird页面。这个时候我们想的是registerThird注册成功后返回到用户中心。 这个时候就用到了返回到根路由的方法。

```
  Navigator.of(context).pushAndRemoveUntil(
                        MaterialPageRoute(builder: (BuildContext context) {
                      return const Tabs();
}), (route) => false);
```



26.10、Flutter Android 和Ios使用同样风格的路由跳转

Material组件库中提供了一个MaterialPageRoute组件，它可以使用和平台风格一致的路由切换动画，如在iOS上会左右滑动切换，而在Android上会上下滑动切换，CupertinoPageRoute是Cupertino组件库提供的iOS风格的路由切换组件如果在Android上也想使用左右切换风格，可以使用CupertinoPageRoute。

1、routers.dart中引入cupertino.dart

```
import 'package:flutter/cupertino.dart';
```

2、MaterialPageRoute改为CupertinoPageRoute

```
import 'package:flutter/cupertino.dart';
import '../pages/tabs.dart';
import '../pages/shop.dart';
import '../pages/user/login.dart';
import '../pages/user/registerFirst.dart';
import '../pages/user/registerSecond.dart';
import '../pages/user/registerThird.dart';
//1、配置路由
Map routes = {
  "/": (contxt) => const Tabs(),
  "/login": (contxt) => const LoginPage(),
```

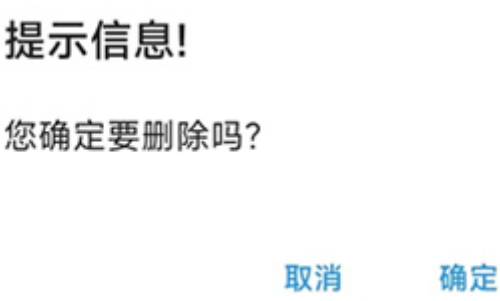
```
  "/registerFirst": (contxt) => const RegisterFirstPage(),
  "/registerSecond": (contxt) => const RegisterSecondPage(),
  "/registerThird": (contxt) => const RegisterThirdPage(),
  "/shop": (contxt, {arguments}) => ShopPage(arguments: arguments),
};

//2、配置onGenerateRoute  固定写法  这个方法也相当于一个中间件，这里可以做权限判断
var onGenerateRoute = (RouteSettings settings) {
  final String? name = settings.name; //  /news 或者 /search
  final Function? pageContentBuilder = routes[name];                          //
Function = (contxt) { return const NewsPage()}

  if (pageContentBuilder != null) {
    if (settings.arguments != null) {
      final Route route = CupertinoPageRoute(
          builder: (context) =>
              pageContentBuilder(context, arguments: settings.arguments));
      return route;
    } else {
      final Route route =
          CupertinoPageRoute(builder: (context) => pageContentBuilder(context));

      return route;
    }
  }
  return null;
};
```

26.11、全局配置主题

```
return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
        appBarTheme: const AppBarTheme(
          centerTitle: true,
        )
      ),
      initialRoute: "/",
      onGenerateRoute: onGenerateRoute,
);
```

二十七、Flutter Dialog

27.1、AlertDialog




```
_alertDialog() async {
    var result = await showDialog(
        barrierDismissible: false, //表示点击灰色背景的时候是否消失弹出框
        context: context,
        builder: (context) {
          return AlertDialog(
            title: const Text("提示信息!"),
            content: const Text("您确定要删除吗?"),
            actions: <Widget>[
              TextButton(
                child: const Text("取消"),
                onPressed: () {
                  print("取消");
                  Navigator.pop(context, 'Cancle');
                },
              ),
              TextButton(
                child: const Text("确定"),
                onPressed: () {
                  print("确定");
                  Navigator.pop(context, "Ok");
                },
              )
            ],
          );
        });

    print(result);
  }
```

27.2、SimpleDialog、SimpleDialogOption


```
_simpleDialog() async {
    var result = await showDialog(
        barrierDismissible: true, //表示点击灰色背景的时候是否消失弹出框
        context: context,
        builder: (context) {
          return SimpleDialog(
            title: const Text("请选择内容"),
            children: <Widget>[
              SimpleDialogOption(
                child: const Text("Option A"),
                onPressed: () {
                  print("Option A");
                  Navigator.pop(context, "A");
                },
              ),
              const Divider(),
              SimpleDialogOption(
                child: const Text("Option B"),
                onPressed: () {
                  print("Option B");
                  Navigator.pop(context, "B");
                },
              ),
              const Divider(),
              SimpleDialogOption(
                child: const Text("Option C"),
                onPressed: () {
                  print("Option C");
                  Navigator.pop(context, "C");
                },
              ),
            ],
          );
        });

    print(result);
  }
```

27.3、showModalBottomSheet


alert弹出框-AlertDialog
select弹出框-SimpleDialog
ActionSheet底部弹出框-showModalBottomSheet

分享 A

分享 B

分享 C


```
_modelBottomSheet() async {
    var result = await showModalBottomSheet(
        context: context,
        builder: (context) {
          return SizedBox(
            height: 220,
            child: Column(
              children: <Widget>[
                ListTile(
                  title: const Text("分享 A"),
                  onTap: () {
                    Navigator.pop(context, "分享 A");
                  },
                ),
                const Divider(),
                ListTile(
                  title: const Text("分享 B"),
                  onTap: () {
```

```
                  Navigator.pop(context, "分享 B");
                },
              ),
              const Divider(),
              ListTile(
                title: const Text("分享 C"),
                onTap: () {
                  Navigator.pop(context, "分享 C");
                },
              )
            ],
          ),
        );
      });

  print(result);
}
```

27.4、Flutter Toast

27.4.1、fluttertoast的使用

https://pub.dev/packages/fluttertoast

```
# add this line to your dependencies
fluttertoast: ^8.0.9
```

```
import 'package:fluttertoast/fluttertoast.dart';
```

```
Fluttertoast.showToast(
        msg: "提示信息",
        toastLength: Toast.LENGTH_SHORT,
        gravity: ToastGravity.BOTTOM,
        timeInSecForIosWeb: 1,
        backgroundColor: Colors.black,
        textColor: Colors.white,
        fontSize: 16.0
    );
```

