

flutter 学习~预习

1. 课前准备

1. [Dart编程语言中文网](#)
2. [Dart安装](#)
3. 开发工具 Flutter开发软件: [Visual Studio Code](#)、[Android Studio](#) 移动端开发软件: [Xcode](#)、[Android Studio](#)

2. 课堂目标

- 环境搭建
- Dart语言及其核心库介绍

3. 知识点

1. 开发工具介绍

- 电脑选择:
 - Window: 无法进行iOS开发
 - Mac两端都可以开发

如果需要两端开发需要使用Mac, 但是window电脑不影响Flutter的学习
- 开发工具的选择:
 - VSCode: 前端开发方便, 支持Flutter开发, 但是无法调试安卓
 - Android Studio: 安卓开发必备, 同样是支持Flutter开发 (官方推荐)

2. 环境搭建

- Mac电脑环境安装:
 - 硬件环境: MacOS (64-bit)、磁盘空间700MB (不包括Xcode或Android Studio的磁盘空间)、Flutter 依赖工具 `bash`, `mkdir`, `rm`, `git`, `curl`, `unzip`, `which`
 - Step1: 使用镜像 (因为网络限制需要设置)

```
//添加到.bash_profile中, 注意bash_profile是隐藏文件
export PUB_HOSTED_URL=https://pub.flutter-io.cn
export FLUTTER_STORAGE_BASE_URL=https://storage.flutter-io.cn
```

注意: 此镜像为临时镜像, 并不能保证一直可用, 参考详情请参考 [Using Flutter in China](#) 以获得有关镜像服务器的最新动态

- Step2: 配置Flutter SDK
 - 前往官网获取SDK, [这是传送门](#)
 - 解压安装包到想安装的目录(这个目录以后不需要动, 故不建议放在deskTop)

```
cd ~/development
unzip ~/Downloads/flutter_macos_v0.5.1-beta.zip
```

- 添加 flutter 相关工具到path中

```
export PATH=`自己存放flutter的路径`/flutter/bin:$PATH
```

- 运行 flutter doctor 查看安装结果如何（注：如果是想在iOS上运行，则不需要安卓相关也是可以的，反之一样）
- Step3: iOS开发环境安装
 - 下载Xcode，可以去AppStore安装最新版本Xcode，注意电脑空间大小，Xcode工具在8G左右
 - 安装cocoapods环境
- Step3: 安卓开发环境安装
 - 下载Android Studio <https://developer.android.google.cn/studio>
 - 设置安卓模拟器：启动 **Android Studio>Tools>Android>AVD Manager** 并选择 **Create Virtual Device**
 - **Perference>Plugins** 安装插件Dart、Flutter，安装完成后重启即可
- Window电脑环境安装：
 - 硬件要求：Windows 7 或更高版本 (64-bit)、磁盘空间400 MB (不包括Android Studio的磁盘空间)
 - Step1: 使用镜像（因为网络限制需要设置）

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

注意：此镜像为临时镜像，并不能保证一直可用，参考详情请参考 [Using Flutter in China](#) 以获得有关镜像服务器的最新动态。

- Step2: 配置Flutter SDK
 - 前往官网获取SDK，[这是传送门](#)
 - 将安装包zip解压到你安装Flutter SDK的路径（如：C:\src\flutter；注意，不要将flutter安装到需要一些高权限的路径如 C:\Program Files\）
 - 在Flutter安装目录的 flutter 文件下找到 flutter_console.bat，双击运行并启动 flutter 命令行，接下来，就可以在Flutter命令行运行flutter命令了
 - 更新环境变量：
 - 转到“控制面板>用户帐户>用户帐户>更改我的环境变量”
 - 在“用户变量”下检查是否有名为“Path”的条目：
 - 如果该条目存在，追加 flutter\bin 的全路径，使用 ; 作为分隔符
 - 如果条目不存在，创建一个新用户变量 Path，然后将 flutter\bin 的全路径作为它的值

