

Flutter 基础 Widget —— 图片和Icon

本节讲 Flutter 图片和Icon的使用。

主要讲一下 Image 和 Icon 这两种 Widget 的使用:

1. [Image \(https://docs.flutter.io/flutter/widgets/Image-class.html\)](https://docs.flutter.io/flutter/widgets/Image-class.html)
2. [Icon \(https://docs.flutter.io/flutter/widgets/Icon-class.html\)](https://docs.flutter.io/flutter/widgets/Icon-class.html)

[1.Image \(https://docs.flutter.io/flutter/widgets/Image-class.html\)](https://docs.flutter.io/flutter/widgets/Image-class.html)

Image 是显示图片的 Widget。

代码所在位置

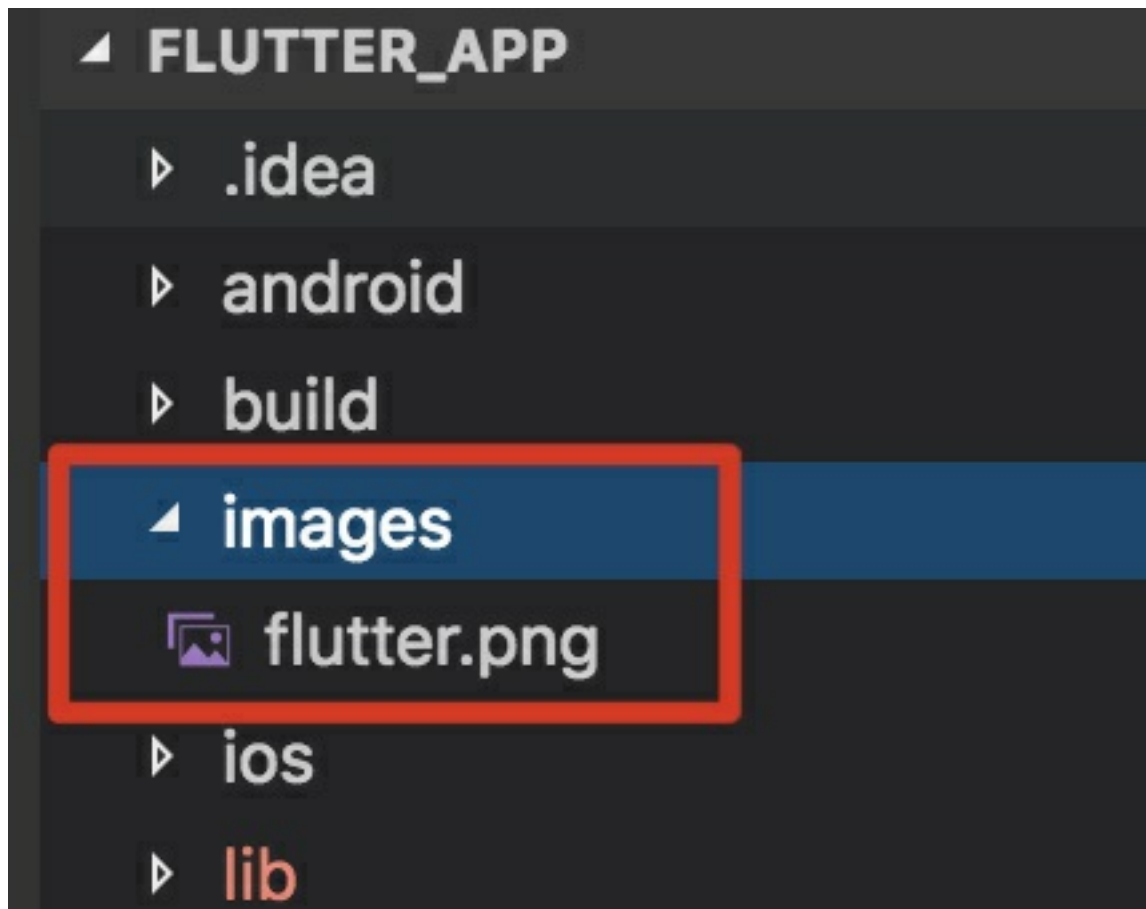
flutter_widget_demo/lib/image/ImageWidget.dart