27.4.2、ftoast的使用

支持LINUX MACOS WEB WINDOWS的另一个插件

https://pub.dev/packages/ftoast

```
dependencies:
  ftoast: ^2.0.0
```

```
import 'package:ftoast/ftoast.dart';
```

```
 FToast.toast(
      context,
      msg: "This is Msg",
      subMsg: "Welcome to use FToast. This is subMsg!",
      subMsgStyle: const TextStyle(color: Colors.white, fontSize: 13),
    );
```

27.5、自定义Flutter Dialog 、Material组件、InkWell组件

自定义Dialog对象，需要继承Dialog类，尽管Dialog提供了child参数可以用来写视图界面，但是往往会达不到我们想要的效果，因为默认的Dialog背景框是满屏的。如果我们想完全定义界面，就需要重写build函数。下面我们通过两个案例给大家演示一下Dialog的使用。

27.5.1、自定义一个提示的Dialog

1、新建myDialog.dart

```
import 'dart:async';

import 'package:flutter/material.dart';

// ignore: must_be_immutable
class MyDialog extends Dialog {
  String title;
  String content;
  Function()? onClosed;

  MyDialog({Key? key, required this.title,required
this.onClosed,this.content=""}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Material(
      type: MaterialType.transparency,
      child: Center(
        child: Container(
          height: 300,
          width: 300,
          color: Colors.white,
           child: Column(
            children: <Widget>[
              Padding(
                padding: const EdgeInsets.all(10),
```

```
                child: Stack(
                  children: <Widget>[
                    Align(
                      alignment: Alignment.center,
                      child: Text(title),
                    ),
                    Align(
                      alignment: Alignment.centerRight,
                      child: InkWell(
                        onTap: onClosed,
                        child: const Icon(Icons.close),
                      ),
                    )
                  ],
                ),
              ),
              const Divider(),
              Container(
                padding: const EdgeInsets.all(10),
                width: double.infinity,
                child: Text(content,textAlign: TextAlign.left),
              )
            ],
          ),

      )),
    );
  }
}
```

2、调用Mydialog

```
void _myDialog() async {
    await showDialog(
        barrierDismissible: true, //表示点击灰色背景的时候是否消失弹出框
        context: context,
        builder: (context) {
          return MyDialog(
              title: '标题',
              onClosed: () {
                print("关闭");
                Navigator.of(context).pop();
              },
              content: "我是一个内容");
        });
  }
```

27.5.2、Flutter定时器 让dialog自动关闭

Flutter定时器

```
  const timeout = Duration(seconds: 3);
  var t=Timer.periodic(timeout, (timer) {
      print('afterTimer='+DateTime.now().toString()););
      // timer.cancel();  // 取消定时器
  });

  t.cancel();  // 取消定时器
```

组件销毁的时候取消定时器

```
  void dispose() {
    super.dispose();
    t.cancel();
  }
```

MyDialog过几秒后关闭

```
import 'dart:async';

import 'package:flutter/material.dart';

// ignore: must_be_immutable
class MyDialog extends Dialog {
  String title;
  String content;
  Function()? onClosed;

  MyDialog({Key? key, required this.title,required
this.onClosed,this.content=""}) : super(key: key);

 _showTimer(context){
    Timer.periodic(
      Duration(milliseconds: 3000), (t) {
        print('关闭');
        Navigator.of(context).pop();
        t.cancel();
    });
  }

  @override
  Widget build(BuildContext context) {
    _showTimer(context);
    return Material(
      type: MaterialType.transparency,
      child: Center(
        child: Container(
          height: 300,
          width: 300,
          color: Colors.white,
           child: Column(
```

```
            children: <Widget>[
              Padding(
                padding: const EdgeInsets.all(10),
                child: Stack(
                  children: <Widget>[
                    Align(
                      alignment: Alignment.center,
                      child: Text(title),
                    ),
                    Align(
                      alignment: Alignment.centerRight,
                      child: InkWell(
                        onTap: onClosed,
                        child: const Icon(Icons.close),
                      ),
                    )
                  ],
                ),
              ),
              const Divider(),
              Container(
                padding: const EdgeInsets.all(10),
                width: double.infinity,
                child: Text(content,textAlign: TextAlign.left),
              )
            ],
          ),

      )),
    );
  }
}
```

二十八、Flutter PageView

Flutter中的轮动图以及抖音上下滑页切换视频功能等等，这些都可以通过 PageView 轻松实现

PageView常见属性：

属性	描述
scrollDirection	Axis.horizonta水平方向 Axis.vertical锤子方向
children	配置子元素
allowImplicitScrolling	缓存当前页面的前后两页
onPageChanged	page改变的时候触发

28.1、PageView 的使用


```
import 'package:flutter/material.dart';

class PageViewPage extends StatefulWidget {
  const PageViewPage({super.key});

  @override
  State<PageViewPage> createState() => _PageViewPageState();
}

class _PageViewPageState extends State<PageViewPage> {
  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
```

```
          title: const Text("pageview演示"),
        ),
        body: PageView(
          // scrollDirection: Axis.vertical, // 滑动方向为垂直方向
          children: [
            Center(
              child: Text("1",style: Theme.of(context).textTheme.headline1,),
            ),
            Center(
              child: Text("2",style: Theme.of(context).textTheme.headline1,),
            ),
            Center(
              child: Text("3",style: Theme.of(context).textTheme.headline1,),
            ),
             Center(
              child: Text("4",style: Theme.of(context).textTheme.headline1,),
            ),
             Center(
              child: Text("5",style: Theme.of(context).textTheme.headline1,),
            ),
            Center(
              child: Text("6",style: Theme.of(context).textTheme.headline1,),
            )
          ],
        ));
  }
}
```

28.2、PageView.builder

```
import 'package:flutter/material.dart';

class PageViewPage extends StatefulWidget {
  const PageViewPage({super.key});

  @override
  State<PageViewPage> createState() => _PageViewPageState();
}

class _PageViewPageState extends State<PageViewPage> {
  int itemCount=10;
  @override
  void initState() {
    // TODO: implement initState
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text("pageview演示"),
```

```
        ),

        body: PageView.builder(
          scrollDirection: Axis.vertical, // 滑动方向为垂直方向
          itemBuilder: (BuildContext context, int index) {
            return MyPage(text:"$index");
          },
          itemCount: 10,
        ));
  }
}

class MyPage extends StatefulWidget {
  final String text;
  const MyPage({super.key,required this.text});

  @override
  State<MyPage> createState() => _MyPageState();
}

class _MyPageState extends State<MyPage> {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text(widget.text,style: Theme.of(context).textTheme.headline1),
    );
  }
}
```

