R2S
Resource Software Solution

StatefulWidget & Form

Contents

- Stateless Widget and Stateful Widget
- Form and Validation

What is State (1)

- The familiar math formula **y = f(x)**. It is a function, when we have the **value of x**, based on a **function f** we get the **value of y**. Whenever **x changes**, it gives us a **new value of y**.
- Flutter is similar, it uses a formula of:


```dart
@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(title: const Text('Stateful Widget'),),
      body: GestureDetector(
          onTap: _handleTap,
          child: Center(
            child: Container(
              width: 200, height: 200,
              decoration: BoxDecoration(
                color: _active ? Colors.lightGreen : Colors.grey,
              ),  // BoxDecoration
              child: Center(
                child: Text(
                  _active ? 'Active' : 'Inactive',
                  style: const TextStyle(fontSize: 32, color: Colors.white),
                ),  // Text
              ),  // Center
            ),  // Container
          )  // Center
      ),  // GestureDetector
    ),  // Scaffold
  );  // MaterialApp
}
```

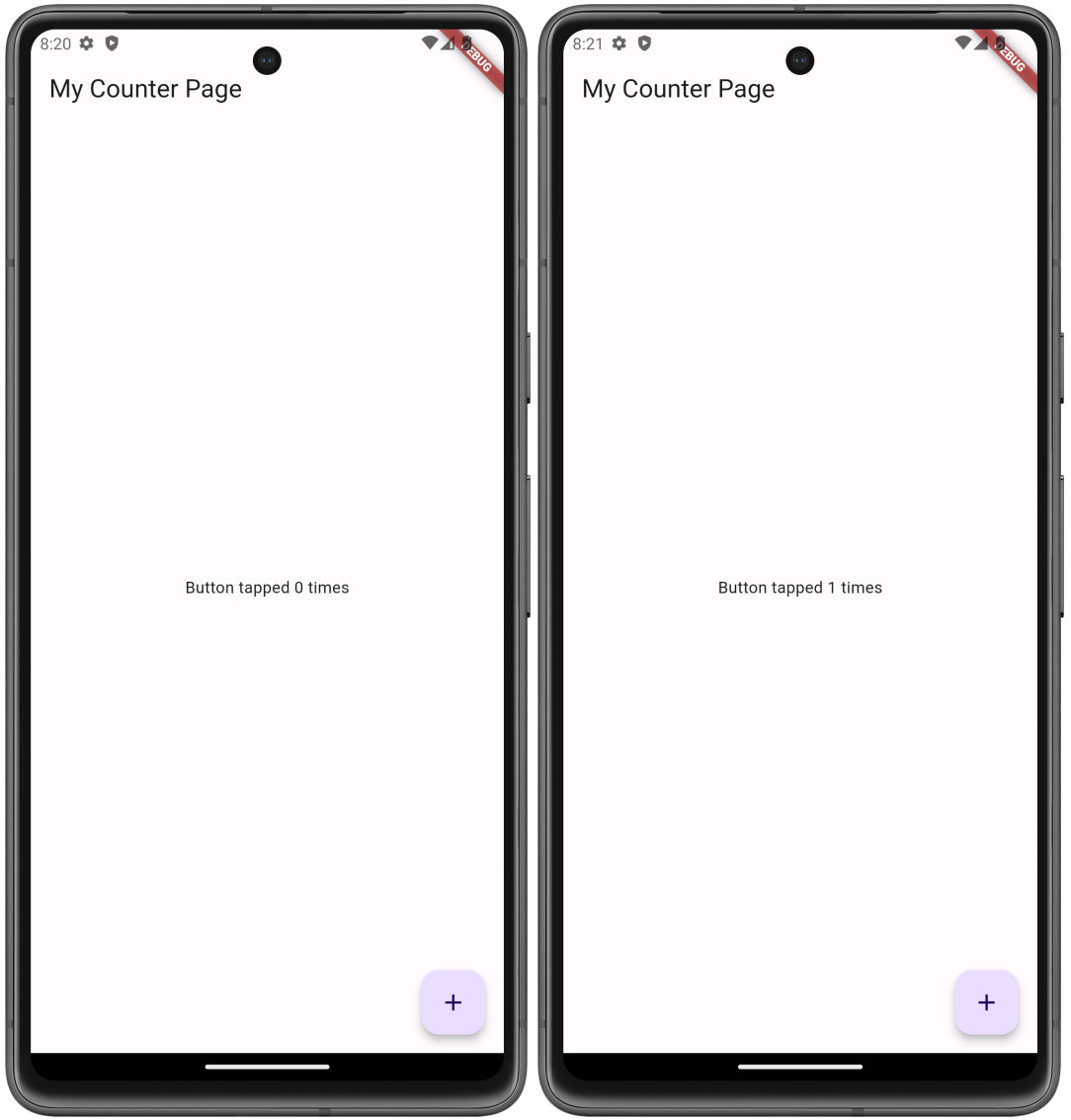
What is State (2)

UI = f(state)