- 在“用户变量”下检查是否有名为“PUB_HOSTED_URL”和“FLUTTER_STORAGE_BASE_URL”的条目，如果没有，也添加它们。

- 重启电脑

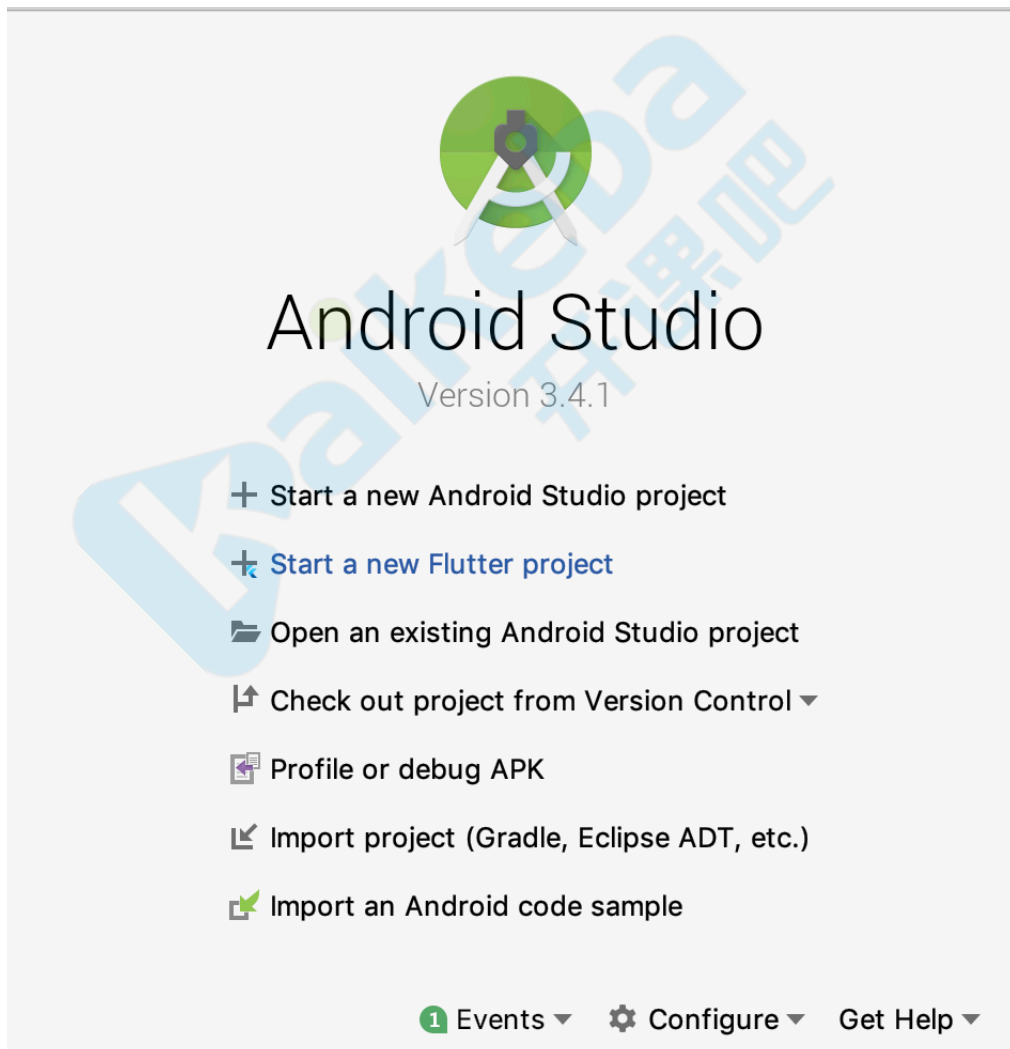
- 安装Android Studio（版本>=3.0） <https://developer.android.google.cn/studio>
- 安装Android Studio的Flutter和Dart插件（**File>Settings>Plugins**），安装完成后重启即可