28.3、PageView上拉无限加载的实现思路

```
import 'package:flutter/material.dart';

class PageViewPage extends StatefulWidget {
  const PageViewPage({super.key});

  @override
  State<PageViewPage> createState() => _PageViewPageState();
}

class _PageViewPageState extends State<PageViewPage> {
  final List<Widget> _list = [];
  @override
  void initState() {
    // TODO: implement initState
    super.initState();
    for (var i = 0; i < 10; i++) {
      _list.add(MyPage(text:"$i"));
    }
  }

  @override
```

```
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text("pageview演示"),
        ),
        body: PageView(
          scrollDirection: Axis.vertical, // 滑动方向为垂直方向
          onPageChanged: (index){
            print(index);
            print(_list.length);
            if(index+2==_list.length){
              setState(() {
                for (var i = 0; i < 10; i++) {
                  _list.add(MyPage(text:"$i"));
                }
              });
            }
          },
          children:_list,
        ));
  }
}

class MyPage extends StatefulWidget {
  final String text;
  const MyPage({super.key,required this.text});

  @override
  State<MyPage> createState() => _MyPageState();
}

class _MyPageState extends State<MyPage> {
  @override
  Widget build(BuildContext context) {
    return Center(
      child: Text(widget.text,style: Theme.of(context).textTheme.headline1),
    );
  }
}
```

28.4、PageView 实现一个无限轮播的轮播图


```
import 'package:flutter/material.dart';

class PageViewPage extends StatefulWidget {
  const PageViewPage({super.key});

  @override
  State<PageViewPage> createState() => _PageViewPageState();
}

class _PageViewPageState extends State<PageViewPage> {
  List<Widget> pageList = [];
  @override
  void initState() {
    List listData = [
      {
        "imageUrl": 'https://www.itying.com/images/flutter/1.png',
      },
      {
        "imageUrl": 'https://www.itying.com/images/flutter/2.png',
      },
      {
        "imageUrl": 'https://www.itying.com/images/flutter/3.png',
      }
    ];
    for (int i = 0; i < listData.length; ++i) {
      pageList.add(PicturePage(
        url: listData[i]["imageUrl"],
      ));
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text("pageview演示"),
        ),
        body: ListView(
          children: [Swiper(pageList: pageList)],
        ));
  }
}
```

```
//Swiper组件
class Swiper extends StatefulWidget {
  final double width;
  final double height;
  final List<Widget> pageList;

  const Swiper(
      {super.key,
      this.width = double.infinity,
      this.height = 200,
      required this.pageList});
  @override
  State<Swiper> createState() => _SwiperState();
}

class _SwiperState extends State<Swiper> {
  int _currentPageIndex = 0;
  @override
  void initState() {
    super.initState();
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      children: [
        SizedBox(
          width: double.infinity,
          height: 200,
          child: PageView.builder(
              onPageChanged: (int index) {
                setState(() {
                  _currentPageIndex = index % (widget.pageList.length);
                });
              },
              itemCount: 10000,
              itemBuilder: (context, index) {
                return widget.pageList[index % (widget.pageList.length)];
              }),
        ),
        Positioned(
          bottom: 10,
          left: 0,
          right: 0,
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: List.generate(widget.pageList.length, (i) {
              return Container(
                margin: const EdgeInsets.fromLTRB(2, 0, 2, 0),
                width: 10,
                height: 10,
                decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    color: _currentPageIndex == i ? Colors.blue : Colors.grey),
              );
            }).toList(),
          ),
        ),
```

```
      ],
    );
  }
}

//PicturePage 图片页面
class PicturePage extends StatefulWidget {
  final String url;
  final double width;
  final double height;
  const PicturePage(
      {super.key,
      required this.url,
      this.width = double.infinity,
      this.height = 200});

  @override
  State<PicturePage> createState() => _PicturePageState();
}

class _PicturePageState extends State<PicturePage> {
  @override
  Widget build(BuildContext context) {
    print(widget.url);
    return SizedBox(
      width: widget.width,
      height: widget.height,
      child: Image.network(widget.url, fit: BoxFit.cover),
    );
  }
}
```

28.5、Flutter定时器

```
  const timeout = Duration(seconds: 3);
  var t=Timer.periodic(timeout, (timer) {
      print('afterTimer='+DateTime.now().toString()););
      // timer.cancel();  // 取消定时器
  });

  t.cancel();  // 取消定时器
```

组件销毁的时候取消定时器

```
  void dispose() {
    super.dispose();
    t.cancel();
  }
```

