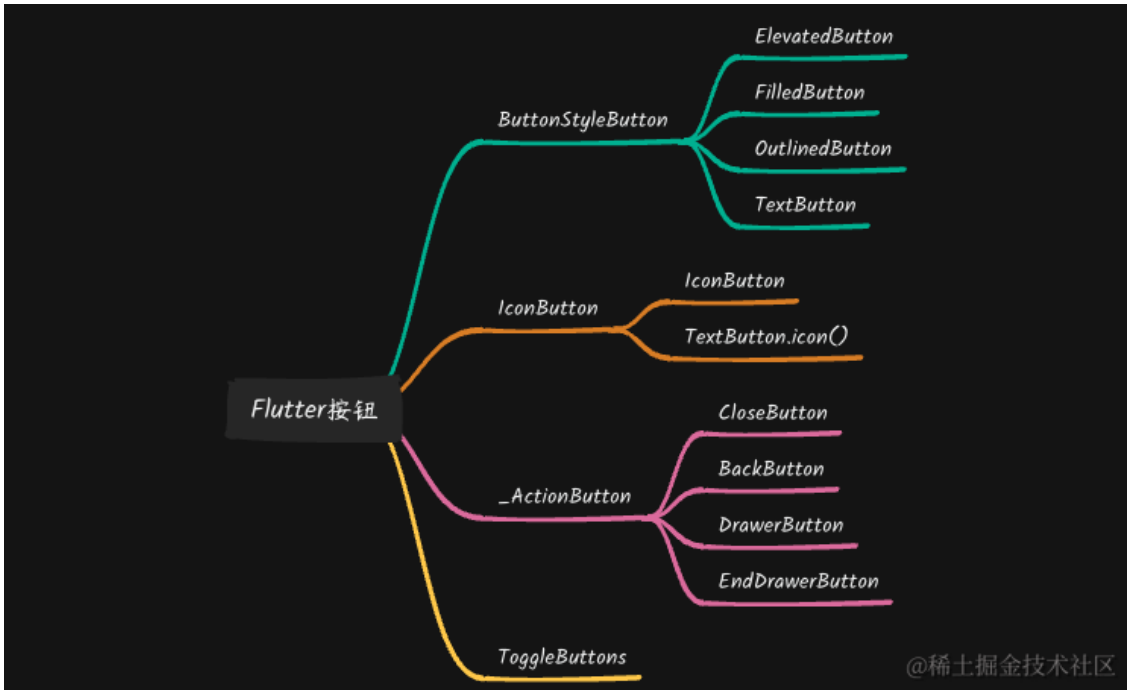
Flutter基建 – 按钮全解析

Taonce

本篇为Flutter基建的第四篇文章，主要介绍Flutter中按钮相关组件，在按钮组件这方面，Flutter为开发者提供了非常全面的样式，总结下来可以分为普通按钮、图标按钮、动作按钮和开关按钮，在日常使用Flutter开发过程中，很多的样式都无需自己在封装一套，可以使用原生提供的这些样式拿来即用，下面我们逐一学习下这样按钮的使用。



Flutter基建 - Dart基础类型

Flutter基建 - Dart方法和类

Flutter基建 - 文本组件

Flutter基建 - 按钮全解析

Flutter基建 - 布局组件全面解析

普通样式按钮ButtonStyleButton

此系列的按钮是我们日常开发中接触最多的一种样式了，它实现的效果就如它名称一样，一种按钮样式的按钮（哈哈🤣），此样式按钮一共有四种子类，分别为TextButton、ElevatedButton、FilledButton和OutlinedButton，那么我们就进入实战环节看看这些按钮实现的具体效果吧。