3. 第一个flutter程序

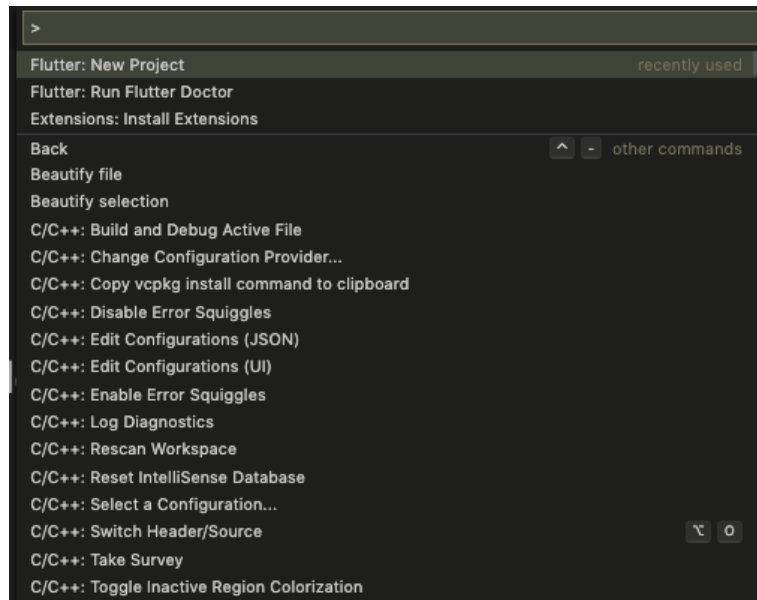
- VS需要添加 `flutter` 和 `dart code` 两个插件，安装完成重启编译器即可
- 创建flutter项目

- 方式一：命令行 `flutter create flutterProject`
- 方式二：Android Studio

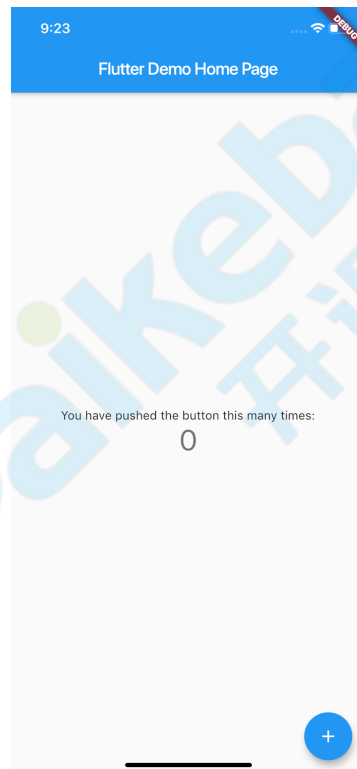
File>New Flutter Project



- 方式三：VSCode
 - ① **View>Command Palette...**
 - ② 输入 flutter, 然后选择 **Flutter: New Project**



- 如果正常，运行效果如下：



4. Dart语言及其核心库介绍[Dart语言官网链接](#)

- 变量与常量

```
//定义变量
var name = 'Jack';
name = 'Rose';
name = 20; //A value of type 'int' can't be assigned to a variable of
type 'String'.
print(name); //Rose

var age;
print(age); //null

int height = 12;
print(height); //12
```

如果对象不限于单一类型（没有明确的类型），使用Object或dynamic关键字

```
Object value1 = 'Lucy';
value1 = 22;
print(value1);

dynamic value2 = '12';
value2 = 'LiLei';
print(value2);
```

如果定义的变量不会发生变化，那么使用 final 或 const，不是 var，也不是一个类型。一个 final 变量只能被初始化一次；const 变量是一个编译时常量，（const 变量是隐式的 final）

```
//定义常量
final userName = 'Jack';
//userName = 'Rose'; //'userName', a final variable, can only be set
once.

const pi = 3.14;
//pi = 3.4; //Constant variables can't be assigned a value.
```

注意：

- (1) 被final修饰的顶级变量或类变量在第一次声明的时候就需要初始化；
- (2) 被final或者const修饰的变量，变量类型可以省略，建议指定数据类型；
- (3) final 或者 const 不能和 var 同时使用

● 数据类型

Dart语言中常用的数据类型包括：Number、String、Boolean、List、Map。

- Number类型包含两个子类：int（ $-2^{53} \sim 2^{53}$ ）、double（双精度浮点型）
- String类型即字符串类型
- Boolean类型（注意：有的语言中0是false、大于0是true，在Dart中是值必须是true或者false）
- List类型即具有一系列相同类型的数据，类型JavaScript中的数组Array对象

- Map类型将键和值相关联的对象。键和值都可以是任何类型的对象，每个键只出现一次，但您可以多次使用相同的值
- 函数

Dart是一个面向对象的语言，所以函数也是对象，属于Function对象。函数也可以像传参一个传递给其他函数。

```
String getUserInfo(String name, [int height]) {  
  if(height != null) {  
    return name + '~height:$height';  
  }  
  return name;  
}
```