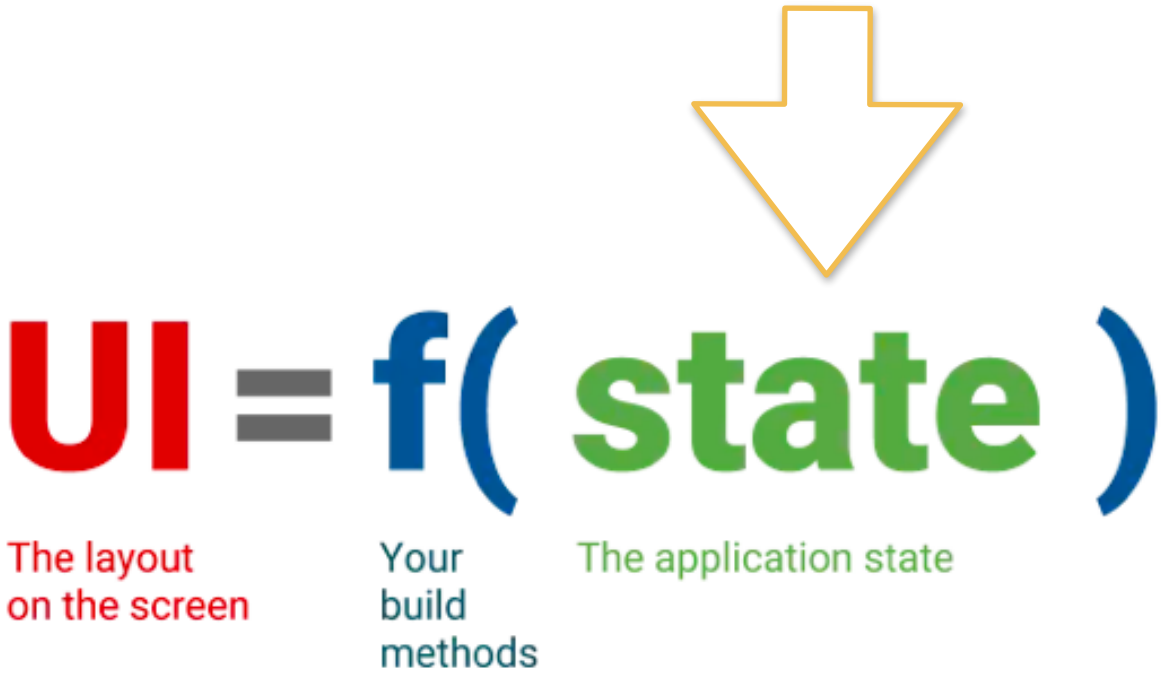
The layout on the screen

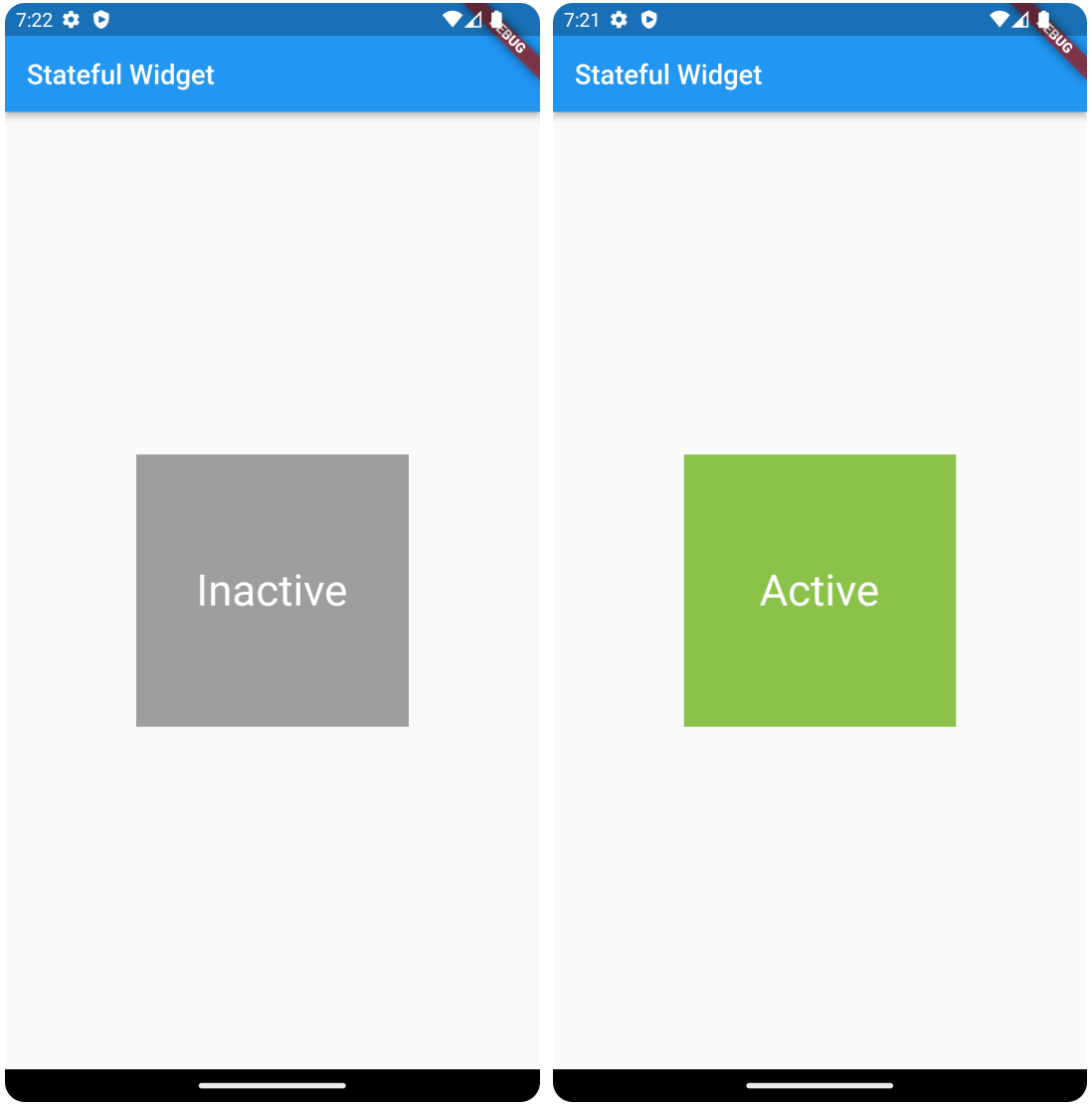
Your build methods

The application state



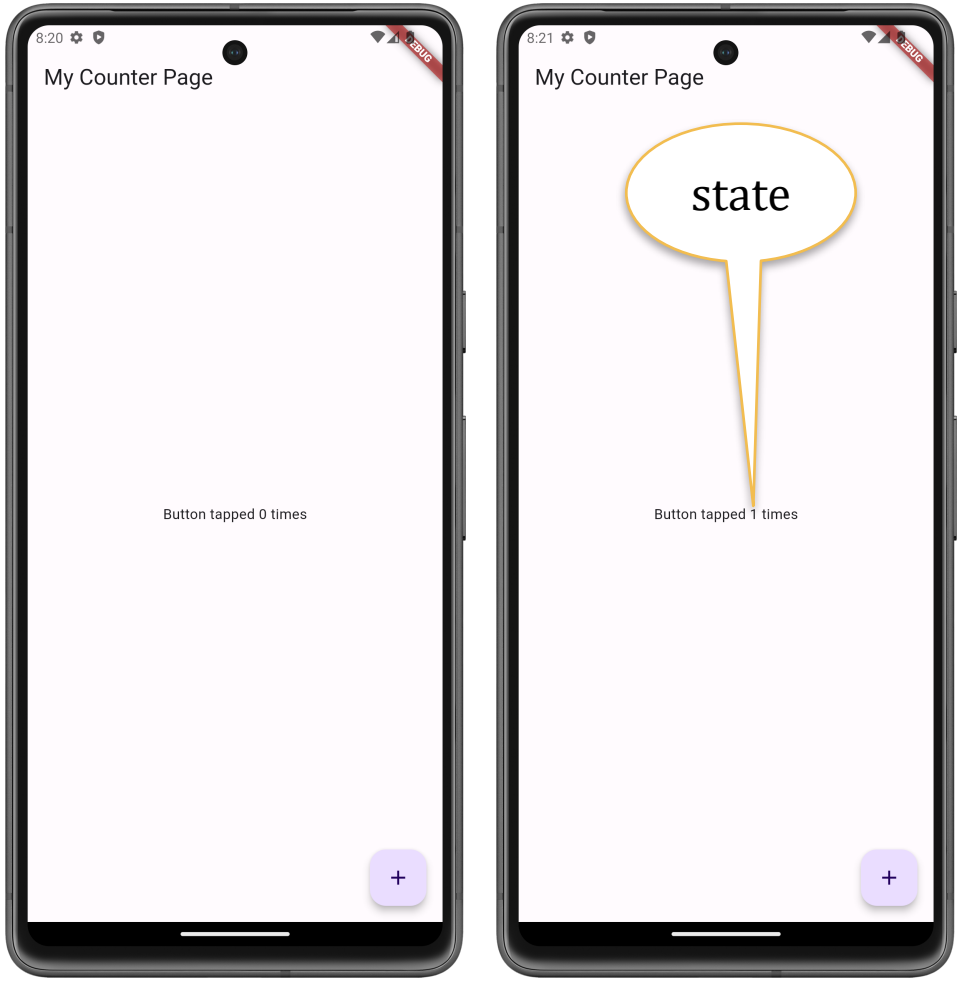
What is State (3)





What is State (4)

- "State" refers to the **data values that can change over time.** For example, the **number** it displays would be its **state**.