28.6、PageController animateToPage自动切换页面

```
import 'dart:async';

import 'package:flutter/material.dart';

class PageViewPage extends StatefulWidget {
  const PageViewPage({super.key});

  @override
  State<PageViewPage> createState() => _PageViewPageState();
}

class _PageViewPageState extends State<PageViewPage> {
  List<Widget> pageList = [];
  @override
  void initState() {
    List listData = [
      {
        "imageUrl": 'https://www.itying.com/images/flutter/1.png',
      },
      {
        "imageUrl": 'https://www.itying.com/images/flutter/2.png',
      },
      {
        "imageUrl": 'https://www.itying.com/images/flutter/3.png',
      }
    ];
    for (int i = 0; i < listData.length; ++i) {
      pageList.add(PicturePage(
        url: listData[i]["imageUrl"],
      ));
    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
        appBar: AppBar(
          title: const Text("pageview演示"),
        ),
        body: ListView(
          children: [Swiper(pageList: pageList)],
        ));
  }
}

//Swiper组件
class Swiper extends StatefulWidget {
  final double width;
  final double height;
  final List<Widget> pageList;

  const Swiper(
      {super.key,
      this.width = double.infinity,
      this.height = 200,
      required this.pageList});
  @override
```

```
  State<Swiper> createState() => _SwiperState();
}

class _SwiperState extends State<Swiper> {
  late PageController _pageController;
  int _currentPageIndex = 0;
  late Timer timer;
  @override
  void initState() {
    super.initState();
    _pageController = PageController(initialPage: 0);

    const timeout = Duration(seconds: 3);
    timer=Timer.periodic(timeout, (timer) {
      //跳转
      _pageController.animateToPage(
          (_currentPageIndex + 1) % (widget.pageList.length),
          curve: Curves.linear,
          duration: const Duration(milliseconds: 200));
      // timer.cancel();  // 取消定时器
    });

  }

  void dispose() {
    super.dispose();
    timer.cancel();
    _pageController.dispose();
  }

  @override
  Widget build(BuildContext context) {
    return Stack(
      children: [
        SizedBox(
          width: double.infinity,
          height: 200,
          child: PageView.builder(
              controller: _pageController,
              onPageChanged: (int index) {
                setState(() {
                  _currentPageIndex = index % (widget.pageList.length);
                });
              },
              itemCount: 10000,
              itemBuilder: (context, index) {
                return widget.pageList[index % (widget.pageList.length)];
              }),
        ),
        Positioned(
          bottom: 10,
          left: 0,
          right: 0,
          child: Row(
            mainAxisAlignment: MainAxisAlignment.center,
            children: List.generate(widget.pageList.length, (i) {
              return Container(
                margin: const EdgeInsets.fromLTRB(2, 0, 2, 0),
```

```
                width: 10,
                height: 10,
                decoration: BoxDecoration(
                    shape: BoxShape.circle,
                    color: _currentPageIndex == i ? Colors.blue : Colors.grey),
              );
            }).toList(),
          ),
        ),
      ],
    );
  }
}

//PicturePage 图片页面
class PicturePage extends StatefulWidget {
  final String url;
  final double width;
  final double height;
  const PicturePage(
      {super.key,
      required this.url,
      this.width = double.infinity,
      this.height = 200});

  @override
  State<PicturePage> createState() => _PicturePageState();
}

class _PicturePageState extends State<PicturePage> {
  @override
  Widget build(BuildContext context) {
    print(widget.url);
    return SizedBox(
      width: widget.width,
      height: widget.height,
      child: Image.network(widget.url, fit: BoxFit.cover),
    );
  }
}
```

28.7. AutomaticKeepAliveClientMixin 缓存PageView页面

通过上面的例子我们会发现 每次滑动的时候都会触发子组件中的 build方法 `print(widget.url);`

可见 PageView 默认并没有缓存功能，一旦页面滑出屏幕它就会被销毁 ,实际项目开发中对页面进行缓存是很常见的一个需求，下面我们就看看如何使用AutomaticKeepAliveClientMixin 缓存页面。

注意：使用时一定要注意是否必要，因为对所有列表项都缓存的会导致更多的内存消耗。

```
import 'package:flutter/material.dart';
import '../res/listData.dart';

class PageViewPage extends StatefulWidget {
  const PageViewPage({super.key});

  @override
  State<PageViewPage> createState() => _PageViewPageState();
}

class _PageViewPageState extends State<PageViewPage> {
  List<Widget> children = <Widget>[];
  @override
  void initState() {
    super.initState();
    for (int i = 0; i < listData.length; ++i) {
      children.add(PicturePage(
        url: listData[i]["imageUrl"],
      ));
    }
  }

  @override
  Widget build(BuildContext context) {
    print("build  ");

    return Scaffold(
        appBar: AppBar(
          title: const Text("pageview演示"),
        ),
        body: ListView(
          children: [
            AspectRatio(
              aspectRatio: 16 / 9,
              child: PageView(
                // scrollDirection: Axis.vertical, // 滑动方向为垂直方向
                children: children,
              ),
            ),
            const Text("text组件")
          ],
        ));
  }
}

//PicturePage子页面
class PicturePage extends StatefulWidget {
  final String url;
  const PicturePage({super.key, required this.url});

  @override
  State<PicturePage> createState() => _PicturePageState();
}
```

```
class _PicturePageState extends State<PicturePage> with
AutomaticKeepAliveClientMixin{
  @override
  Widget build(BuildContext context) {
    print(widget.url);

    return Center(
      child: AspectRatio(
        aspectRatio: 16 / 9,
        child: Image.network(widget.url, fit: BoxFit.cover),
      ),
    );
  }

  @override
  // TODO: implement wantKeepAlive
  bool get wantKeepAlive => true;
}
```

28.8 自定义KeepAliveWrapper 缓存页面

AutomaticKeepAliveClientMixin 可以快速的实现页面缓存功能，但是通过混入的方式实现不是很优雅， 所以我们有必要对AutomaticKeepAliveClientMixin 混入进行封装

```
import 'package:flutter/material.dart';

class KeepAliveWrapper extends StatefulWidget {
  const KeepAliveWrapper(
      {Key? key, @required this.child, this.keepAlive = true})
      : super(key: key);

  final Widget? child;
  final bool keepAlive;

  @override
  State<KeepAliveWrapper> createState() => _KeepAliveWrapperState();
}

class _KeepAliveWrapperState extends State<KeepAliveWrapper>
    with AutomaticKeepAliveClientMixin {
  @override
  Widget build(BuildContext context) {
    return widget.child!;
  }

  @override
  bool get wantKeepAlive => widget.keepAlive;

  @override
  void didUpdateWidget(covariant KeepAliveWrapper oldWidget) {
    if (oldWidget.keepAlive != widget.keepAlive) {
      // keepAlive 状态需要更新，实现在 AutomaticKeepAliveClientMixin 中
      updateKeepAlive();
    }
```

```
    super.didUpdateWidget(oldWidget);
  }
}

//这里使用covariant协变关键字,感兴趣可以研究一下dart  covariant
```

