Flutter Forms & Gestures

Key Points

Notes

1. **User Interaction and Gestures**

   a. Interaction Events

      i. Flutter contains a gestures detector for its widgets. For example all button widgets contain `onPressed` and `onTap`

   b. The `GestureDetector` Widget

      i. You can wrap this widget around another widget and let its child listen for interaction from user.

      ii. For example:

```
GestureDetector(
    onTap: () => print("tapped!"),
    child: Text("Tap Me"),
);
```

      iii. There many gestures that you can use but below are the most common gestures:

         1. `onTap`
         2. `onTapUp`
         3. `onTapDown`
         4. `onLongPress`
         5. `onDoubleTap`
         6. `onHorizontalDragStart`
         7. `onVerticalDragDown`
         8. `onPanDown`
         9. `onScaleStart`

      iv. Refer to Flutter Gestures Widget https://flutter.dev/docs/development/ui/advanced/gestures

   c. The `Dismissible` Widget

      i. A gesture that you can use to implement “swipe to remove” function.

      ii. It is also one of the widgets that requires you to pass in a key. This is similar to a list or array.

Summary

Key Points

Notes

i. Refer to Flutter Dismissible Widget
https://flutter.dev/docs/cookbook/gestures/dismissible

2. **Flutter Forms**

a. The `Form` widget

i. It manages the state of all the fields (inputs) in the form without having to manage each field's state individually.

ii. The form widget works by passing a key of type FormState, which it is associated with global key. This allows you to access the object state everywhere.

b. `GlobalKey<FormState>`

i. It is similar to a controller when working with form logic because it contains various methods pertaining to `FormState` object.

c. `FormField` widgets

i. The FormState is handy when working with multiple inputs fields that are related to each other but they must be in FormField widgets.

ii. For example:

```
return FormField(
    child: Checkbox(
        //...
```

iii. There are three (3) `FormField` widgets:

1. `FormField` – The standard field, which can turn any input widget into a form field.
2. `TextFormField` – A specialized field that wraps a text field.
3. `DropdownButtonFormField` – A convenience widget that wraps a DropdownButton in a form field.

iv. Refer to Form Validation
https://flutter.dev/docs/cookbook/forms/validation

Summary

Key Points

Notes

3. **Form UI and Working with Focus Nodes**
    a. `InputDecoration`
        i. It accepts many arguments to style your form field including changing background color, text color, shape, label and more.

    b. `AutoFocus`
        i. It is one of FocusNodes that allows you to change focus when an external event is triggered such as validation, leaving a screen, etc.

        ii. Refer to Focus and Text Fields https://flutter.dev/docs/cookbook/forms/focus

4. **Managing Form States with Form Methods**
    a. `onChanged`
        i. It is called when one of the form fields changes. A callback (call another function or execute a block of code) is executed and all the form fields will be rebuilt.

        ii. Refer to API https://api.flutter.dev/flutter/widgets/Form/onChanged.html

    b. `onWillPop`
        i. It disables or overrides the back button that contains the form.

        ii. Refer to API https://api.flutter.dev/flutter/widgets/Form/onWillPop.html

    c. `onSaved`
        i. It holds the final value when the form is saved through `FormState.save`.

        ii. Refer to API https://api.flutter.dev/flutter/widgets/FormField/onSaved.html

Summary

Key Points	Notes
	d. `validator` i. It validates an input and returns error string if the input is invalid, or null otherwise. ii. Refer to API https://api.flutter.dev/flutter/widgets/FormField/validator.html **5. References** a. Windmill, E. (2020). Flutter in Action (1st Ed.). USA: Manning Publications. b. Flutter Official Documentation. Retrieved on 1 December 2020 from https://flutter.dev/docs

Summary