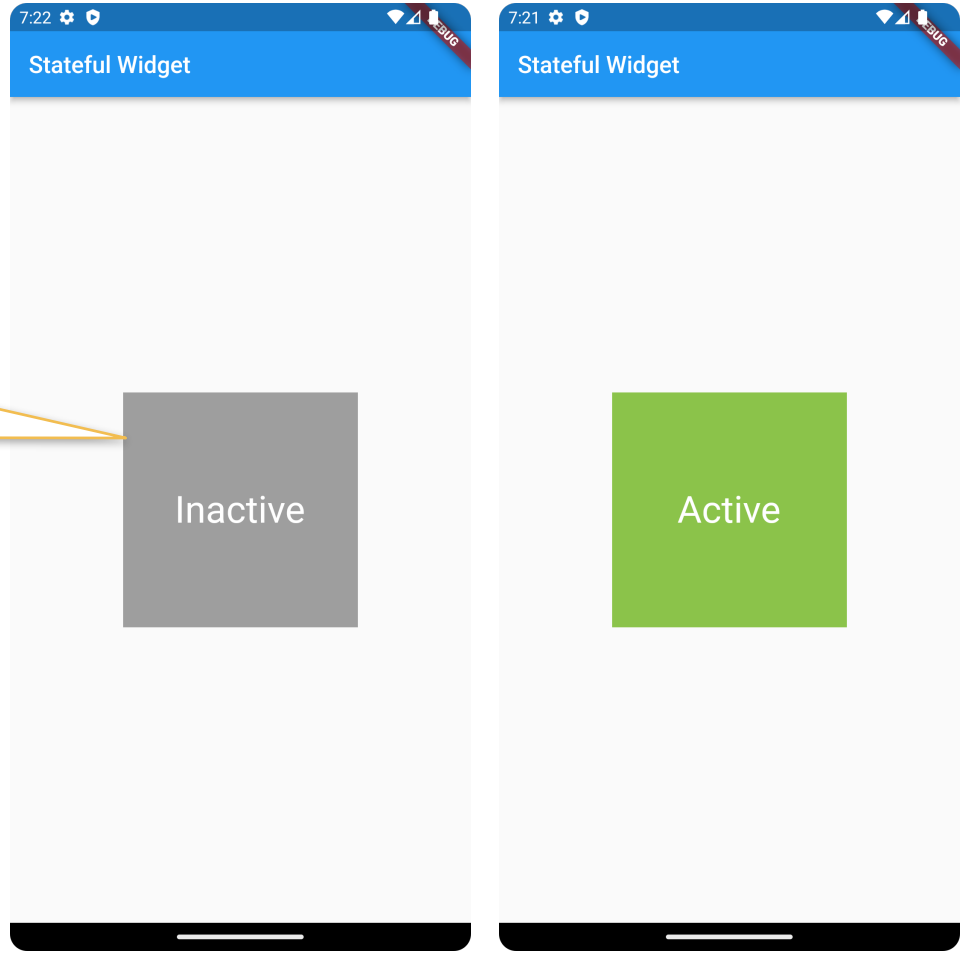


When the user taps a button to increment the counter, the state changes, triggering the widget to rebuild with the updated counter value.

What is State (5)

- "State" refers to the **data values that can change over time**, influencing the appearance and behavior of a widget.

State: Color (Grey or Green) and String (Inactive or Active)



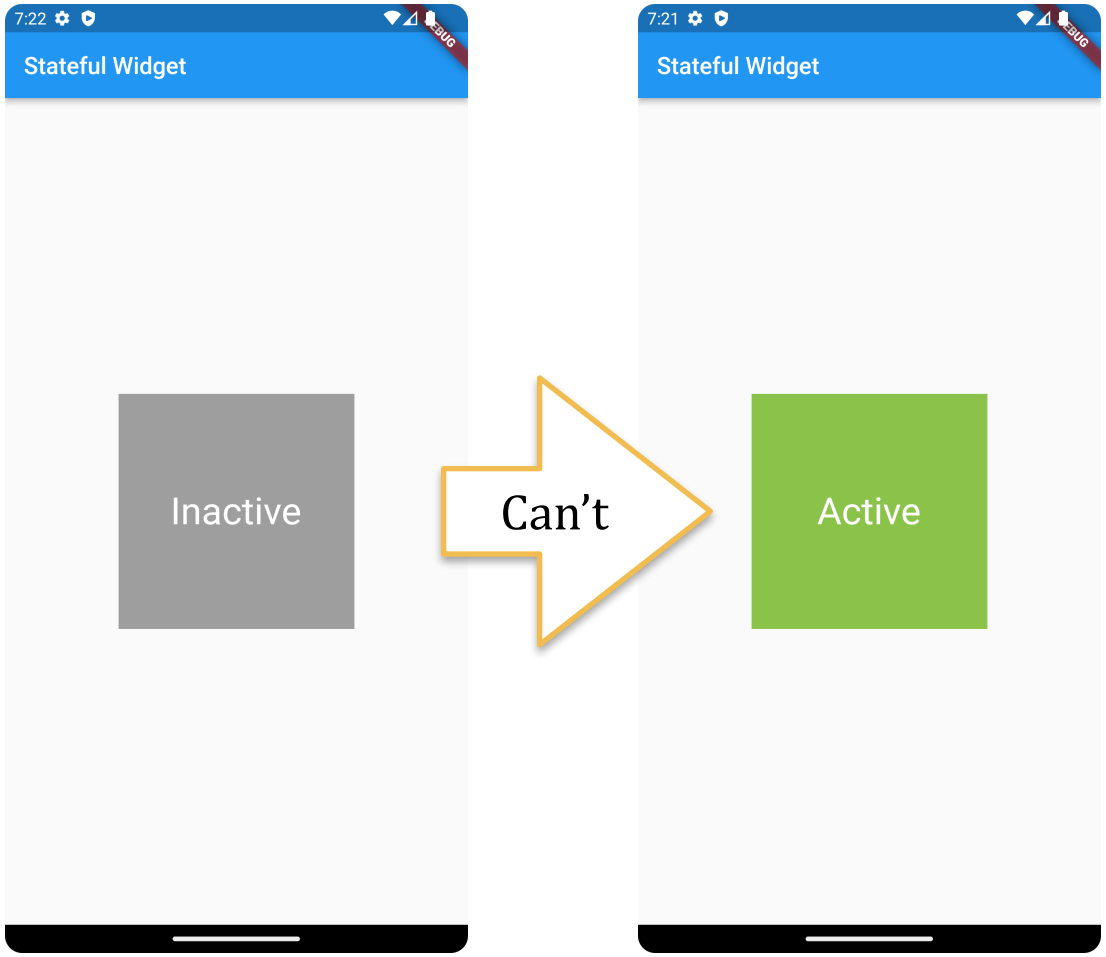
What is State (6)

