**什麼是Flutter**

一個能建立原生跨平台的框架，用一種**程式語言(dart)寫一份Code**能同時讓**iOS**與**Android**平台使用，也支援Web以及桌面應用程式。Flutter因擁有豐富且方便的元件(widget)以及優秀的性能，目前正在快速發展。只需要學習一種程式語言就能夠寫兩個平台的App，買一送一，好Flutter不學嗎?

Flutter的SDK(Software Development Kit)會將我們寫的程式碼編譯成native machine code，轉換成Android或iOS讀得懂的code。Flutter內建了許多常用的UI元件(widget)，可以用像堆積木的方式建立你的介面。

**什麼是Dart**

Dart是Google開發的程式語言，主要針對前端介面開發所使用的程式語言(mobile apps, web)

Dart屬於物件導向語言，語法結合了javascript、java、c#的特性，基本上只要你有程式基礎要上手是非常容易的。

**Flutter跟Dart的關係**

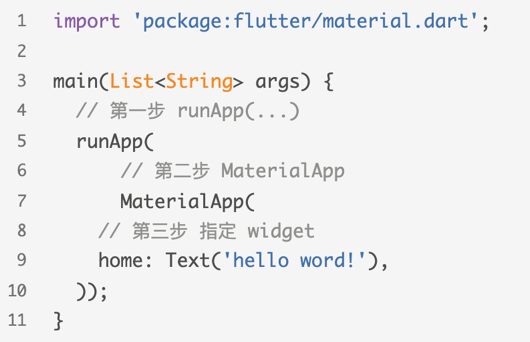
Flutter是一個框架(內含許多專用的Library、widget)，而Dart則是程式語言。 有點像Python與Django的關係

**建立Flutter專案**command + p -> new flutter project  
choose a folder  
name it

**第一個程式- Hello World**

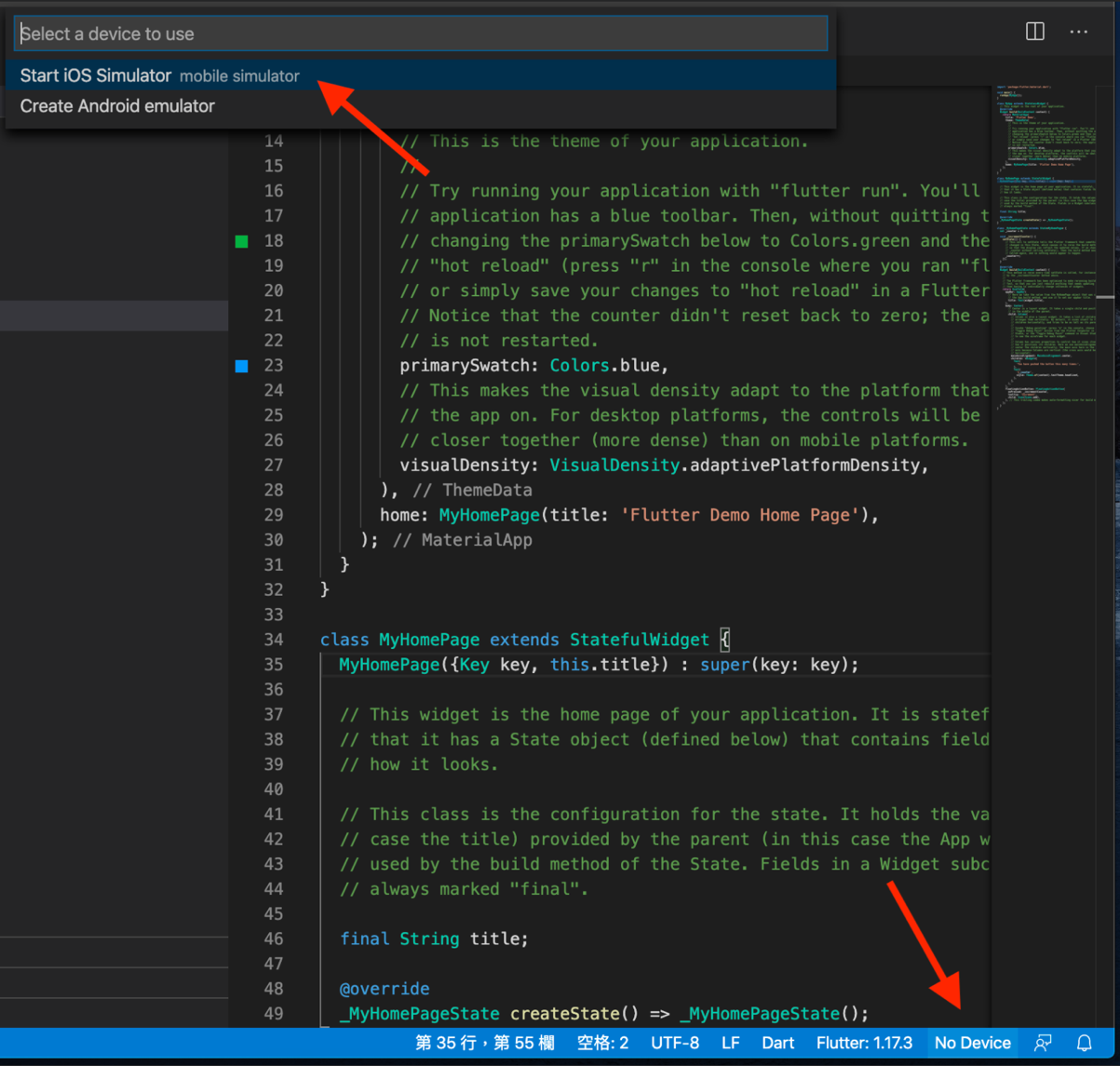
1. 第一步 runApp(...)  
2. 第二步 MaterialApp(...)  
3. 第三步 指定 widget Text(...)

