Flutter Fundamentals
Gestures

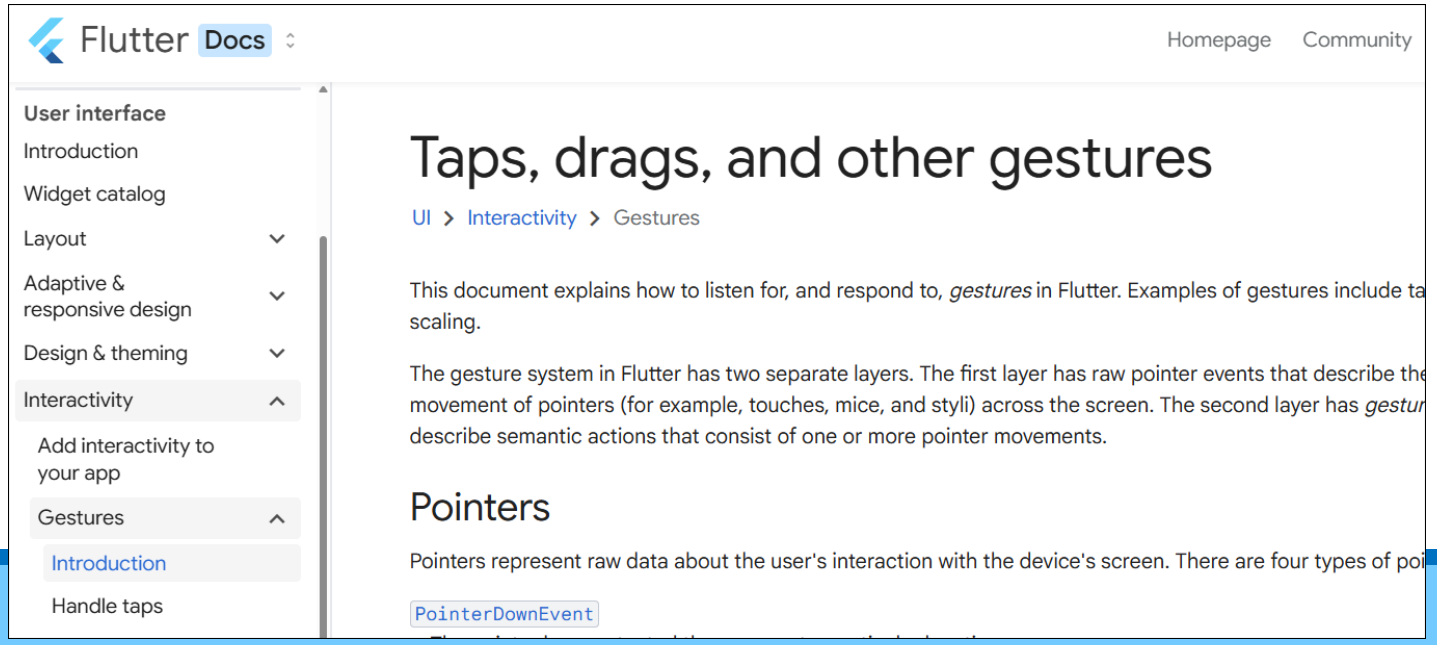
Peter Kassenaar –
info@kassenaar.com

Using Gestures

Listening for- and responding to user interaction with gestures like tap, drag, scale.

Tap, drag, other gestures

- You can listen to specific gestures in Flutter
  - Tap, drag, pinch, zoom,
- Many widgets already respond to gestures (tap)
- Otherwise: use `GestureDetector()`
- https://docs.flutter.dev/ui/interactivity/gestures



Possible Gestures

- Tap
  - Events: `onTap()`, `onTapDown()`, `onTapUp()`,
- Double tap
  - `onDoubleTap()`
- LongPress
  - `onLongPress()`, etc...
- Vertical Drag
- Horizontal Drag
- Pan