Image.asset 的快速上手

Image.asset 使用有些特殊，因为这里涉及到如何把本地图片资源添加到 Flutter APP 里。

将本地图片资源添加到 Flutter APP

1. 在 Flutter 工程的根目录下创建一个 images 目录（名字随便取），然后将一张 flutter.png 的图片拷贝到该目录，如下图：



2. 打开 pubspec.yaml ,在 flutter 中添加图片的配置信息：
如果只添加一张图片：

```
flutter:  
  uses-material-design: true  
  
assets:  
  - images/flutter.png
```

或者可以把整个目录配置上，就不用一张张图片添加了：

```
flutter:  
  uses-material-design: true  
  
  assets:  
    - images/
```

3. 使用的时候，需要传入图片的路径：

```
Image.asset("images/flutter.png"),
```

Image.asset 的使用

Image.asset 的最简单使用方式就是传入图片路径：

```
Image.asset("images/flutter.png"),
```

Image.asset 写到一个页面的完整 Demo代码如下：

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

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

class ImageWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Flutter Demo",
      theme: ThemeData(
        primaryColor: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: Text("Flutter UI基础Widget -- 图片和Icon")),
        body:
Image.asset("images/flutter.png", fit:
BoxFit.cover, ),
      ),
    );
  }
}
```

运行的效果如下：



Image.network 的快速上手

Image.network 的最简单使用方式就是传入图片 URL:

```
Image.network("https://tingsa.baidu.com/timg?image&quality=80&size=b9999_10000&sec=1557781801455&di=17f9f2fc3ded4efb7214d2d8314e8426&imgtype=0&src=http%3A%2F%2Fimg2.mukewang.com%2F5b4c075b000198c216000586.jpg"),
```

Image.network 写到一个页面的完整 Demo 代码如下:

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

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

class ImageWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Flutter Demo",
      theme: ThemeData(
        primaryColor: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: Text("Flutter UI基础Widget -- 图片和Icon")),
        body:
Image.network("https://tingsa.baidu.com/timg?image&quality=80&size=b9999_10000&sec=1557781801455&di=17f9f2fc3ded4efb7214d2d8314e8426&imgtype=0&src=http%3A%2F%2Fimg2.mukewang.com%2F5b4c075b000198c216000586.jpg"),
      ),
    );
  }
}
```