code



|  |  |
| --- | --- |
|  |  |

**開啟編譯器**

****

**切換圖片, stateless, stateful**

載入資源文件：pubspec.yaml -> # To add assets to your application -> 改一下模板  
 assets:

- assets/

顯示圖片 -> stateless -> 組件：statelessW

import 'package:flutter/material.dart';

class ImageShow extends StatelessWidget {

final String imageName;

const ImageShow({Key? key, required this.imageName}) : super(key: key);

@override

Widget build(BuildContext context) {

return Image.asset('assets/$imageName');

}

Main -> mateapp

import 'package:flutter/material.dart';

import 'package:state\_less\_ful/image\_show.dart';

void main() => runApp(MyApp());

class MyApp extends StatefulWidget {

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

@override

MyAppState createState() => MyAppState();

}

class MyAppState extends State<MyApp> {

String imageFileName = 'pika1.JPG';

@override

Widget build(BuildContext context) {

return MaterialApp(

title: 'Material App',

home: Scaffold(

appBar: AppBar(

title: Text('Material App Bar'),

),

body: Center(

child: Column(

children: <Widget>[

//圖片顯示 less widget

ImageShow(

imageName: imageFileName,

),

//按鈕

RaisedButton(

onPressed: (){

if (imageFileName == 'pika1.JPG'){

imageFileName = 'pika2.JPG';

}else{

imageFileName = 'pika1.JPG';

}

setState(() {

imageFileName = imageFileName;

});

},

child: Text('Change Photo'),

),

],

)),

),

);

}

}

**基礎組件**

<Text>

1. Text

| 文字顯示樣式和屬性 this.style， | 文字對齊方式this.textAlign |
| --- | --- |
| 文字顯示方向 this.textDirection | 設置語言環境 this.locale |
| 是否自動換行 this.softWrap | 文字溢出後處理方式 this.overflow |
| 字體縮放 this.textScaleFactor | 最大顯示行數 this.maxLines |
| 圖像的語義描述，用於向Andoid上的TalkBack和iOS上的VoiceOver提供圖像描述 this.semanticsLabel | |

1. TextStyle

| 是否繼承父類元件屬性 this.inherit = true | 字體顏色 this.color |
| --- | --- |
| 文字大小，預設14px this.fontSize | 字體粗細 this.fontWeight |
| 字體樣式，normal或italic this.fontStyle | 字體基線 this.textBaseline |
| 字母間距，預設為0，負數間距縮小，正數間距增大 this.letterSpacing | 單詞間距，預設為0，負數間距縮小，正數間距增大 this.wordSpacing |
| 行高 this.height | 設置區域 this.locale |
| 前景色 this.foreground | 背景色 this.background |
| 陰影 this.shadows | 文字劃線，下換線等等裝飾 this.decoration |
| 劃線顏色 this.decorationColor | 劃線樣式，虛線、實線等樣式 this.decorationStyle |
| 描述資訊 this.debugLabel | 字體 String fontFamily,  List<String> fontFamilyFallback, String package |