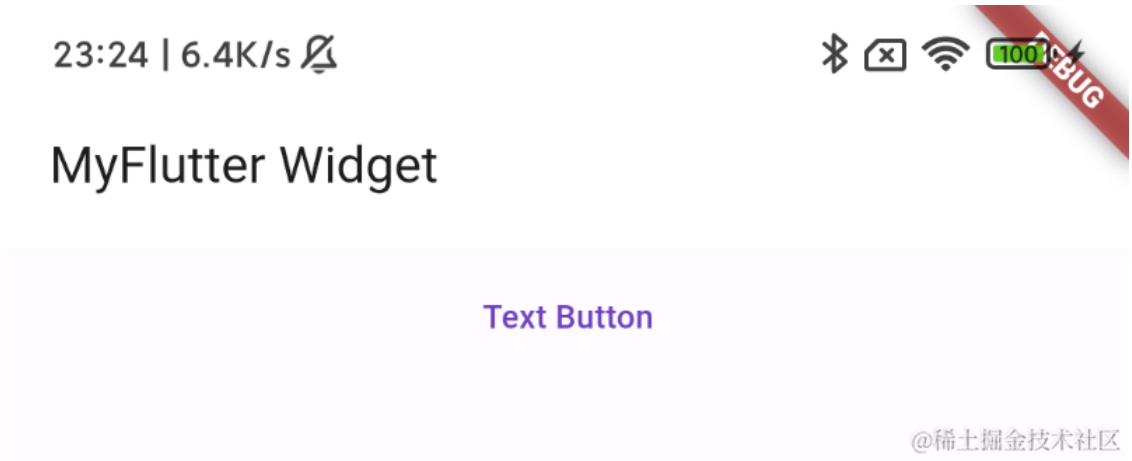
TextButton

elevated 升高的，高层的，outlined 画出...的轮廓

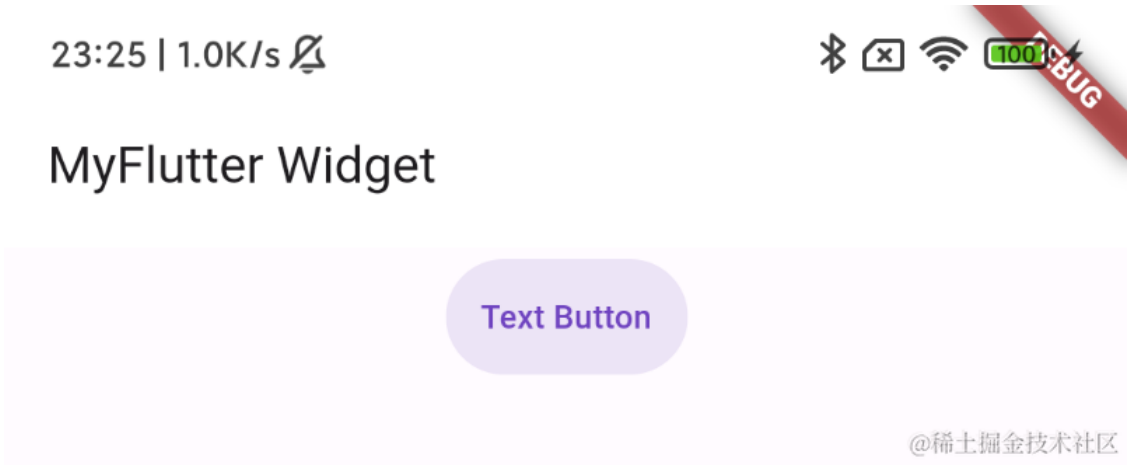
TextButton是一种文本按钮，用于显示单纯文本并且不加任何边框和阴影修饰的一种按钮，使用起来也是极为简单

```
TextButton(
  onPressed: () {},
  child: const Text("Text Button"),
),
```

正常状态下样式：



手指按下时样式：



可以看到TextButton在正常状态下看上去就是一个文本组件，没有其它任何修饰，只是在手指按下去之后会有个圆润的背景阴影。

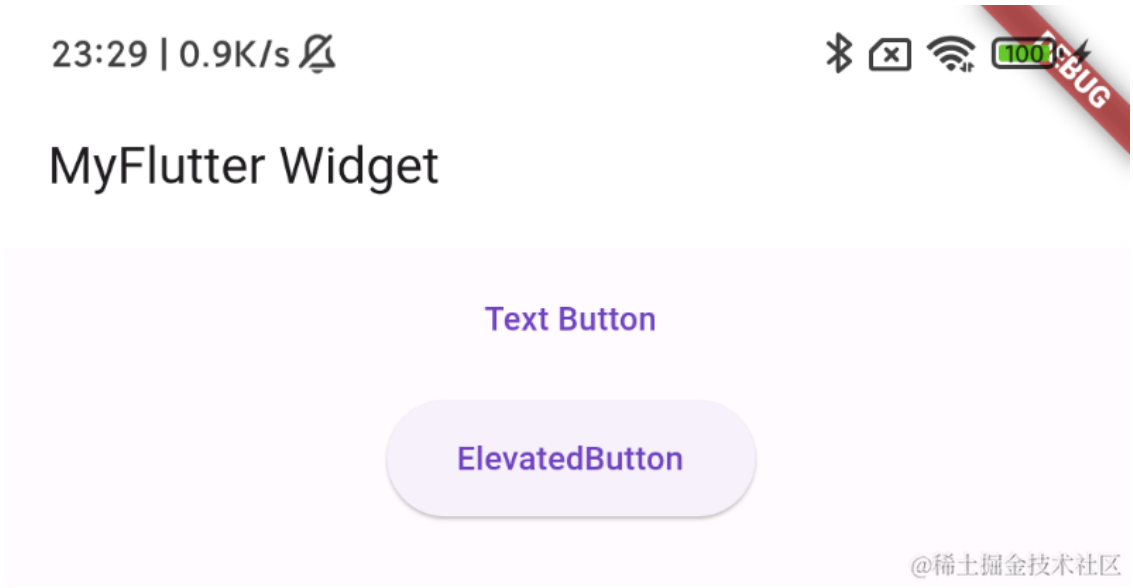
TextButton有一个必传参数就是onPressed，用于检测点击事件，可以在此事件回调中处理点击事件的具体逻辑；onLongPressed则是用于检测长按事件。