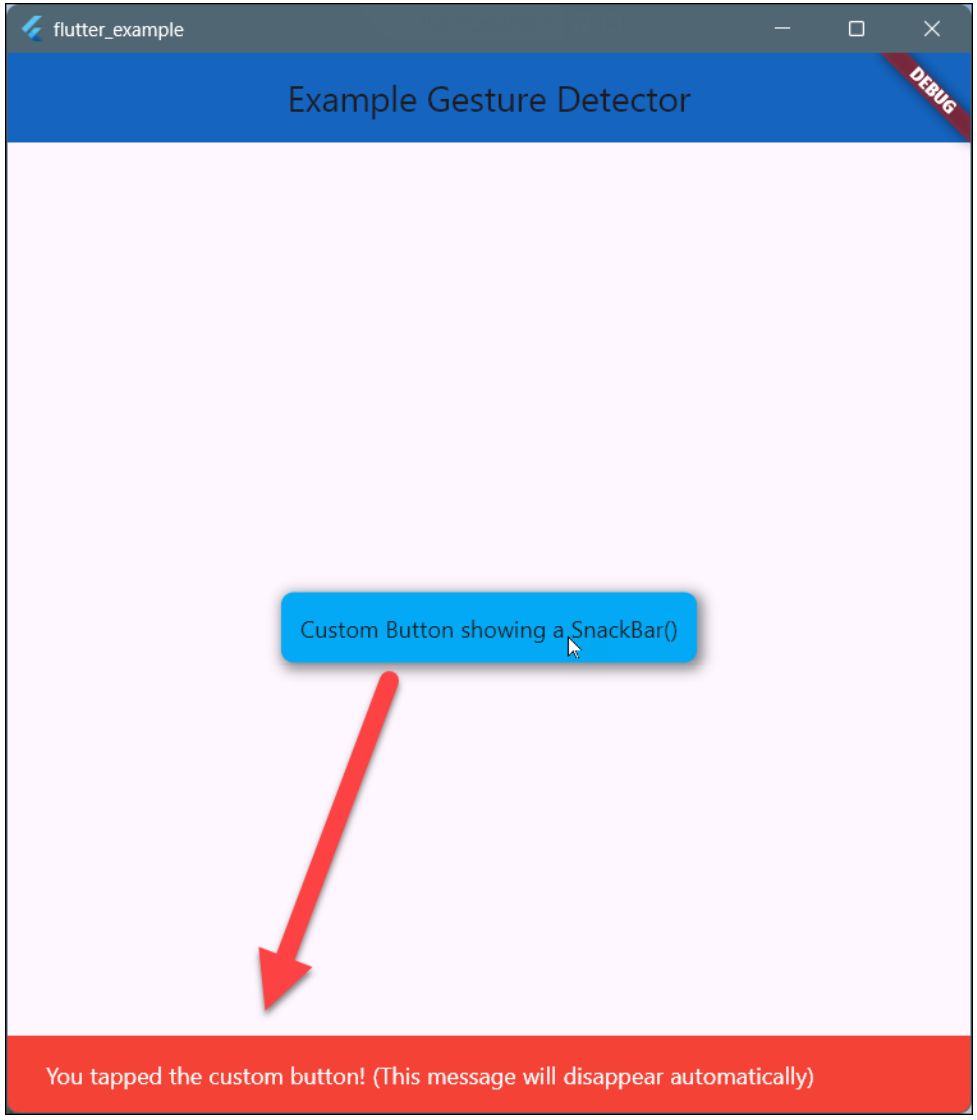
Example GestureDetector

- Example: creating a custom button that shows a `SnackBar()` when tapped
- General steps
  - Create your UI element (here: a `Container()`, acting as a button)
  - Wrap it in a `GestureDetector()` with an `onTap()` callback

Code for GestureDetector

```
// The text in the SnackBar()
const snackbarText =
    'You tapped the custom button! (This message will disappear automatically)’;
…
GestureDetector(
                // When the child is tapped, show a snackbar.
            onTap: () {
              const snackBar = SnackBar(
                content: Text(
                  snackbarText,
                  style: TextStyle(color: Colors.white),
                ),
                backgroundColor: Colors.red,
                duration: Duration(seconds: 2),
              );
              ScaffoldMessenger.of(context).showSnackBar(snackBar);
            },
            // Our custom button
            child: Container(
              padding: const EdgeInsets.all(12),
              decoration: BoxDecoration(
                color: Colors.lightBlue,
                borderRadius: BorderRadius.circular(8),
                child: const Text('Custom Button showing a SnackBar()'),
                ),
…
}
```

Result GestureDetector()



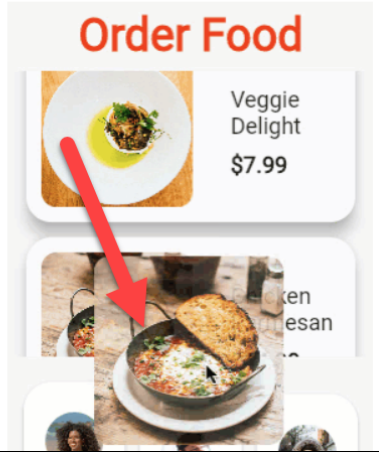
More options

Drag a UI element

Cookbook > Effects > Drag a UI element

Drag and drop is a common mobile app interaction. As the user long presses (sometimes called *touch & hold*) o
a widget, another widget appears beneath the user's finger, and the user drags the widget to a final location an
releases it. In this recipe, you'll build a drag-and-drop interaction where the user long presses on a choice of
food, and then drags that food to the picture of the customer who is paying for it.

The following animation shows the app's behavior:



https://docs.flutter.dev/ui/interactivity/gestures

Workshop

- Look at the `GestureDetector()` example and study the code
  - Optional: create your own `GestureDetector`, not using the `onTap()`, but one of the many other gestures available (`onDoubleTap`, `onLongPress`, etc)
  - See https://docs.flutter.dev/ui/interactivity/gestures for options
- Optional: use `GestureDetector()` to create a drag/drop option for your container
- Optional: implement swipe-to-dismiss