- There are two types of widgets (Each with a **build** function, but the way they call the build function to update the UI is different)
  1. **StatefulWidget**
  2. **StatelessWidget**

StatelessWidget

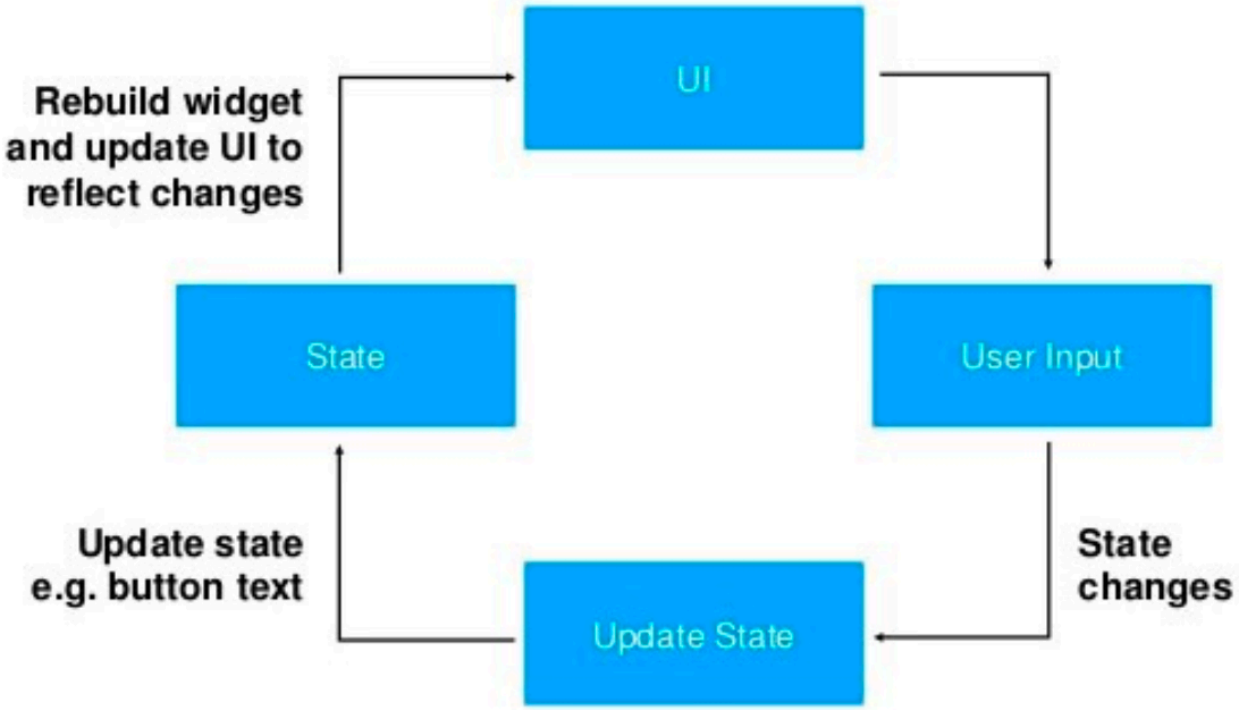
- In the Stateless widget, the **build** function is **called only once** which makes the UI of the screen. It will render(build) the UI at once. So for those widgets, you need to **send the information before** building the widget.
- Below is the structure for a stateless widget:

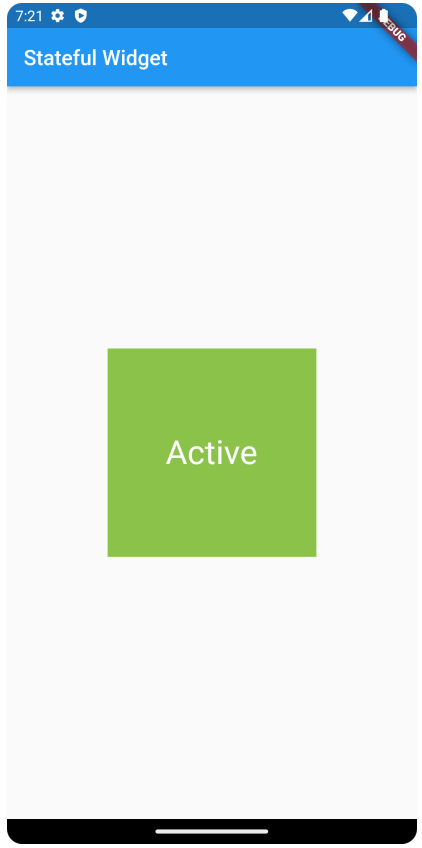
```dart
import 'package:flutter/material.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return Container();
  }
}
```



StatefulWidget (1)

- The widgets whose state can be altered once they are built are called **stateful Widgets**. When the **state changes**, it calls the **build function again** to **rebuild the widget**. The **UI changes**.