<Photo>

1. code

| // assets  Text('assets'),  Image.asset(\_assetImg), | // 網路讀取  Text('網路讀取'),  Image.network(\_imgUrl), |
| --- | --- |
| // NetworkImage  Text('NetworkImage'),  Image(image: NetworkImage(\_imgUrl)), | Text('佔位圖'),  FadeInImage(  fadeInCurve: Curves.bounceIn,  placeholder: AssetImage(\_assetImg),  image: NetworkImage(\_imgUrl)), |
| // 原型頭像  Text('原型頭像'),  CircleAvatar(  backgroundColor: Colors.brown.shade800,  child: Text('圆角图片'),  backgroundImage: AssetImage(\_assetHeaderImg),  radius: 50.0), | // 圖標  Text('圖標'),  ImageIcon(  NetworkImage(\_imgUrl),  size: 100,  ), |
| // ClipRRect 圓角  Text('ClipRRect 圓角'),  ClipRRect(  child: Image.network(\_imgUrl),  borderRadius: BorderRadius.all(Radius.circular(20)),  ), | // 圓角矩形框  Text('圓角矩形框'),  Container(  width: 200,  height: 80,  decoration: BoxDecoration(  shape: BoxShape.rectangle,  borderRadius: BorderRadius.circular(10.0),  image: DecorationImage(  image: NetworkImage(\_imgUrl), fit: BoxFit.cover),  ),  ), |
| // 橢圓圖  Text('橢圓圖'),  ClipOval(  child: Image.network(  \_imgUrl,  scale: 8.5,  ),  ), | // 混色  Text('混色'),  Image.asset(  \_assetHeaderImg,  color: Colors.amber,  colorBlendMode: BlendMode.dstATop,  ), |
| // 裁剪  Text('裁剪'),  Image.asset(  \_assetImg,  width: 400,  height: 50,  fit: BoxFit.cover,  ), |  |

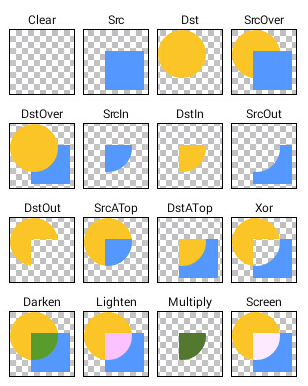
1. 缩放

enum BoxFit

| **名稱** | **説明** |
| --- | --- |
| fill | 圖片按照指定的大小在 Image 中顯示，拉伸顯示圖片，不保持原比例，填滿 Image。 |
| contain | 以原圖正常顯示為目的，如果原圖大小大於 Image 的 size，就按照比例縮小原圖的寬高，居中顯示在 Image 中。如果原圖 size 小於 Image 的 size，則按比例拉升原圖的寬和高，填充 Image 一邊並居中顯示。 |
| cover | 以原圖填滿 Image 為目的，如果原圖 size 大於 Image 的 size，按比例縮小，居中顯示在 Image 上。如果原圖 size 小於 Image 的 size，則按比例拉升原圖的寬和高，填充 Image 居中顯示。 |
| fitWidth | 以原圖正常顯示為目的，如果原圖寬大小大於（小於）Image 的寬，就縮小（放大）原圖的寬與 Image 一致，居中顯示在 Image 中。 |
| fitHeight | 以原圖正常顯示為目的，如果原圖高大小大於（小於）Image 的高，就縮小（放大）原圖的高與 Image 一致，居中顯示在 Image 中。 |
| none | 保持原圖的大小，顯示在 Image 的中心。當原圖的 size 大於 Image 的 size 時，多出來的部分被截掉。 |
| scaleDown | 以原圖正常顯示為目的，如果原圖大小大於 Image 的 size，就按照比例縮小原圖的寬高，居中顯示在 Image 中。如果原圖 size 小於 Image 的 size，則不做處理居中顯示圖片。 |