注意：

上述getUserInfo函数中height参数就是可选的

- 如果需要给参数设置默认值，则

```
String getUserInfo(String name, [int height = 170]) {  
  if(height != null) {  
    return name + '~height:$height';  
  }  
  return name;  
}
```

- 匿名函数

```
([[Type] param1[, ...]]) {  
  codeBlock;  
};  
  
//eg:  
//(context) => HomePage(widget.isDark, themeChanger)  
//等价于：  
//func_name(context){return HomePage(widget.isDark, themeChanger);}
```

- 级联符号

```
final addressBook = (AddressBookBuilder()  
  ..name = 'jenny'  
  ..email = 'jenny@example.com'  
  ..phone = (PhoneNumberBuilder()  
    ..number = '415-555-0100'  
    ..label = 'home')  
  .build())  
  .build();
```

- 运算符

描述	运算符
一元后缀	expr++ expr-- () [] . ?.
一元前缀	-expr !expr ~expr ++expr --expr
乘除	* / % ~/
加减	+-
位移	<< >>
按位与	&
按位异或	^
按位或	
关系和类型测试	>= > <= < as is is!
等于	== !=
逻辑与	&&
逻辑或	
条件	expr1?expr2:expr3
级联	..
赋值	= *= /= ~/= %= += -= <<= >>= &= ^= ??=

- 流程控制语句

- 条件语句：if、if...elseif、if...elseif...else

```
int score = 90;
if (score >=90) {
    print('优秀');
} else if (80>=score && score<90) {
    print('良');
} else if (60>=score && score<80) {
    print('及格');
} else {
    print('不及格');
}
```

- 循环语句：for、forin

```

var list = [1,2,3];
for(var i=0; i<list.length; i++) {
    print(list[i]);
}
//使用forin循环
for (var item in list) {
    print(item);
}

```

- 循环语句：while循环、do...while循环

```

//while 先判断条件在执行
int i = 0;
while (i < 2) {
    print(i++);
};

//do...while 先执行一次在判断条件，至少执行一次
do {
    print(i--);
} while (i > 0 && i < 3);

```

- break(终止离break最近的循环，只能终止一层循环)，continue(跳出当前循环，只能跳出一层循环)

```

//当i==2时候已经终止循环了
for (var i = 0; i < 4; i++) {
    if (i == 2) {
        break;
    }
    print(i);
}

//break终止的只是当前循环，只能终止一层循环
for (var i = 0; i < 2; i++) {
    for (var j = 0; j < 4; j++) {
        if (j == 1) {
            break;
        }
        print(['$i,$j']);
    }
};

```

```

//当i==2时候已经跳出循环,执行下一次循环
for (var i = 0; i < 4; i++) {
    if (i == 2) {
        continue;
    }
}

```



```

    }
    print(i);
}

//continue跳出的只是当前循环，只能跳出一层循环
for (var i = 0; i < 2; i++) {
    for (var j = 0; j < 4; j++) {
        if (j == 1) {
            continue;
        }
        print(['$i,$j']);
    };
};

```

o switch...case语句

```

//language 也可以是num、String、编译期常量、对象、枚举这几种类型
String language = 'java';
switch(language) {
    case 'dart':
        print('dart language');
        break;
    case 'java':
        print('java language');
        break;
    case 'python':
        print('python language');
        break;
    default:
        print('none');
} //java language

/*
    用continue跳转到位置执行最近的case
    TopTest名字可以随便起
*/
switch(language) {

    case 'dart':
        print('dart language');
        break;
    case 'java':
        print('java language');
        continue TopTest
        //break;
    case 'python':
        print('python language');
        break;
    TopTest:

```

```

        default:
            print('none');
    }
    /*
        java language
        none
    */

```

- 比较类型可以有num、String、编译期常量、对象、枚举
- 空case必须有一个默认情况
- default 处理默认情况
- continue 跳转标签

○ assert断言

```

//确保这个变量不为空值.
assert(text != null);

//确保这个变量小于100.
assert(number < 100);

//确保它是一个https协议类型的URL.
assert(urlString.startsWith('https'));

```

- 如果一个布尔条件值为false，使用assert语句来中断正常执行的代码
- 只能在调试模式下使用，在生产模式下没有任何作用。

● 异常处理

```

//(1)抛出异常
throw new FormatException('抛出一个FormatException异常');
//你也可以抛出任意对象
throw '数据非法!';

//(2)捕获异常
try{
    //逻辑代码操作
} on Exception catch (e){
    print('exception details:\n $e');
} catch (e,s){
    print('exception details:\n $e');
    print('stack trace:\n $s');
}

//(3)finally
try{
    //逻辑代码操作
} on Exception catch (e){
    print('exception details:\n $e');
}