运行的效果如下：

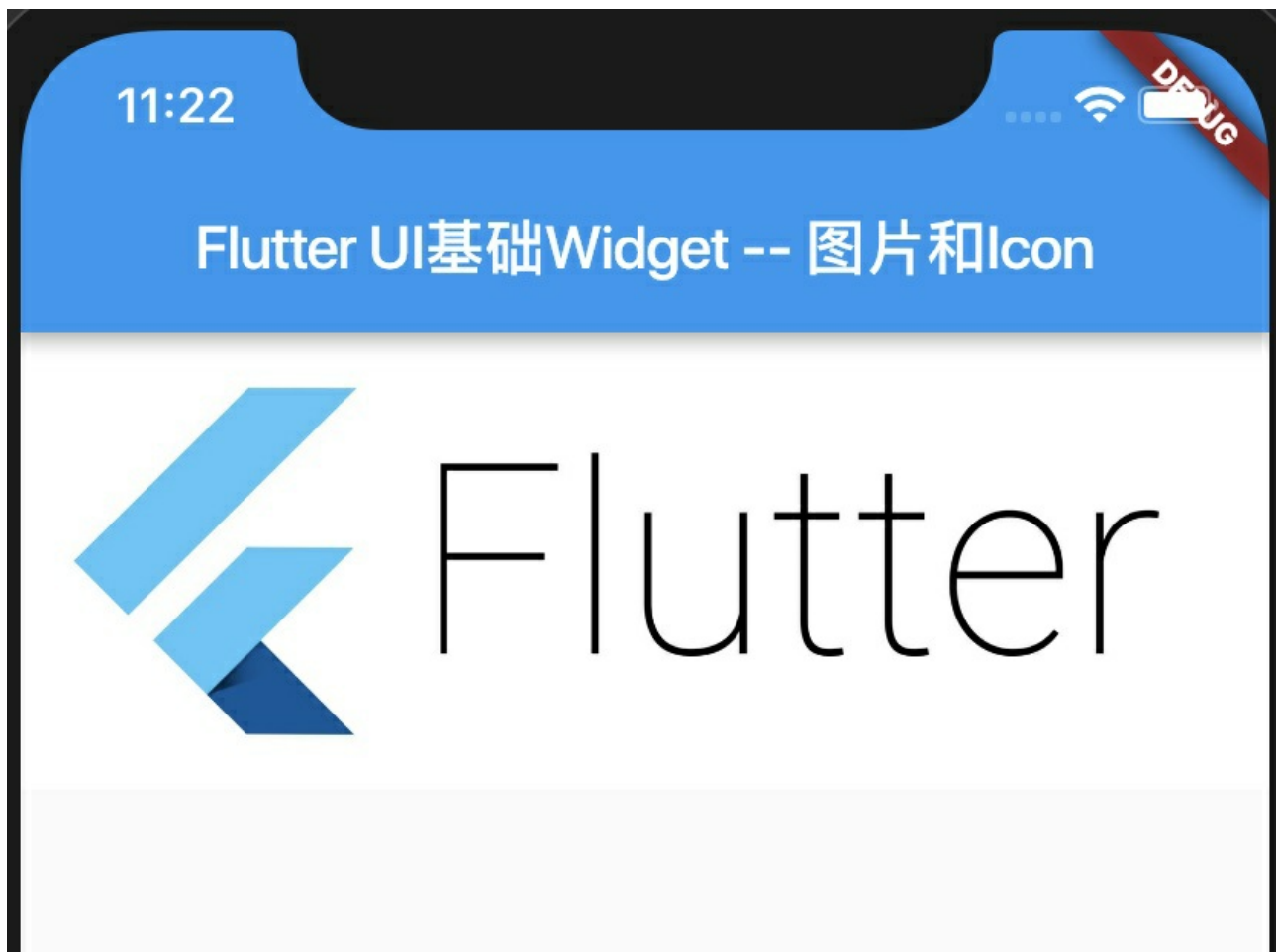


Image 的构造函数及参数说明

Image 总共有 5 个构造函数，除了默认的构造函数，还有四种命名构造函数：

```
class Image extends StatefulWidget {  
  
    const Image({  
        @required this.image,  
        ...  
    })  
  
    Image.network(String src, {  
        ...  
    })  
  
    Image.file(File file, {  
        ...  
    })  
  
    Image.asset(String name, {  
        ...  
    })  
  
    Image.memory(Uint8List bytes, {  
        ...  
    })  
  
}
```

我们一般使用四种命名构造函数，而这四种命名构造函数也表示了图片的不同来源：

1. `Image.network(url)`：从网络加载显示图片

这里需要传入图片的url，

2. `Image.file(file)`：从本地文件加载显示图片

这里需要传入图片的本地地址

- 3. Image.asset(name): 从Flutter APP的资源文件里加载显示图片

这里需要传入Flutter APP图片资源文件的路径及图片名字

- 4. Image.memory(bytes): 从内存加载显示图片

这里需要传入图片的bytes数据，类型是Uint8List

Image 的五个构造函数的参数都有所不同，这里讲一下共同都有的参数：

参数名字	参数类型	意义	必选 or 可选
key	Key	Widget 的标识	可选
scale	double	图形显示的比例	可选
semanticsLabel	String	给 Image 加上一个语义标签 没有实际用处	可选
width	double	图片的宽 如果为null的话，则图像将选择最佳大小显示，而且会保留其固有宽高比的大小	可选
height	double	图片的高 如果为null的话，则图像将选择最佳大小显示，而	可