1. 混合模式（29種）

enum BlendMode {  
 clear,src,dst,srcOver,dstOver,srcIn,dstIn,srcOut,dstOut,srcATop,dstATop,xor,plus，modulate,screen,overlay,darken,lighten,colorDodge,colorBurn,hardLight,softLight,difference,exclusion,multiply,hue,saturation,color,luminosity,  
}



<Button>

定義按鈕的基色，以及按鈕的最小尺寸，內部填充和形狀的預設值 ButtonTextTheme textTheme,

按鈕文字的顏色 Color textColor,

按鈕禁用時的文字顏色 Color disabledTextColor,

按鈕背景顏色 Color color,

按鈕禁用時的背景顏色 Color disabledColor,

按鈕按下時的背景顏色 Color highlightColor,

按鈕主題，預設是淺色主題，分為深色和淺色 Brightness colorBrightness,

按鈕的填充間距 EdgeInsetsGeometry padding,

外形 ShapeBorder shape, Clip clipBehavior = Clip.none, MaterialTapTargetSize materialTapTargetSize,

高亮改變，按下和抬起時都會調用的方法 ValueChanged<bool> onHighlightChanged,

點擊時，水波動畫中水波的顏色，不要水波紋效果設置透明顏色即可 Color splashColor,



| | 1 2 3 4 5 6 7 8 | ButtonBar(  children: <Widget>[  BackButton(  color: Colors.orange,  ),  CloseButton(),  ], ), | | --- | --- |   後退、關閉 | | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 | ButtonBar(  children: <Widget>[  FlatButton(  child: Text('扁平按钮'),  onPressed: () {  print('我是扁平按钮');  },  ),  FlatButton(  child: Text(  '扁平按钮 禁用',  ),  onPressed: null,  ),  ], ), | | --- | --- |   扁平按紐 FlatButton |
| --- | --- | --- | --- | --- | --- |
| 扁平帶圖標按鈕 FlatButton.icon   | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | ButtonBar(  children: <Widget>[  FlatButton.icon(  label: Text('带图标扁平按钮'),  icon: Icon(Icons.add\_call, size: 18.0),  onPressed: () {},  ),  FlatButton.icon(  icon: const Icon(Icons.add\_call, size: 18.0),  label: const Text('带图标扁平按钮 禁用'),  onPressed: null,  ),  ], ), | | --- | --- | | 帶框按鈕 OutlineButton   | 1 2 3 4 5 6 7 8 9 10 11 12 | ButtonBar(  children: <Widget>[  OutlineButton(  onPressed: () {},  child: Text('带框按钮'),  ),  OutlineButton(  onPressed: null,  child: Text('带框按钮 禁用'),  ),  ], ), | | --- | --- | |
| 帶框圖標按鈕 OutlineButton.icon   | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 | ButtonBar(  children: <Widget>[  OutlineButton.icon(  label: Text('带框图标按钮'),  icon: Icon(Icons.add\_to\_photos, size: 18.0),  onPressed: () {},  ),  OutlineButton.icon(  disabledTextColor: Colors.orange,  icon: Icon(Icons.add\_to\_photos, size: 18.0),  label: Text('带框图标按钮 禁用'),  onPressed: null,  ),  ], ), | | --- | --- | | 立體按鈕 RaisedButton   | 1 2 3 4 5 6 7 8 9 10 11 12 | ButtonBar(  children: <Widget>[  RaisedButton(  child: Text('立体按钮'),  onPressed: () {},  ),  RaisedButton(  child: Text('立体按钮 禁用'),  onPressed: null,  ),  ], ), | | --- | --- | |
| 立體按鈕帶圖標RaisedButton.icon   | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | ButtonBar(  children: <Widget>[  RaisedButton.icon(  icon: Icon(Icons.add, size: 18.0),  label: Text('立体按钮带图标'),  onPressed: () {},  ),  RaisedButton.icon(  icon: Icon(Icons.add, size: 18.0),  label: Text('立体按钮带图标 禁用'),  onPressed: null,  ),  ], ), | | --- | --- | | MaterialButton   | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | ButtonBar(  children: <Widget>[  MaterialButton(  child: Text('Material按钮'),  onPressed: () {  // Perform some action  },  ),  MaterialButton(  child: Text('Material按钮 禁用'),  onPressed: null,  ),  ], ), | | --- | --- | |
| RawMaterialButton   | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 | ButtonBar(  children: <Widget>[  RawMaterialButton(  child: Text('RawMaterial按钮'),  onPressed: () {  // Perform some action  },  ),  RawMaterialButton(  child: Text('RawMaterial按钮 禁用'),  onPressed: null,  ),  ], ), | | --- | --- | | 浮動按鈕FloatingActionButton   | 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 | ButtonBar(  children: <Widget>[  FloatingActionButton(  child: const Icon(Icons.add),  heroTag: '浮动按钮',  onPressed: () {  // Perform some action  },  tooltip: '浮动按钮提示1',  ),  FloatingActionButton(  child: const Icon(Icons.add),  onPressed: null,  heroTag: '浮动按钮 禁用',  tooltip: '浮动按钮提示2',  ),  ], ), | | --- | --- | |