```

```

}catch (e,s){
    print('exception details:\n $e');
    print('stack trace:\n $s');
}finally{
    print('Do sth');
}

```

- 面向对象

- 类的声明，使用class关键字定义类。

```

class Person {
    //类成员
    String name;
    int age;
    //final修饰的属性，不能被外部重新赋值，只可读，不可写
    final String address = null;

    void work() {
        print('Name is $name, Age is $age');
    }
    //dart默认都是公开的，在变量名或方法名前加入_前缀即为私有
    void _work() {
        print('Name is $_name, Age is $age');
    }
}

```

- 创建对象，使用new关键字或不写new关键字

```

var person = Person();
//var person = new Person();
person.name = 'Tom';
person.age = 20;
print(person.name);
person.work();

```

- 构造函数：即用来构造当前类的函数，函数名必须和类名相同

```

class Person {
    //类成员
    String name;
    int age;

    // (1) 默认无参构造方法
    Person() {
    }
    // (2) 常规的构造函数
    Person(this.name, this.age);
}

```

```

// (3) 命名的构造函数
Person.withName(String name) {
    this.name = name;
}

void work() {
    print('Name is $name, Age is $age');
}
}

```

常量构造方法：当类的属性设置一次之后，就不会再设置了，那么这个类就可以声明为常量类，常量类的属性使用final修饰，而构造方法使用const修饰。

```

class Person {
    final String name;
    final int age;
    final String gender;

    const Person(this.name, this.age, this.gender);

    void work() {
        print('Name is $name, Age is $age, Gender is $gender, He is
working');
    }
}

void main() {
    //如果需要将对象作为常量，就需要将构造方法声明为常量构造方法
    //使用常量构造方法的对象，属性和构造方法都为常量，属性使用final修饰，构造方法使用const修饰
    //常量型对象运行时更快，如果对象的值设置一次后就不会改变，可以使用这种方式
    const person = const Person('Tom', 18, 'Male');
    person.work();
}

```

- 读取和写入对象：每个类的实例系统都会隐式的包括set和get方法

```

class Rectangle {
  num width;
  num height;
  num left;
  num top;

  Rectangle(this.width, this.height, this.left, this.top);

  //获取right值
  num get right => left+width;
  //设置right值的同时left也需要更改
  set right(num value) => left = value - width;
}

```

- operator重运算符重载操作

```

Class Vector {
  final int x;
  final int y;
  const Vector(this.x,this.y);

  //重载加号 + (a+b)
  Vector operator + (Vector v){
    return new Vector(x + v.x,y + v.y);
  }
}

main() {
  //实例化两个变量
  final result1 = new Vector(10,20);
  final result2 = new Vector(30,40);

  final result = result1 + result2;

  print('result.x = '+result.x.toString()+', '+result.y =
'+result.y.toString());
  //打印结果
  //result.x = 40,result.y = 60
}

```

- 继承类：继承它允许创建分等级层次的类。继承就是子类继承父类的特征和行为，使得子类对象具有父类的实例域和方法；或子类从父类继承方法，使得子类具有父类相同的行为。Dart里面使用extends关键字来实现继承，super关键字来指定父类。

```

Class Animal {
  void eat(){
    print('动物会吃');
  }
}

```

```

        void run(){
            print('动物会跑');
        }
    }

    Class Human extends Animal {
        void say(){
            print('人会说');
        }

        void study(){
            print('人会学习');
        }
    }

    main(){
        var animal = new Animal();
        animal.eat();
        animal.run();

        value human = new Human();
        human.eat();
        human.run();
        human.say();
        human.study();

        //打印结果
        //动物会吃
        //动物会跑

        //动物会吃
        //动物会跑
        //人会说
        //人会学习
    }

```

- 抽象类：不具体实现方法，只是写好定义接口，具体实现留着调用的人去实现。抽象类可以使用abstract关键字定义类

```

abstract class Animal{
    eat();    //抽象方法
    run();   //抽象方法
    printInfo(){
        print('我是一个抽象类里面的普通方法');
    }
}

class Dog extends Animal{