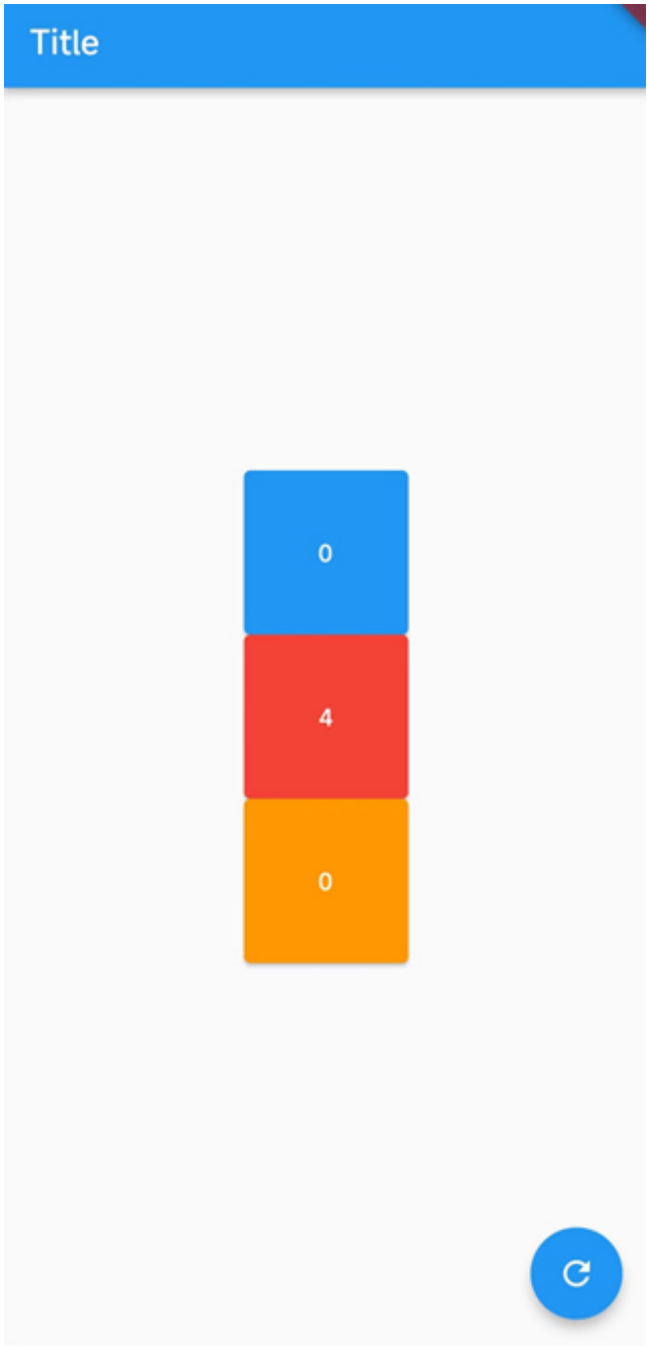
二十九、Flutter Key详解

我们平时一定接触过很多的 Widget，比如 Container、Row、Column 等，它们在我们绘制界面的过程中发挥着重要的作用。但是不知道你有没有注意到，在几乎每个 Widget 的构造函数中，都有一个共同的参数，它们通常在参数列表的第一个，那就是 Key。

在Flutter中，**Key是不能重复使用的**，所以Key一般用来做唯一标识。组件在更新的时候，其状态的保存主要是通过判断**组件的类型**或者**key值**是否一致。因此，当各组件的类型不同的时候，类型已经足够用来区分不同的组件了，此时我们可以不必使用key。但是如果同时存在多个同一类型的控件的时候，此时类型已经无法作为区分的条件了，我们就需要使用到key。

29.1、没有 Key 会发生什么奇怪现象

如下面例： 定义了一个StatefulWidget的Box，点击Box的时候可以改变Box里面的数字，当我们重新对Box排序的时候Flutter就无法识别到Box的变化了， 这是什么原因呢?




```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
```

```
      home: const MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  List<Widget> list = [
    Box(
      color: Colors.blue,
    ),
    Box(
      color: Colors.red,
    ),
    Box(
      color: Colors.orange,
    )
  ];
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {

          setState(() {
            list.shuffle();  //打乱list的顺序
          });
        },
        child: const Icon(Icons.refresh),
      ),
      appBar: AppBar(
        title: const Text('Title'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: list,
        ),
      ),
    );
  }
}

class Box extends StatefulWidget {
  Color color;
  Box({super.key, required this.color});

  @override
  State<Box> createState() => _BoxState();
}

class _BoxState extends State<Box> {
```

```
  int _count = 0;
  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 100,
      width: 100,
      child: ElevatedButton(
        style: ButtonStyle(
            backgroundColor: MaterialStateProperty.all(widget.color)),
        onPressed: () {
          setState(() {
            _count++;
          });
        },
        child: Center(
          child: Text("$_count"),
        ),
      ),
    );
  }
}
```

运行后我们发现改变list Widget顺序后，Widget颜色会变化，但是每个Widget里面的文本内容并没有变化，为什么会这样呢？当我们List重新排序后Flutter检测到了Widget的顺序变化，所以重新绘制List Widget，但是Flutter 发现List Widget 里面的元素没有变化，所以就没有改变Widget里面的内容。

把List 里面的Box的颜色改成一样，这个时候您重新对list进行排序，就很容易理解了。重新排序后虽然执行了setState，但是代码和以前是一样的，所以Flutter不会重构List Widget里面的内容, 也就是Flutter没法通过Box里面传入的参数来识别Box是否改变。如果要让FLutter能识别到List Widget子元素的改变，就需要给每个Box指定一个key。