ElevatedButton

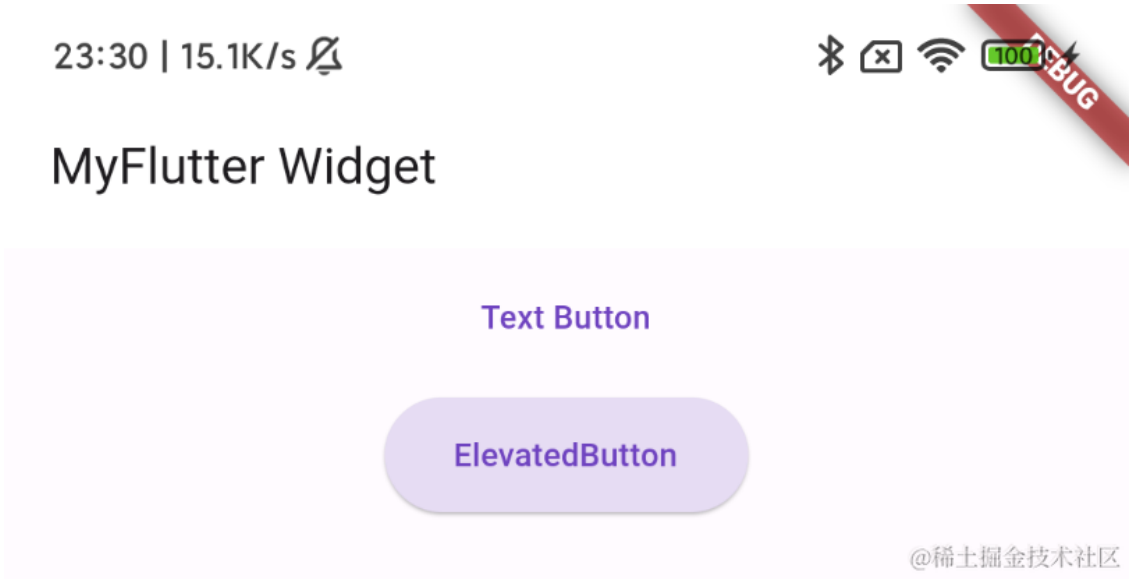
ElevatedButton可以理解为TextButton的提升版，加上了背景颜色和带弧度的边框，并且在手指按下去之后，它的背景色会加深，有个明显的区别。

```
ElevatedButton(
  onPressed: () {},
  child: const Text('ElevatedButton'),
),
```