StatefulWidget (2)

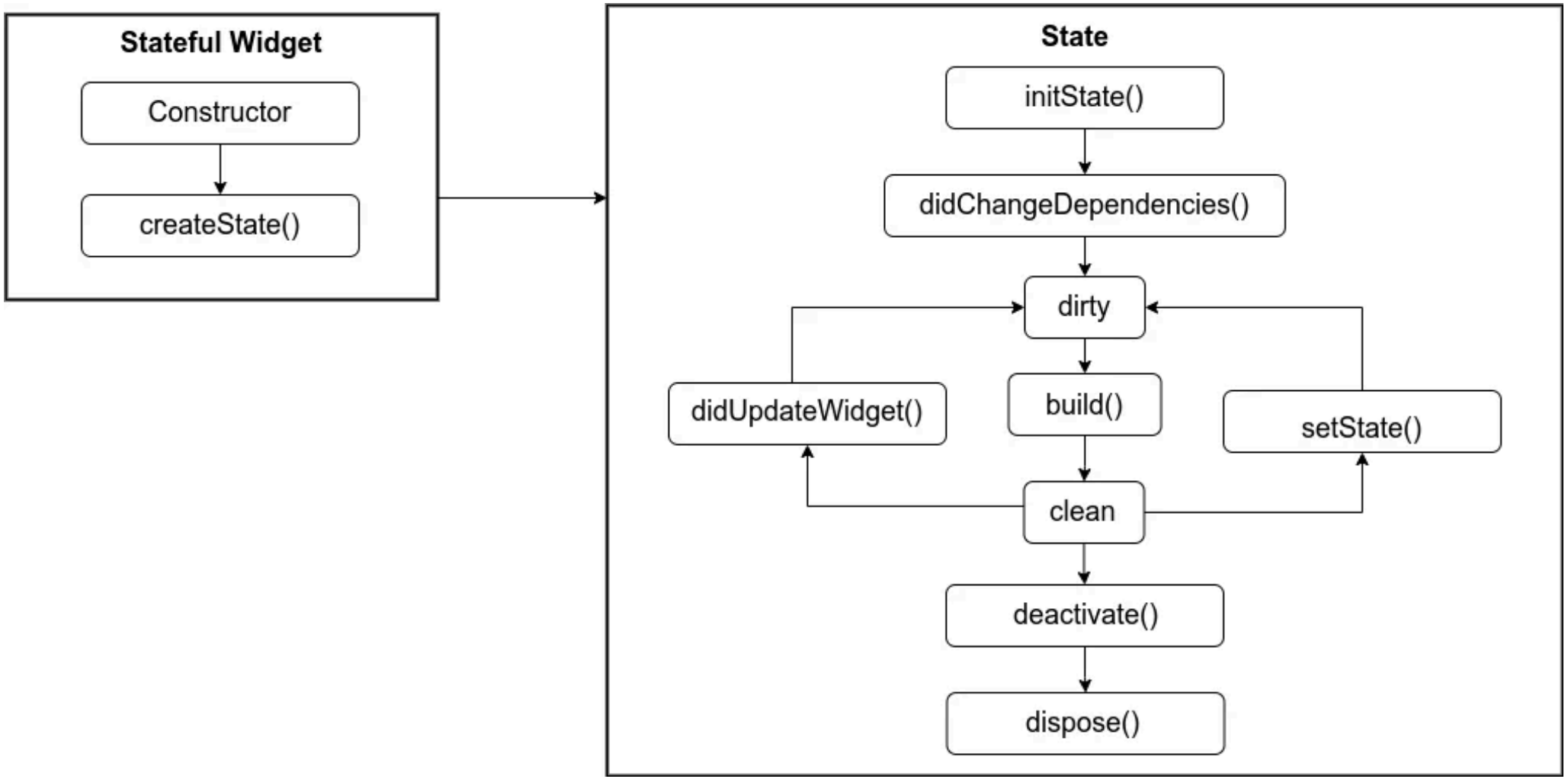
- Structure of Stateful widgets

```dart
class MyApp extends StatefulWidget {
  const MyApp({ Key? key }) : super(key: key);

  @override
  _MyAppState createState() => _MyAppState();
}

class _MyAppState extends State<MyApp> {
  @override
  Widget build(BuildContext context) {
    return Container(

    );
  }
}
```

```dart
void _handleTap() {
  setState(() {
    _active = !_active;
  });
}
```

StatefulWidget (3)