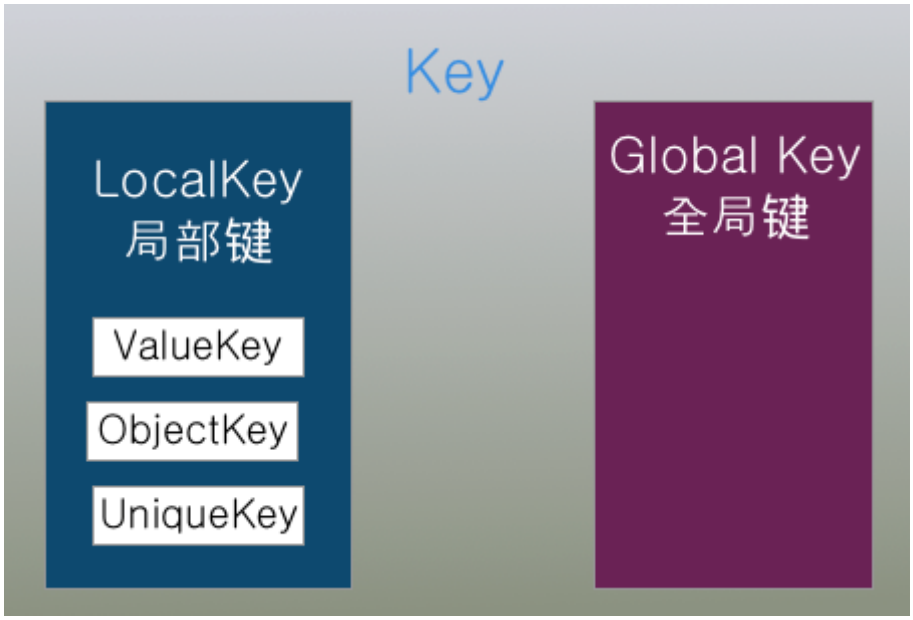
```
   List<Widget> list = [
    Box(
      color: Colors.blue,
    ),
    Box(
      color: Colors.blue,
    ),
    Box(
      color: Colors.blue,
    )
  ];
```

29.2、Flutter key：LocalKey、GlobalKey

在Flutter中，**Key是不能重复使用的**，所以Key一般用来做唯一标识。组件在更新的时候，其状态的保存主要是通过判断**组件的类型**或者**key值**是否一致。因此，当各组件的类型不同的时候，类型已经足够用来区分不同的组件了，此时我们可以不必使用key。但是如果同时存在多个同一类型的控件的时候，此时类型已经无法作为区分的条件了，我们就需要使用到key。

Flutter key子类包含`LocalKey`和`GlobalKey`。

- 局部键（LocalKey）：ValueKey、ObjectKey、UniqueKey
- 全局键（GlobalKey）： GlobalKey、GlobalObjectKey



`ValueKey`（值key）把一个值作为key，UniqueKey（唯一key）程序生成唯一的Key，当我们不知道如何指定ValueKey的时候就可以使用UniqueKey，ObjectKey（对象key）把一个对象实例作为key。

GlobalKey（全局key），GlobalObjectKey（全局Objec key，和ObjectKey有点类似）

29.1.1 LocalKey改造上面的例子

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
```

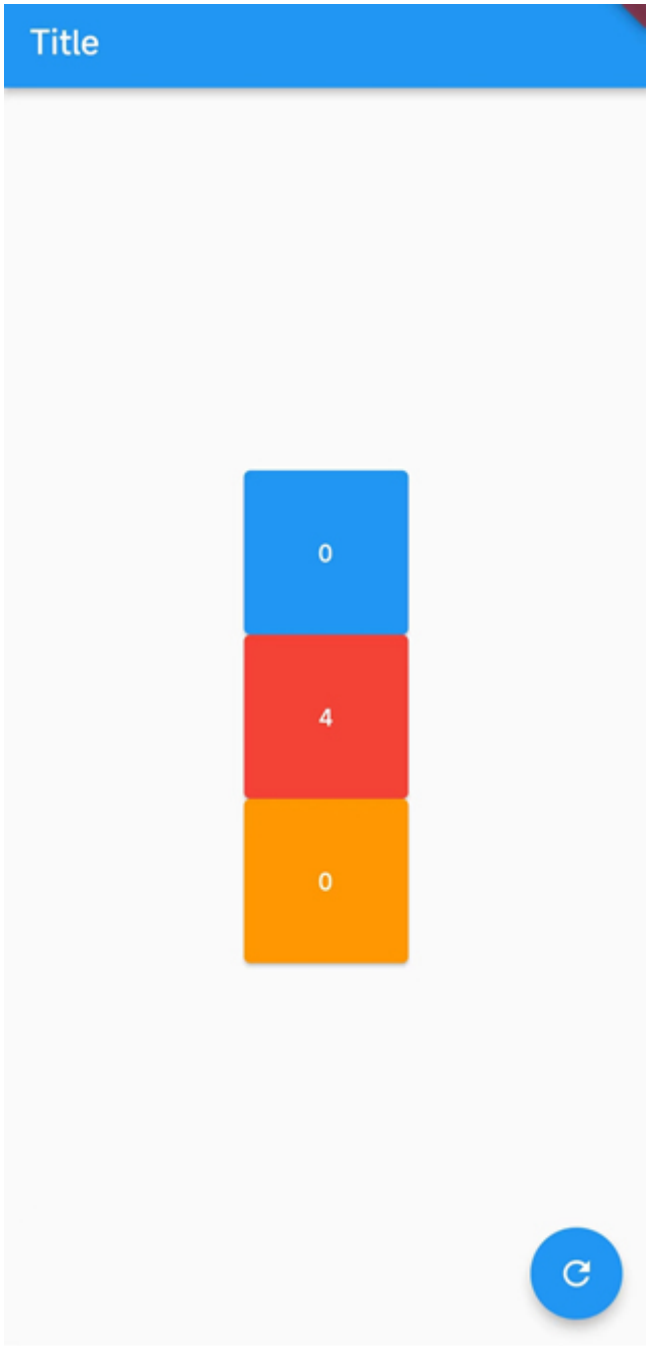
```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  List<Widget> list = [
    Box(
      key: const ValueKey(1),
      color: Colors.blue,
    ),
    Box(
      key: ObjectKey(Box(color: Colors.red)),
      color: Colors.red,
    ),
    Box(
      key:UniqueKey(), //程序自动生成一个key
      color: Colors.orange,
    )
  ];
  @override
  Widget build(BuildContext context) {

    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(() {
            list.shuffle();  //打乱list的顺序
          });
        },
        child: const Icon(Icons.refresh),
      ),
      appBar: AppBar(
        title: const Text('Title'),
      ),
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: list,
        ),
      ),
    );
  }
}
```

```
class Box extends StatefulWidget {
  Color color;
  Box({super.key, required this.color});