正常状态下样式：



手指按下时样式：



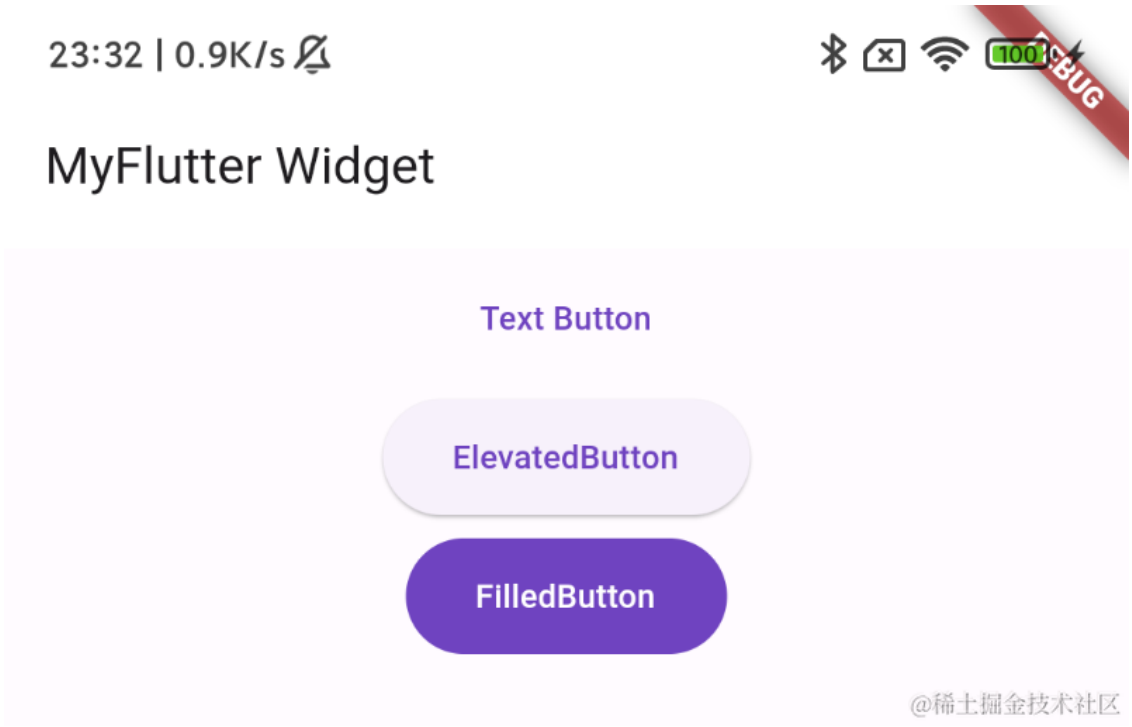
FilledButton

FilledButton为填充样式的按钮，说白了就是给背景上个色，这种样式在手指点击时给人的观感不是很强烈，只有微弱的颜色变化，大家可以自己感受一下。

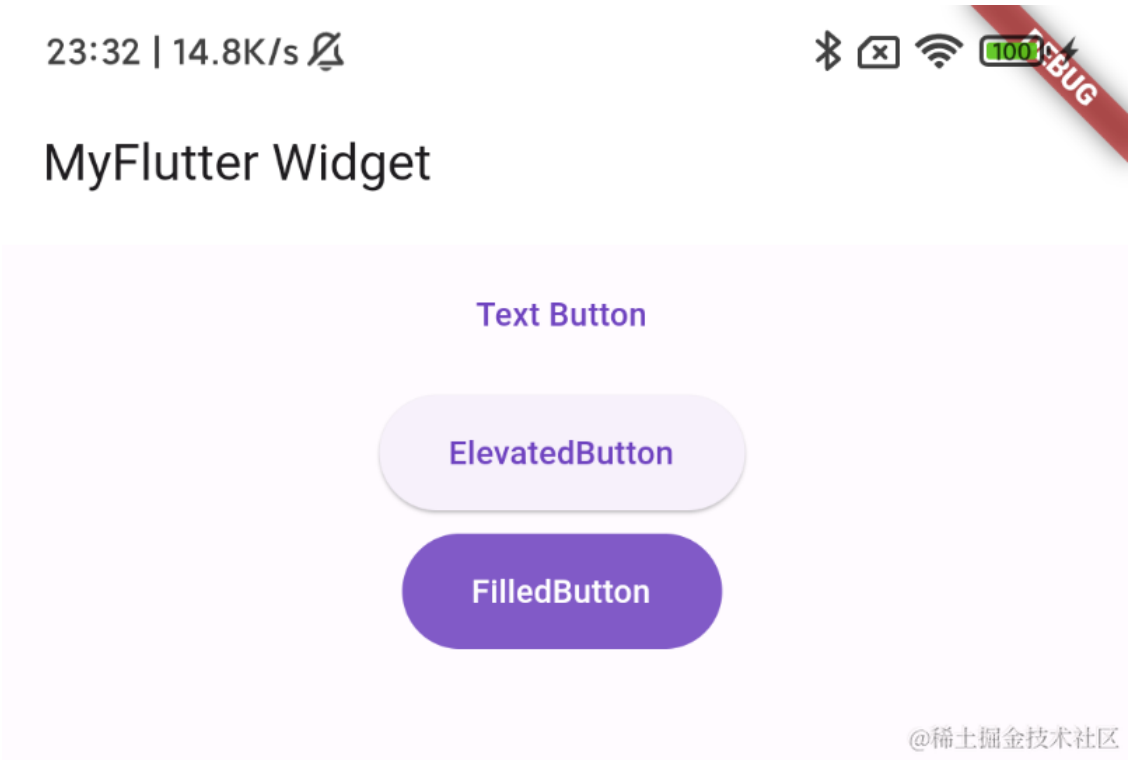
```
FilledButton(f
  onPressed: () {},
```

```
  child: const Text('FilledButton'),
),
```