		且会保留其固有宽高比的大小	选
color	Color	图片的混合色值	可选
colorBlendMode	BlendMode	图片与颜色的混合模式	可选
fit	BoxFit	用于在图片的显示空间和图片本身大小不同时指定图片的适应模式	可选
alignment	Alignment	图片的对齐方式	可选
repeat	ImageRepeat	当图片本身大小小于显示空间时，指定图片的重复规则	可选
centerSlice	Rect	在这个矩形范围内的图片会被当成.9的图片	可选
matchTextDirection	bool	图片的绘制方向 true:从左往右 false:从右往左	可选
gaplessPlayback	bool	当图像提供者更改时 true：继续显示旧图像 false：简单地显示任何内容	可选
filterQuality	FilterQuality	设置图片的过滤质量	可选

- fit: BoxFit, 用于在图片的显示空间和图片本身大小不同时指定图片的适应模式。

BoxFit的值

含义

BoxFit.fill

会拉伸填充满显示空间，图片本身长宽比会发生变化，图片会变形。

BoxFit.contain	会按图片的长宽比放大后居中填满显示空间，图片不会变形，超出显示空间部分会被剪裁。
BoxFit.cover	这是图片的默认适应规则，图片会在保证图片本身长宽比不变的情况下缩放以适应当前显示空间，图片不会变形。
BoxFit.fitWidth	图片的宽度会缩放到显示空间的宽度，高度会按比例缩放，然后居中显示，图片不会变形，超出显示空间部分会被剪裁。
BoxFit.fitHeight	图片的高度会缩放到显示空间的高度，宽度会按比例缩放，然后居中显示，图片不会变形，超出显示空间部分会被剪裁。
BoxFit.scaleDown	对齐目标框内的源（默认情况下，居中），并在必要时缩小源以确保源适合框内。这与contains相同，如果这会缩小图像，否则它与none相同。
BoxFit.none	图片没有适应策略，会在显示空间内显示图片，如果图片比显示空间大，则显示空间只会显示图片中间部分

- alignment: Alignment, 图片的对齐方式

Alignment的值	含义
Alignment.topLeft	左上对齐
Alignment.topCenter	上部居中对齐
Alignment.topRight	右上对齐
Alignment.centerLeft	中间居左对齐
Alignment.center	中间对齐
Alignment.centerRight	中间居右对齐
Alignment.bottomLeft	左下对齐
Alignment.bottomCenter	底部居中对齐
Alignment.bottomRight	右下对齐

- repeat: ImageRepeat, 当图片本身大小小于显示空间时, 指定图片的重复规则。

ImageRepeat的值	含义
ImageRepeat.repeat	在x和y轴上重复图像, 直到填满空间。
ImageRepeat.repeatX	在x轴上重复图像, 直到填满空间。
ImageRepeat.repeatY	在y轴上重复图像, 直到填满空间。
ImageRepeat.noRepeat	不重复

- filterQuality: FilterQuality, 设置图片的过滤质量

FilterQuality的值	含义
FilterQuality.none	最快的过滤。
FilterQuality.low	比none的过滤质量好, 但是比none的时间要长。
FilterQuality.medium	比low的过滤质量好, 但是也比low的时间要长
FilterQuality.high	过滤质量最高, 但也最慢

2.Icon

(<https://docs.flutter.io/flutter/class.html>)

Flutter 中, 可以像 web 开发一样使用 iconfont, iconfont 即“字体图标”, 它是将图标做成字体文件, 然后通过指定不同的字符而显示不同的图片。用于显示 iconfont 的就是 Icon Widget。

iconfont 和图片相比有如下优势：

1. 体积小：可以减小安装包大小。
2. 矢量的：iconfont 都是矢量图标，放大不会影响其清晰度。
3. 可以应用文本样式：可以像文本一样改变字体图标的颜色、大小对齐等。
4. 可以通过 TextSpan 和文本混用。