  @override
  State<Box> createState() => _BoxState();
}

class _BoxState extends State<Box> {
  int _count = 0;
  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 100,
      width: 100,
      child: ElevatedButton(
        style: ButtonStyle(
            backgroundColor: MaterialStateProperty.all(widget.color)),
        onPressed: () {
          setState(() {
            _count++;
          });
        },
        child: Center(
          child: Text("$_count"),
        ),
      ),
    );
  }
}
```

29.1.2 GlobalKey的使用

如果把LocalKey比作局部变量， GlobalKey就类似于全局变量

下面使用了LocalKey，当屏幕状态改变的时候把 Colum换成了Row，Box的状态就会丢失。




```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
```

```
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  List<Widget> list = [
    Box(
      key: const ValueKey(1),
      color: Colors.blue,
    ),
    Box(
      key: ObjectKey(Box(color: Colors.red)),
      color: Colors.red,
    ),
    Box(
      key:UniqueKey(), //程序自动生成一个key
      color: Colors.orange,
    )
  ];
  @override
  Widget build(BuildContext context) {
    print(MediaQuery.of(context).orientation);
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(() {
            list.shuffle();  //打乱list的顺序
          });
        },
        child: const Icon(Icons.refresh),
      ),
      appBar: AppBar(
        title: const Text('Title'),
      ),
      body: Center(
        child: MediaQuery.of(context).orientation==Orientation.portrait?Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: list,
        ):Row(
          mainAxisAlignment: MainAxisAlignment.center,
          children: list,
        ),
      ),
    );
  }
}
```

```
class Box extends StatefulWidget {
  Color color;
  Box({super.key, required this.color});

  @override
  State<Box> createState() => _BoxState();
}

class _BoxState extends State<Box> {
  int _count = 0;
  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 100,
      width: 100,
      child: ElevatedButton(
        style: ButtonStyle(
            backgroundColor: MaterialStateProperty.all(widget.color)),
        onPressed: () {
          setState(() {
            _count++;
          });
        },
        child: Center(
          child: Text("$_count"),
        ),
      ),
    );
  }
}
```

在前面我们介绍过一个Widget状态的保存主要是通过判断**组件的类型**或者**key值**是否一致。LocalKey只在当前的组件树有效，所以把Colum换成了Row的时候Widget的状态就丢失了。为了解决这个问题我们就可以使用GlobalKey。

GlobalKey优化，把LocalKey换成GlobalKey，如下：

```
import 'package:flutter/material.dart';

void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
```

```
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const MyHomePage(),
    );
  }
}

class MyHomePage extends StatefulWidget {
  const MyHomePage({super.key});

  @override
  State<MyHomePage> createState() => _MyHomePageState();
}

class _MyHomePageState extends State<MyHomePage> {
  List<Widget> list = [];
  final GlobalKey _key1 = GlobalKey();
  final GlobalKey _key2 = GlobalKey();
  final GlobalKey _key3 = GlobalKey();
  @override
  void initState() {
    // TODO: implement initState
    super.initState();
    list = [
      Box(
        key: _key1,
        color: Colors.blue,
      ),
      Box(
        key: _key2,
        color: Colors.red,
      ),
      Box(
        key: _key3, //程序自动生成一个key
        color: Colors.orange,
      )
    ];
  }

  @override
  Widget build(BuildContext context) {
    print(MediaQuery.of(context).orientation);
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          setState(() {
            list.shuffle(); //打乱list的顺序
          });
        },
        child: const Icon(Icons.refresh),
      ),
      appBar: AppBar(
        title: const Text('Title'),
      ),
      body: Center(
```

```
        child: MediaQuery.of(context).orientation == Orientation.portrait
            ? Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: list,
              )
            : Row(
                mainAxisAlignment: MainAxisAlignment.center,
                children: list,
              ),
      ),
    );
  }
}

class Box extends StatefulWidget {
  Color color;
  Box({super.key, required this.color});

  @override
  State<Box> createState() => _BoxState();
}

class _BoxState extends State<Box> {
  int _count = 0;
  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 100,
      width: 100,
      child: ElevatedButton(
        style: ButtonStyle(
            backgroundColor: MaterialStateProperty.all(widget.color)),
        onPressed: () {
          setState(() {
            _count++;
          });
        },
        child: Center(
          child: Text("$_count"),
        ),
      ),
    );
  }
}
```

29.3、 GlobalKey 获取子组件

*globalKey.currentState 可以获取子组件的状态，执行子组件的方法，*globalKey.currentWidget可以获取子组件的属性，_globalKey.currentContext!.findRenderObject()可以获取渲染的属性。

####

```
import 'package:flutter/material.dart';
```

```
void main() {
  runApp(const MyApp());
}

class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  // This widget is the root of your application.
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'Flutter Demo',
      theme: ThemeData(
        primarySwatch: Colors.blue,
      ),
      home: const HomePage(),
    );
  }
}

class HomePage extends StatefulWidget {
  const HomePage({super.key});

  @override
  State<HomePage> createState() => _HomePageState();
}

class _HomePageState extends State<HomePage> {
  final GlobalKey _globalKey = GlobalKey();
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        child: const Icon(Icons.add),
        onPressed: (){
          //1、获取子组件的状态 调用子组件的属性
          var state=(_globalKey.currentState as _BoxState);
          setState(() {
            state._count++;
          });

          //2、获取子组件的属性
           var box=(_globalKey.currentWidget as Box);
           print(box.color);

           //3、获取子组件渲染的属性
          var renderBox= (_globalKey.currentContext!.findRenderObject() as
RenderBox);
          print(renderBox.size);

        },
      ),
      appBar: AppBar(
        title: const Text('Title'),
      ),
      body: Center(
        child: Box(
          key: _globalKey,
```

```
          color: Colors.red,
        ),
      ),
    );
  }
}


class Box extends StatefulWidget {
  final Color color;
  const Box({Key? key, required this.color}):super(key: key);