正常状态下样式：



手指按下时样式：

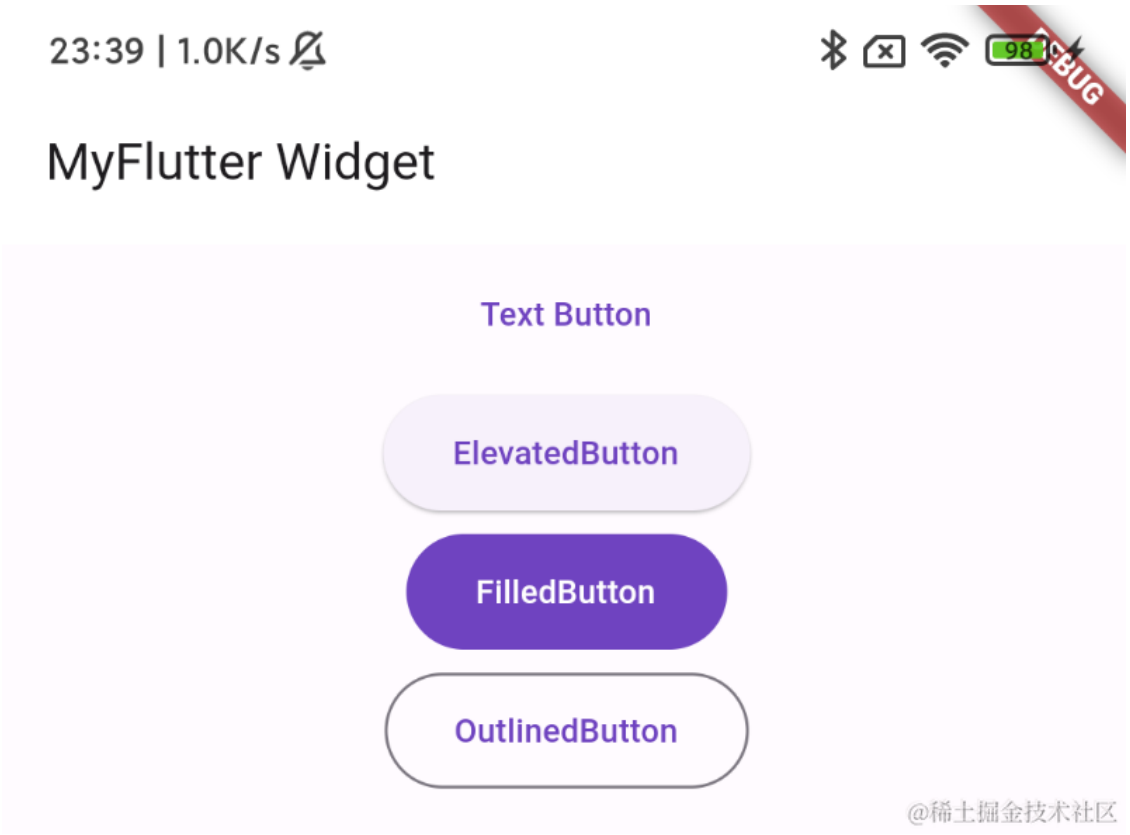


OutlinedButton

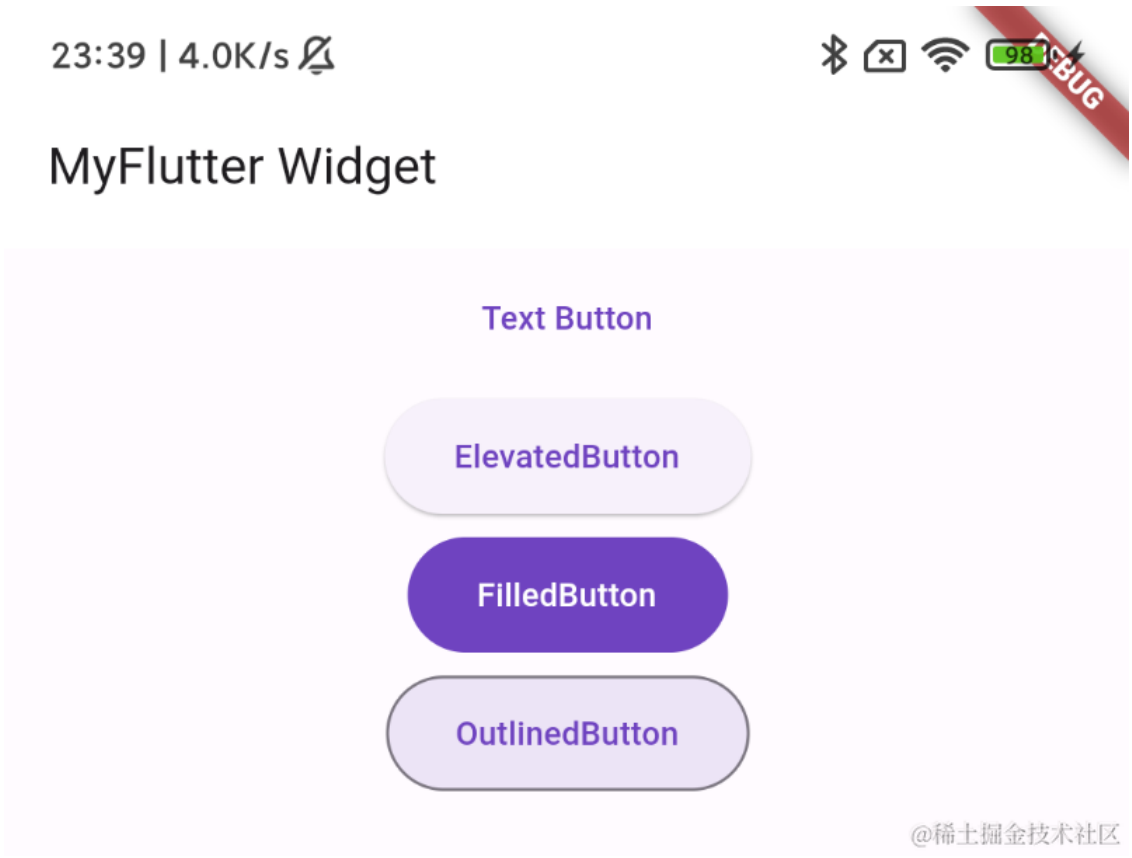
OutlinedButton区别则是在于边框，它的背景色默认为白色，给边框添加了主题色。

```
OutlinedButton(
  onPressed: () {},
  child: const Text('OutlinedBut
),
```

正常状态下样式：


23:39 | 1.0K/s
MyFlutter Widget
DEBUG
Text Button
ElevatedButton
FilledButton
OutlinedButton
@稀土掘金技术社区