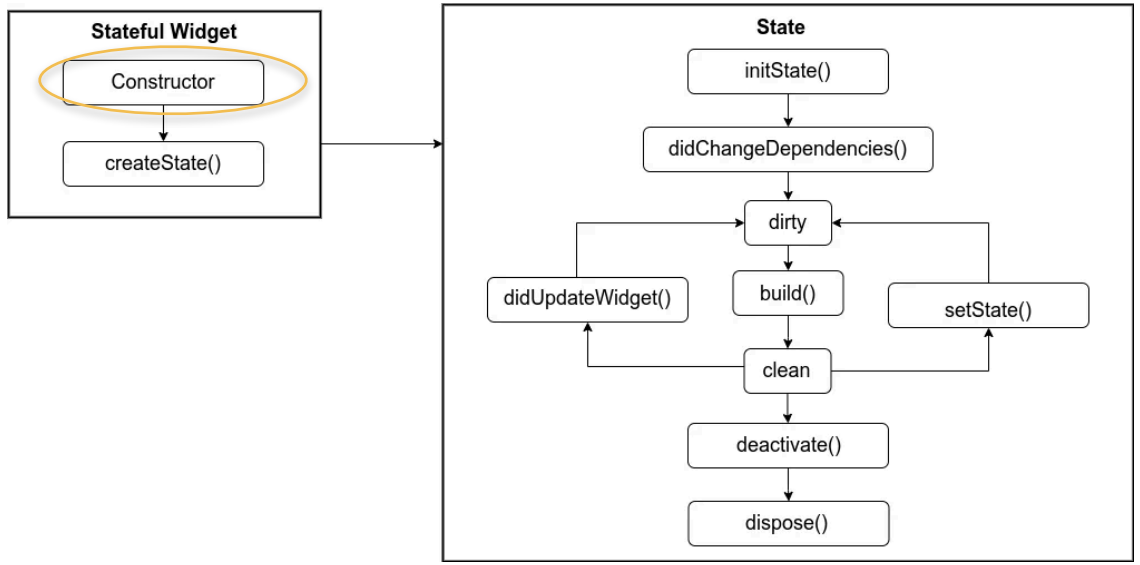
- Lifecycle



StatefulWidget (4)

- **Constructor()**:

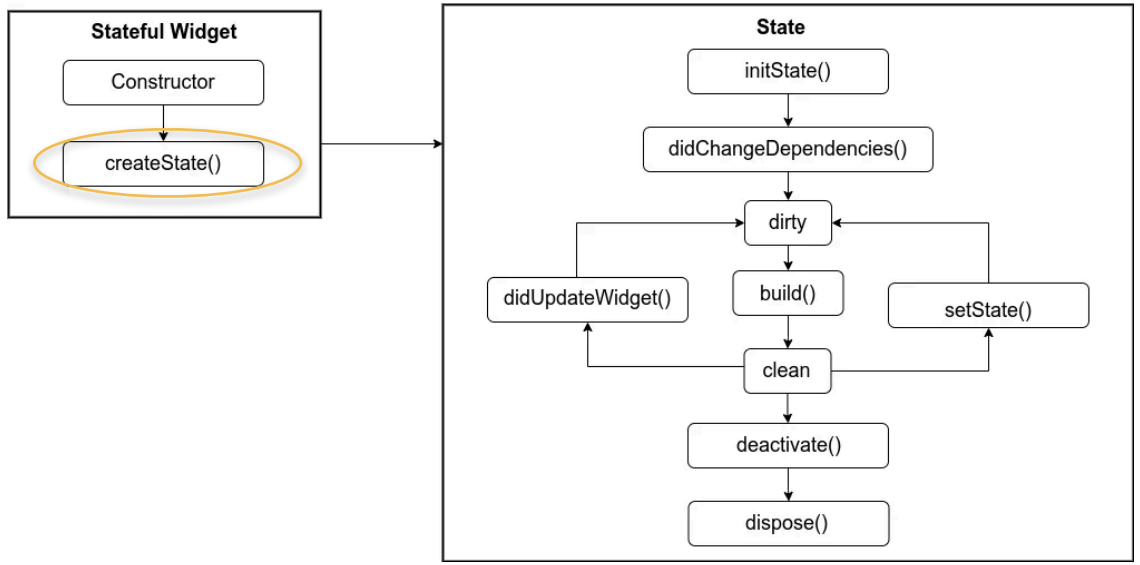
```dart
class StatefulWidgetDemo extends StatefulWidget {
  const StatefulWidgetDemo({super.key});

  @override
  State<StatefulWidgetDemo> createState() => _StatefulWidgetDemoState();
}
```



StatefulWidget (5)

