# 手势相关Widget

2021年7月1日 10:30

Flutter中的手势系统有两个独立的层，第一层是原始指针事件，它描述了屏幕上指针，比如触摸、鼠标、触控笔的位置和移动。第二层是手势，由一个或多个指针移动组成的动作会被识别为不同的手势。

一、指针事件

指针表示用户与设备屏幕交互的原始数据。有四种类型的指针事件：

1. PointerDownEvent：指针接触到屏幕的特定位置
2. PointerMoveEvent： 指针已从屏幕上的一个位置移动到另一个位置
3. PointerUpEvent： 指针已停止接触屏幕
4. PointerCancelEvent：此指针的输入不再指向此应用，通俗来讲就是事件取消

在指针按下时，Flutter 框架会对当前应用程序执行命中测试，以确定指针与屏幕接触的位置存在哪个 Widget 上，然后将 PointerDownEvent 事件（以及该指针的后续事件）调度到命中测试找到的最内部的 Widget，事件的分配路径为：从最里面的 Widget 到树的根路径上的所有 Widget。

二、手势

手势表示由一个或多个指针移动组成的动作。主要有以下几种：

1. 点击

   onTapDown：指针已经在特定位置与屏幕接触。

   onTapUp：指针停止在特定位置与屏幕接触。

   onTap ： 点击事件触发。

   onTapCancel： 先前指针触发的 onTapDown 不会再触发点击事件。
2. 双击

   onDoubleTap：用户快速连续两次在同一位置轻敲屏幕。
3. 长按

   onLongPress：指针在相同位置长时间保持与屏幕接触。
3. 垂直拖动

   onVerticalDragStart：指针已经与屏幕接触并可能开始垂直移动。

   onVerticalDragUpdate 指针与屏幕接触并已沿垂直方向移动。

   onVerticalDragEnd 先前与屏幕接触并垂直移动的指针不再与屏幕接触，并且在停止接触屏幕时以特定速度移动。
4. 水平拖动

   onHorizontalDragStart：指针已经接触到屏幕并可能开始水平移动

   onHorizontalDragUpdate：指针与屏幕接触并已沿水平方向移动

   onHorizontalDragEnd：先前与屏幕接触并水平移动的指针不再与屏幕接触，并在停止接触屏幕时以特定速度移动。

三、使用GestureDetector

要检测单击、双击、垂直拖动等手势，只需要用GestureDetector嵌套要检测手势Widget并实现想要监听的手势的方法。

```
classGestureDetectorWidgetextendsStatelessWidget{
constGestureDetectorWidget({Key?key}):super(key:key);

@override
Widgetbuild(BuildContextcontext){
returnMaterialApp(
title:'Flutter',
home:Scaffold(
appBar:AppBar(title:Text('demo'),),
body:Center(
child:GestureDetector(
child:Text('手势识别'),
onTap:(){
print('点击');
},
onDoubleTap:(){
print('双击');
},
onLongPress:(){
print('长按');
},
onHorizontalDragStart:(DragStartDetailsdetails){
print('水平拖动');
print(details);
},
)
),
),
);
}
}
```

执行相应的操作，就会在控制台上打印相关的内容

```
I/flutter ( 6590): 点击
I/flutter ( 6590): 长按
I/flutter ( 6590): 双击
I/flutter ( 6590): 水平拖动
I/flutter ( 6590): DragStartDetails(Offset(189.4, 420.0))
```

四、使用Dismissible

Flutter提供了Dismissible来实现滑动删除。

```
classDismissibleWidgetextendsStatelessWidget{
constDismissibleWidget({Key?key,requiredthis.items}):super(key:key);

finalList<String>items;

@override
```

```
Widgetbuild(BuildContextcontext){
returnScaffold(
appBar:AppBar(
title:Text('demo'),
),
body:ListView.builder(
itemCount:items.length,
itemBuilder:(context,index){
returnDismissible(
key:Key(items[index]),
onDismissed:(direction){
items.removeAt(index);
print(index);
},
child:ListTile(
leading:Icon(Icons.access_time),
title:Text('${items[index]}'),
),
);
},
),
);
}
}
```

可以左右删除某一行，当执行删除操作时，ListView中的onDismissed方法会被回调