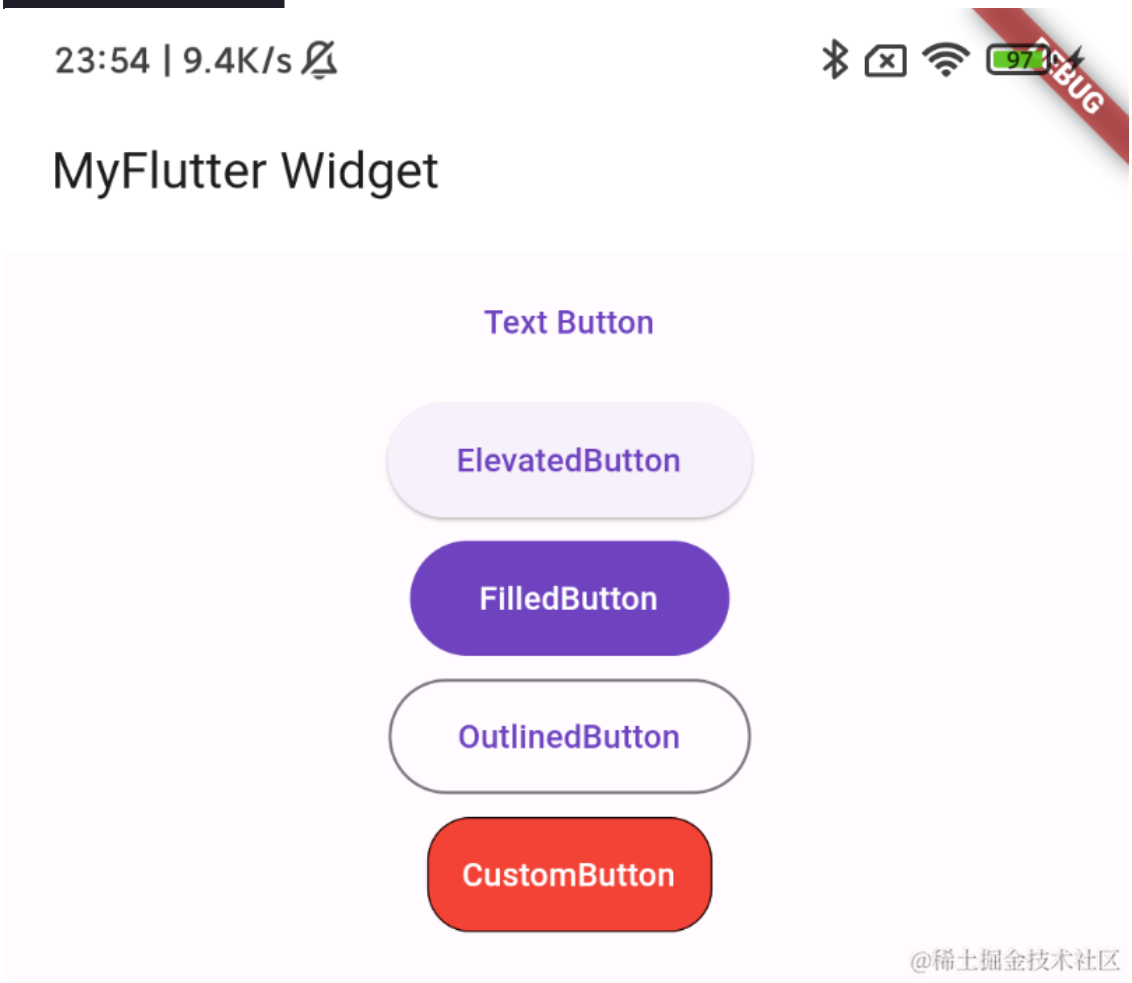
手指按下时样式：



上面就是 ButtonStyleButton 四种样式的按钮，每一种都有自己的特色，小伙伴们可以按需选择，如果都不满足自己的需要也可以使用 ButtonStyleButton 自定义自己的按钮样式，下面我们以这种形式自己定义一套目标的样式。

```
TextButton(
  onPressed: () {},
  style: const ButtonStyle(
    backgroundColor:                    WidgetStatePropertyAll
MaterialStatePropertyAll<Color>(Colors.red),
    elevation: MaterialStatePropertyAll(10.0),
    shape: MaterialStatePropertyAll(
      RoundedRectangleBorder(
        borderRadius:
BorderRadius.all(Radius.circular(15)),
        side: BorderSide(color: Colors.black,
width: 0.5),
      ),
    ),
```

```
  ),
  child: const Text(
    'CustomButton',
    style: TextStyle(color: Colors.white),
  ),
),
```



这里我们使用TextButton方法中style参数自定义了一套按钮的样式，style参数为ButtonStyle类型，ButtonStyle类中可以使用backgroundColor自定义背景颜色，这里需要注意的是传入的为MaterialStatePropertyAll类型；elevation用于定义阴影；shape用于定义按钮的边框，这里使用的是RoundedRectangleBorder，在内部定义了15的圆弧角度和边

框宽度为0.5颜色为黑色样式，实现的效果如上图所示。

上面这样实现方式有可能会觉得复杂，也可以在child的文本组件中定义想要的样式，这里就不过多介绍了～

IconButton

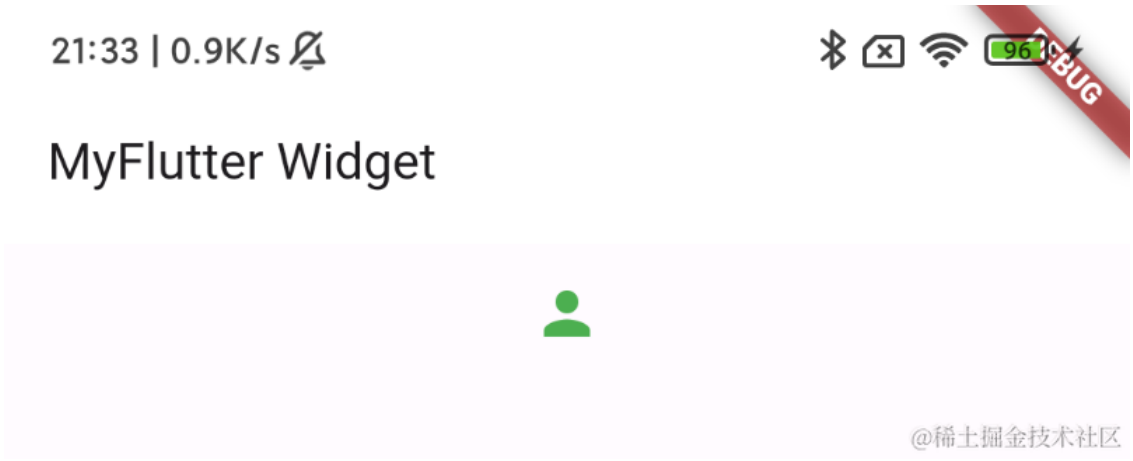
图标按钮

IconButton是专门为图标按钮设计的，它的样式仅显示一个Icon，有点类似单纯的Icon组件，不过它增加了许多额外的属性，下面我们来看看IconButton具体的实现。