代码所在位置

flutter_widget_demo/lib/image/IconWidget.dart

Icon 的快速上手

Icon 的使用：

```
Icon(  
  Icons.android,  
  size: 50.0,  
  color: Colors.green,  
)
```

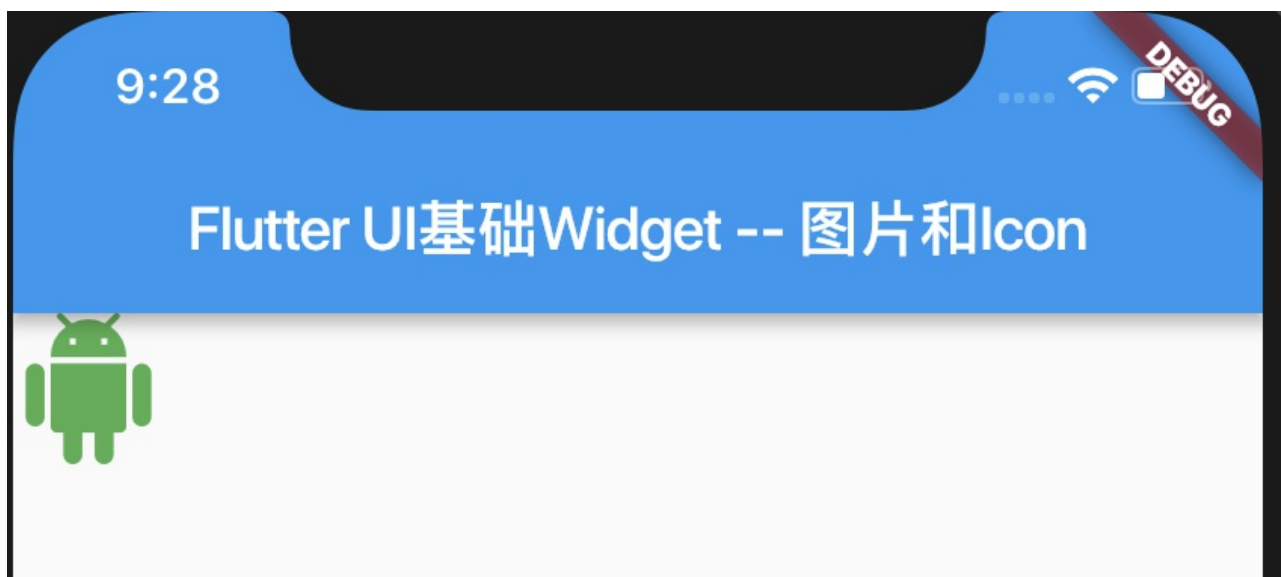
Icon 写到一个页面的完整 Demo 代码如下：

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

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

class IconWidget extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: "Flutter Demo",
      theme: ThemeData(
        primaryColor: Colors.blue,
      ),
      home: Scaffold(
        appBar: AppBar(title: Text("Flutter UI基础Widget -- 图片和Icon")),
        body: Icon(
          Icons.android,
          size: 50.0,
          color: Colors.green,
        ),
      ),
    );
  }
}
```

运行效果：



Icon 的构造函数及参数说明

Icon 的构造函数为:

```
class Icon extends StatelessWidget {  
  
  const Icon(this.icon, {  
    Key key,  
    this.size,  
    this.color,  
    this.semanticLabel,  
    this.textDirection,  
  }) : super(key: key);  
  
  ...  
}
```

参数名字	参数类型	意义	必选 or 可选
key	Key	Widget 的标识	可选
color	double	icon的大小	可选
size	Color	icon的颜色	可选

semanticsLabelString	给 Image 加上一个语义标签 没有实际用处	可选
textDirection	TextDirection Icon的绘制方向	可选