```

```

@Override
eat() {
    print('小狗在吃骨头');
}

@Override
run() {
    // TODO: implement run
    print('小狗在跑');
}
}

class Cat extends Animal{
    @Override
    eat() {
        // TODO: implement eat
        print('小猫在吃老鼠');
    }

    @Override
    run() {
        // TODO: implement run
        print('小猫在跑');
    }
}

main(){
    Dog d=new Dog();
    d.eat();
    d.printInfo();

    Cat c=new Cat();
    c.eat();
    c.printInfo();

    // Animal a=new Animal();    //抽象类没法直接被实例化
}

```

- Mixins: (混入功能) 相当于多继承, 也就是说可以继承多个类, 使用with关键字来实现 Mixins的功能

```

Class First {
    void printSth(){
        print('im first printSth');
    };
}

Class Second {
    void printSth(){

```

```

        print('im Second printSth');
    };

    void secondPrint(){
        print('test');
    }
}

Class A = Second with First;

main (){
    A a = new A();
    a.printSth();
    a.secondPrint();

    //打印结果
    //im first printSth
    //test
}

```

- 库的使用

通过import语句在一个库中引用另一个库的文件，需要注意一下事项：

- 在import语句后面需要接上库文件的路径
- 对Dart语言提供的库文件使用dart:xx格式
- 第三方的库文件使用package:xx格式

```

import 'dart:io';
import 'package:mylib/mylib.dart';

```

- 指定一个库的前缀

```

import 'package:lib1/lib1.dart';
import 'package:lib2/lib2.dart' as lib2;

Element emelent1 = new Element(); //默认使用lib1里面的Element
lib2.Element emelent2 = new lib2.Element(); //使用lib2里面的Element

```

注意：lib1/lib1.dart及lib2/lib2.dart里面都有Element类，如果直接引用就不知道具体引用哪个Element类，所以代码中把lib2/lib2.dart指定成lib2，这样使用lib2.Element就不会发生冲突。

- show关键字：只引用一点
- hide关键字：除此之外都引用


```
//导入foo
import 'package:lib1/lib1.dart' show foo;

//除了foo导入其他所有内容
import 'package:lib1/lib1.dart' hide foo;
```

- 延迟(deferred)加载（也称为延迟(lazy)加载）允许应用程序按需加载库

目的：

- (1) 减少应用程序的初始启动时间；
- (2) 执行A / B测试-尝试的算法的替代实施方式中；
- (3) 加载很少使用的功能

```
import 'package:deferred/hello.dart' deferred as hello;
```

- [异步支持](#)

Dart是一个单线程编程语言。如果任何代码阻塞线程执行都会导致程序卡死。异步编程防止出现阻塞操作。Dart使用Future对象表示异步操作。

Future表示在将来某时获取一个值的方式。当一个返回Future的函数被调用的时候，做了两件事情：

- 函数把自己放入队列和返回一个未完成的Future对象
- 之后当值可用时，Future带着值变成完成状态。

获得Future的方式：（1）使用async和await；（2）使用Future的接口

- async和await：async和await关键字是Dart异步支持的一部分。允许像写同步代码一样写异步代码和不需要使用Future接口。

在Dart2中有轻微的改动。async函数执行时，不是立即挂起，而是要执行到函数内部的第一个await。

- 元数据：又称中介数据、中继数据，为描述数据的数据（data about data），主要是描述数据属性（property）的信息，用来支持如指示存储位置、历史数据、资源查找、文件记录等功能

- (1) 使用元数据给代码添加更多的信息
- (2) 元数据是以@开始的修饰符，在@后面接着编译时的常量或调用一个常量构造函数
- (3) 目前提供的修饰符（@deprecated、@override、@proxy、@required）

4. 总结

难点是环境安装，重点是dart语法。

5. 作业 && 答疑

练习今天学习的内容

6. 下节课内容

- App结构和导航组件
 - MaterialApp应用组件
 - Scaffold脚手架组件
 - Flutter主题
- 常用组件的使用
 - Image
 - Text
 - 按钮相关组件 (MaterialButton、RaiseButton、FlatButton、DropDownButton、悬浮按钮、IconButton)
 - 单组件容器布局组件 (Container、Padding、Center、Align)
 - 多组件容器布局组件 (Row、Column、Stack、Wrap)