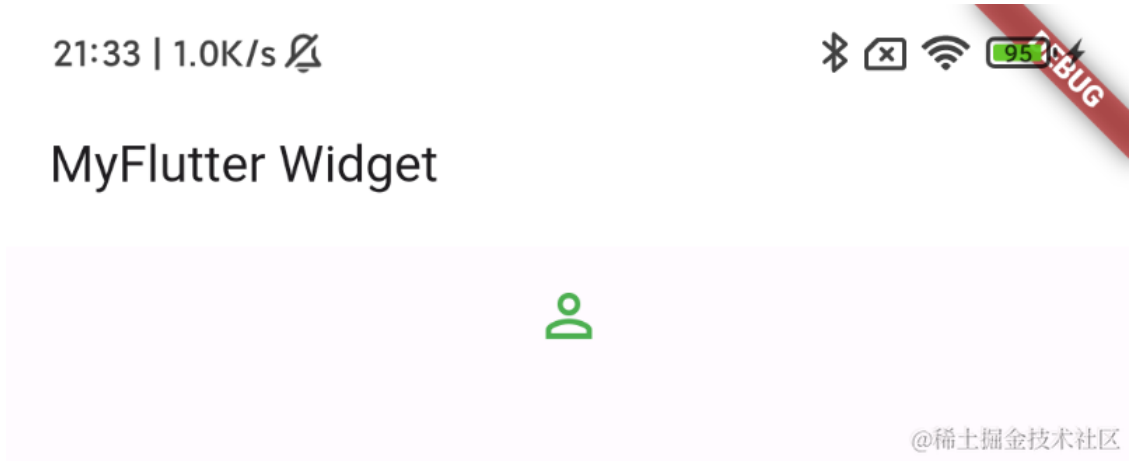
```
var iconButtonIsSelected = false;
```

```
IconButton(
  onPressed: () {
    setState(() {
      iconButtonIsSelected = !iconButtonIsSelected;
    });
  },
  selectedIcon: const Icon(Icons.person_outline),
  isSelected: iconButtonIsSelected,
  color: Colors.green,
  icon: const Icon(Icons.person),
),
```

正常状态下样式：



selected状态下样式：

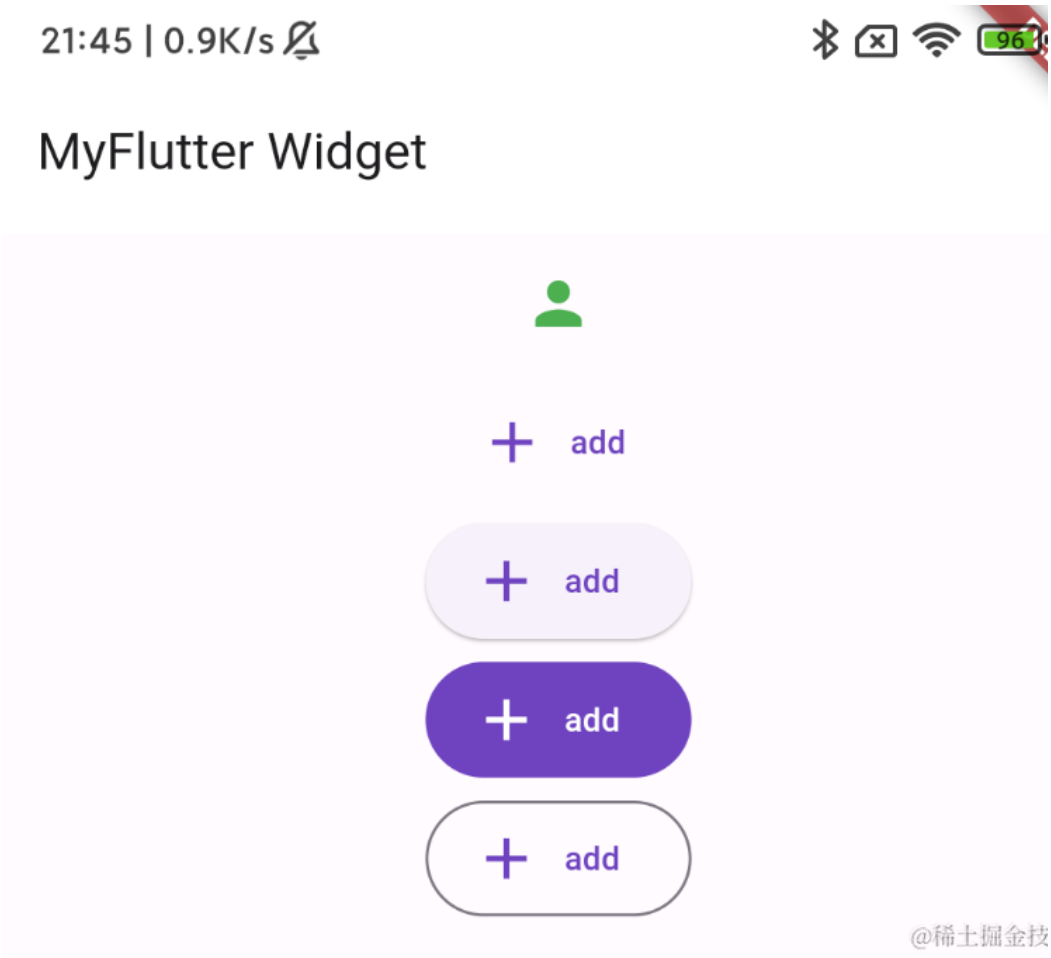


IconButton提供了正常状态和selected状态下不同的样式，我们可以分别给这两种状态设置不一样的Icon图标，并且还可以通过color参数给Icon设置自定义的颜色值。

如果需要使用单独的Icon按钮，可以采用IconButton来实现。其实除了IconButton以外，Flutter还提供了图标+文本样式的按钮，相对于IconButton样式图标+文本样式的按钮在日常开发中使用的过多一点，接下来我们看看这种样式是如何实现的。

图标+文本按钮

```
TextButton.icon(
  onPressed: () {},
  icon: const Icon(Icons.add),
  label: const Text('add'),
),
ElevatedButton.icon(
  onPressed: () {},
  icon: const Icon(Icons.add),
  label: const Text('add'),
),
FilledButton.icon(
  onPressed: () {},
  icon: const Icon(Icons.add),
  label: const Text('add'),
),
OutlinedButton.icon(
  onPressed: () {},
  icon: const Icon(Icons.add),
  label: const Text('add'),
),
```