<Container>

1. Def.

Container({ Key key, …})

| 容器子Widget對齊方式 this.alignment | 容器內部padding this.padding |
| --- | --- |
| 背景色 Color color | 背景裝飾 Decoration decoration |
| 前景裝飾 this.foregroundDecoration | 容器的寬度 double width |
| 容器的高度 double height | 容器大小的限制條件 BoxConstraints constraints |
| 容器外部margin this.margin | 變換，如旋轉 this.transform |
| 容器內子Widget this.child |  |

1. BoxDecoration 裝飾

const BoxDecoration({…})

| 背景色this.color, | 背景图片this.image, |
| --- | --- |
| 边框样式this.border, | 边框圆角this.borderRadius, |
| 阴影this.boxShadow, | 渐变this.gradient, |
| 背景混合模式this.backgroundBlendMode, | 形状this.shape = BoxShape.rectangle, |

<Row>

1. Row：水平横向，children 子元素按照水平方向進行排列。

Row({Key key,…})

| 主軸方向上的對齊方式（Row的主軸是橫向軸）  MainAxisAlignment mainAxisAlignment = MainAxisAlignment.start, |
| --- |
| 在主軸方向（Row的主軸是橫向軸）佔有空間的值，預設是max  MainAxisSize mainAxisSize = MainAxisSize.max, |
| 在交叉軸方向（Row是縱向軸）的對齊方式，Row的高度等於子元素中最高的子元素高度  CrossAxisAlignment crossAxisAlignment = CrossAxisAlignment.center, |
| 水準方向子元素的排列方向：從左到右排列還是反向  TextDirection textDirection, |
| 表示縱軸（垂直）的對齊排列方向，預設是VerticalDirection.down，表上到下。 這個參數一般用於Column元件里  VerticalDirection verticalDirection = VerticalDirection.down, |
| 字元對齊基線方式  TextBaseline textBaseline, |
| 子元素集合  List<Widget> children = const <Widget>[], |

1. MainAxisAlignment主轴属性：主軸方向上的對齊方式，Row 是横向軸為主軸。

enum MainAxisAlignment {}

| 按照主轴起点对齐，例如：按照靠近最左侧子元素对齐start, | 将子元素放置在主轴的末尾，按照末尾对齐 end, |
| --- | --- |
| 子元素放置在主轴中心对齐 center, | 将主轴方向上的空白区域均分，使得子元素之间的空白区域相等，首尾子元素都靠近首尾，没有间隙。有点类似于两端对齐spaceBetween, |
| 将主轴方向上的空白区域均分，使得子元素之间的空白区域相等，但是首尾子元素的空白区域为1/2 spaceAround, | 将主轴方向上的空白区域均分，使得子元素之间的空白区域相等，包括首尾子元素 spaceEvenly, |

1. CrossAxisAlignment交叉屬性：交叉軸方向的對齊方式，Row是縱軸。Row高=子元素中最高的子元素高度enum CrossAxisAlignment {}

子元素在交叉轴上起点处展示start,

子元素在交叉轴上末尾处展示end,

子元素在交叉轴上居中展示center,

让子元素填满交叉轴方向stretch,

在交叉轴方向，使得子元素按照baseline对齐baseline,

1. MainAxisSize在主轴方向子元素占有空间的方式，Row 的主轴是横向轴。默认是 max  
   enum MainAxisSize {}

根据传入的布局约束条件，最大化主轴方向占用可用空间，也就是尽可能充满可用宽度max,  
与max相反，是最小化占用主轴方向的可用空间 min,

<Column>

縱向排序的子元素，參數方法同上。