  @override
  State<Box> createState() => _BoxState();
}

class _BoxState extends State<Box> {
  int _count = 0;
  run(){
    print("run");
  }
  @override
  Widget build(BuildContext context) {
    return SizedBox(
      height: 100,
      width: 100,
      child: ElevatedButton(
        style: ButtonStyle(
            backgroundColor: MaterialStateProperty.all(widget.color)),
        onPressed: () {
          setState(() {
            _count++;
          });
        },
        child: Center(
          child: Text("$_count"),
        ),
      ),
    );
  }
}
```

29.4、 Widget Tree、Element Tree 和 RenderObject Tree

Flutter应用是由是Widget Tree、Element Tree 和 RenderObject Tree组成

Widget可以理解成一个类，Element可以理解成Widget的实例，Widget与Element的关系可以是一对多，一份配置可以创造多个Element实例

属性	描述
Widget	Widget就是一个类， 是Element 的配置信息。与Element的关系可以是一对多，一份配置可以创造多个Element实例
Element	Widget 的实例化，内部持有Widget和RenderObject。
RenderObject	负责渲染绘制

默认情况下面，当Flutter同一个 Widget的大小，顺序变化的时候，FLutter不会改变Widget的state。

三十、AnimatedList 实现动态列表

30.1、 AnimatedList实现动画

AnimatedList 和 ListView 的功能大体相似，不同的是， AnimatedList 可以在列表中插入或删除节点时执行一个动画，在需要添加或删除列表项的场景中会提高用户体验。

AnimatedList 是一个 StatefulWidget，它对应的 State 类型为 AnimatedListState，添加和删除元素的方法位于 AnimatedListState 中：

```
void insertItem(int index, { Duration duration = _kDuration });

void removeItem(int index, AnimatedListRemovedItemBuilder builder, { Duration
duration = _kDuration }) ;
```

AnimatedList常见属性：

属性	描述
key	globalKey final globalKey = GlobalKey();
initialItemCount	子元素数量
itemBuilder	方法 (BuildContext context, int index, Animation animation) {}

关于GlobalKey： 每个`Widget`都对应一个`Element`，我们可以直接对`Widget`进行操作，但是无法直接操作`Widget`对应的`Element`。而`GlobalKey`就是那把直接访问`Element`的钥匙。通过`GlobalKey`可以获取到`Widget`对应的`Element`。

30.2、 AnimatedList增加列表FadeTransition、ScaleTransition

FadeTransition Demo

```
import 'package:flutter/material.dart';

class AnimatedListPage extends StatefulWidget {
  const AnimatedListPage({super.key});

  @override
  State<AnimatedListPage> createState() => _AnimatedListPageState();
}

class _AnimatedListPageState extends State<AnimatedListPage> {

  final globalKey = GlobalKey<AnimatedListState>();
   List<String> list = ["第一条数据","第二条数据"];
  @override
  void initState() {
    // TODO: implement initState
    super.initState();
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: (){
          list.add("这是一个数据");
          globalKey.currentState!.insertItem(list.length - 1);
        },
        child: const Icon(Icons.add),
      ),
      appBar: AppBar(
        title: const Text("AppBar组件"),
      ),
      body: AnimatedList(
          key: globalKey,
          initialItemCount: list.length,
          itemBuilder: (context, index, animation) {
            return FadeTransition(
              opacity: animation,
              child: ListTile(
                title: Text(list[index]),
                trailing: Icon(Icons.delete)
              ),
            );
          }),
    );
  }
}
```

ScaleTransition demo

```
import 'package:flutter/material.dart';

class AnimatedListPage extends StatefulWidget {
  const AnimatedListPage({super.key});
```

```
  @override
  State<AnimatedListPage> createState() => _AnimatedListPageState();
}

class _AnimatedListPageState extends State<AnimatedListPage> {

  final globalKey = GlobalKey<AnimatedListState>();
   List<String> list = ["第一条数据","第二条数据"];
  @override
  void initState() {
    // TODO: implement initState
    super.initState();
  }
  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: (){
          list.add("这是一个数据");
          globalKey.currentState!.insertItem(list.length - 1);
        },
        child: const Icon(Icons.add),
      ),
      appBar: AppBar(
        title: const Text("AppBar组件"),
      ),
      body: AnimatedList(
          key: globalKey,
          initialItemCount: list.length,
          itemBuilder: (context, index, animation) {
            print(animation);
            return ScaleTransition(
              scale: animation,
              child: ListTile(
                title: Text(list[index]),
                trailing: Icon(Icons.delete)
              ),
            );
          }),
    );
  }
}
```

30.3、 AnimatedList 删除列表

```
import 'dart:async';

import 'package:flutter/material.dart';

class AnimatedListPage extends StatefulWidget {
  const AnimatedListPage({super.key});

  @override
  State<AnimatedListPage> createState() => _AnimatedListPageState();
```

```
}

class _AnimatedListPageState extends State<AnimatedListPage> {
  final globalKey = GlobalKey<AnimatedListState>();
  bool flag = true;
  List<String> list = ["第一条数据", "第二条数据"];
  @override
  void initState() {
    // TODO: implement initState
    super.initState();
  }

  Widget _buildItem(context, index) {
    return ListTile(
        key: ValueKey(index),
        title: Text(list[index]),
        trailing: IconButton(
          icon: Icon(Icons.delete),
          // 点击时删除
          onPressed: () => _deleteItem(context, index),
        ));
  }

  _deleteItem(context, index) {
    if (flag == true) {
      flag = false;
      print(index);
      //注意：删除后需要重新setState
      setState(() {
        //  删除过程执行的是反向动画，animation.value 会从1变为0
        globalKey.currentState!.removeItem(index, (context, animation) {
          //注意先build然后再去删除
          var item = _buildItem(context, index);
          list.removeAt(index);
          return FadeTransition(
            opacity: animation,
            child: item,
          );
        },duration: Duration(milliseconds: 500));
      });
      //解决快速删除bug  重置flag
      const timeout = Duration(milliseconds: 600);
      Timer.periodic(timeout, (timer) {
        flag=true;
        timer.cancel();
      });

    }
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          // 增加 animation.value 会从0变为1
          list.add("这是一个数据");
          globalKey.currentState!.insertItem(list.length - 1);
```

```
        },
        child: const Icon(Icons.add),
      ),
      appBar: AppBar(
        title: const Text("AppBar组件"),
      ),
      body: AnimatedList(
          key: globalKey,
          initialItemCount: list.length,
          itemBuilder: (context, index, animation) {
            return FadeTransition(
              opacity: animation,
              child: _buildItem(context, index),
            );
          }),
    );
  }
}
```