普通按钮样式的TextButton、ElevatedButton、FilledButton和OutlinedButton都提供了icon()的扩展方法，用于实现按钮文字前面加图标的功能，在用法上面只是多了Widget icon和Widget label两个参数，分别用于显示图标和提示文本的效果。

ActionButton

ActionButton

ActionButton其实就是Flutter给我们额外封装了一层的IconButton，它内部还是采用的IconButton来实现，可以先看一下它的构造方法：

```
const _ActionButton({
```

```
  super.key,
  this.color,
  required this.icon,
  required this.onPressed,
  this.style,
});
```

构造方法还是比较简单的，定义了color颜色值、icon图标、onPressed点击事件和style按钮样式，然后在其内部的build(context)方法中直接调用了IconButton🤣，这里注意一点就是点击事件，如果我们传入了onPressed参数，那么就会采用我们传入的回调，否则就会直接调用_onPressedCallback(context)方法，此方法下面会介绍。

```
@override
Widget build(BuildContext context) {

assert(debugCheckHasMaterialLocalizations(context))
;
  return IconButton(
    icon: icon,
    style: style,
    color: color,
    tooltip: _getTooltip(context),
    onPressed: () {
      if (onPressed != null) {
        onPressed!();
      } else {
        _onPressedCallback(context);
      }
    },
  );
}
```

它是一个抽象类，我们不可以直接拿来使用，Flutter帮我们封装了几种特定的ActionButton，分别为：BackButton、

CloseButton、DrawerButton和EndDrawerButton。

具体样式

```
BackButton(),
CloseButton(),
DrawerButton(),
EndDrawerButton()
```

这四种样式的ActionButton使用起来应该是最为简单的了，可以不传入任何参数，直接引入即可，这里我们挑BackButton看看内部的实现是如何。

```
class BackButton extends _ActionButton {
  const BackButton({
    super.key,
    super.color,
    super.style,
    super.onPressed,
```

```
  }) : super(icon: const BackButtonIcon());
```

```
  @override
  void _onPressedCallback(BuildContext context) =>
Navigator.maybePop(context);
```

```
  @override
  String _getTooltip(BuildContext context) {
    return
MaterialLocalizations.of(context).backButtonTooltip
;
  }
}
```

BackButton内部实现也是极为简单的，构造方法中实现了父类的icon参数，传入了BackButtonIcon()，这个返回Icon组件，然后复写了_onPressedCallback方法，调用的是Navigator.maybePop()，此方法意思就是将当前页面退出页面堆栈，通俗来说就是退出当前页面，Navigator知识后续文章会详细介绍。

看到这大致就明白了BackButton的具体实现了，如果我们需要一个点击退出界面的按钮，可以考虑使用BackButton组件，毕竟Flutter已经为我们实现好了逻辑。

其余三个ActionButton就不再一一介绍了，小伙伴可以自行阅读源码哈～

ToggleButtons　toggle 切换键，转换键

ToggleButtons是一个特殊的按钮，说它是按钮其实有点勉强，它其实是一组可以开关的按钮，根据当前开关的状态有不一样的表现，子组件为List，并且子组件中每一个都可以相应

onPressed点击事件。

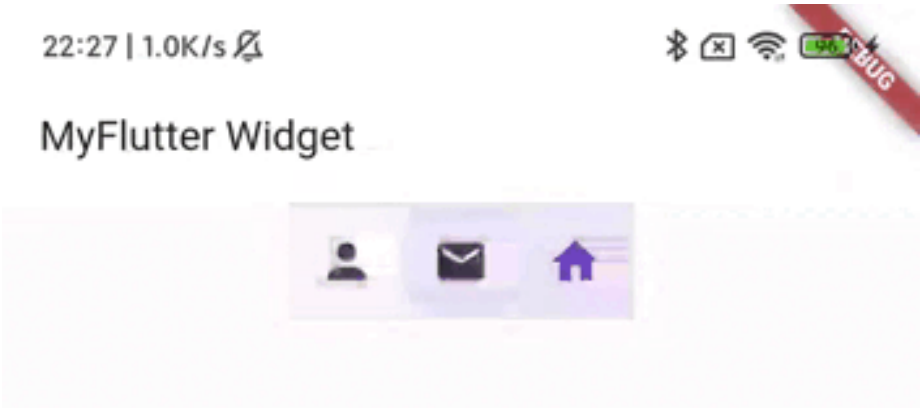
```
var isSelected = [false, false, false];
```

```
ToggleButtons(
  isSelected: isSelected,
  onPressed: (int index) {
    setState(() {
      isSelected[index] = !isSelected[index];
    });
  },
  children: const <Widget>[
    Icon(Icons.person),
    Icon(Icons.email),
    Icon(Icons.home),
  ],
),
```

这里我们分别介绍下三个重要的参数：

- isSelected参数为List类型，根据子组件的数量，需要传入对应大小的bool列表，这里默认开关都是关闭状态；
- onPressed参数为Function(int index)类型，index就是点击的子组件对应的下标，这里就是将对应下标的状态置为相反值；
- children参数为List类型，需要传入和isSelected大小一致的组件数量。

最后来看看具体效果：



写在最后

本篇文章全面的介绍了 Flutter 中按钮的相关知识，希望可以帮助大家了进一步了解和熟悉按钮的相关知识，后续会循序渐进逐步接触 Flutter 更多的知识。

我是**Taonce**，如果觉得本文对你有所帮助，帮忙关注、赞或者收藏三连一下，谢谢😆😆～