- **createState()**: This method is responsible for creating a **State object**

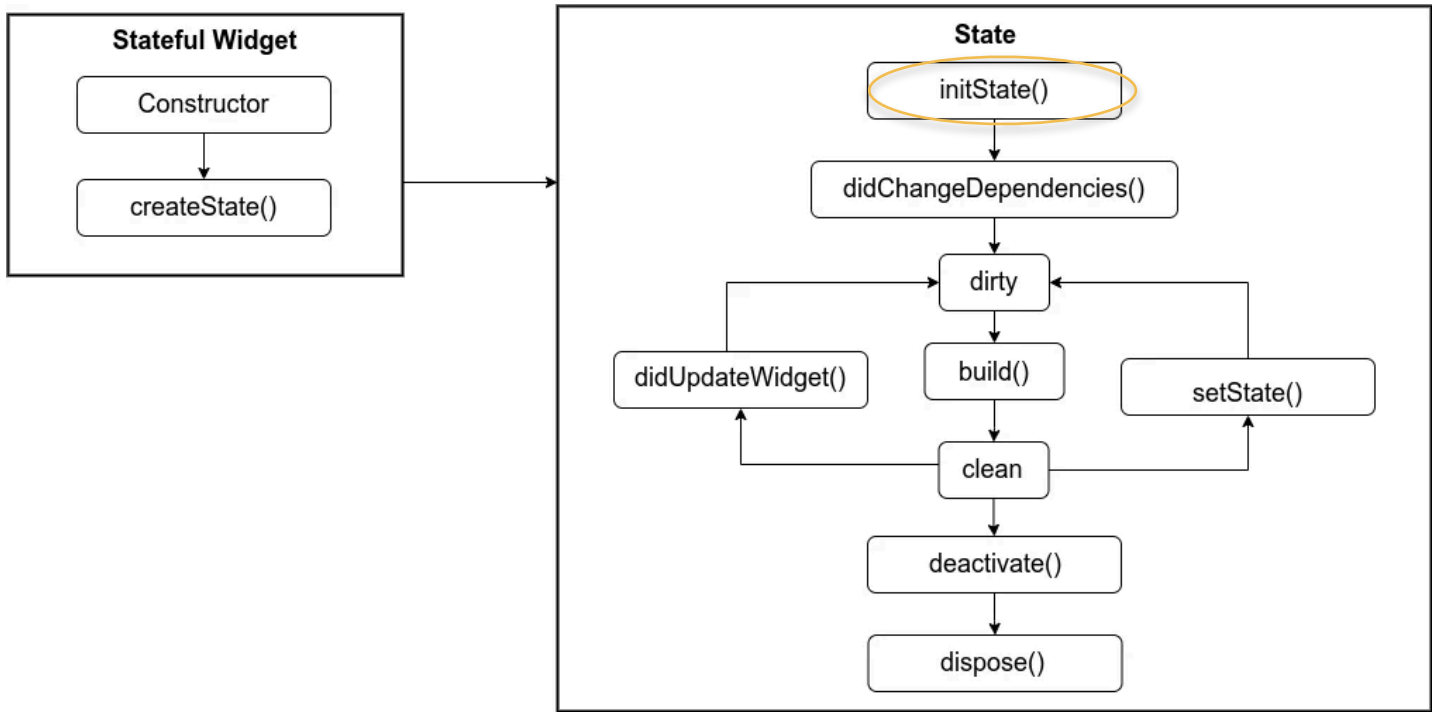
```dart
class StatefulWidgetDemo extends StatefulWidget {
  const StatefulWidgetDemo({super.key});

  @override
  State<StatefulWidgetDemo> createState() => _StatefulWidgetDemoState();
}
```



StatefulWidget (6)

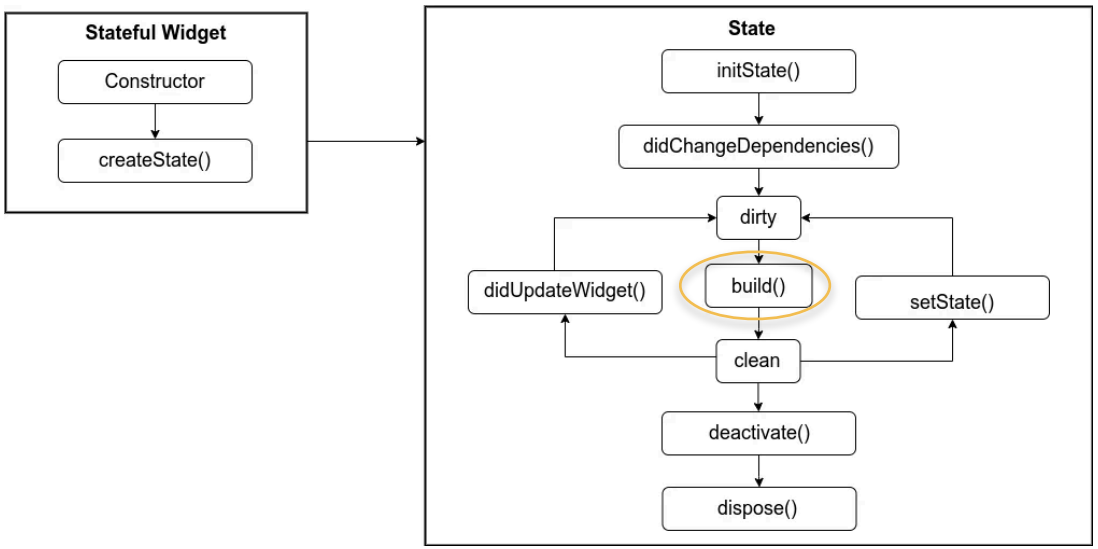
- **initState()**: We can **initialize** variables, data, properties, etc. This method strictly executes **only once.** It also requires to **call the *super.initState()*** method

```dart
@override
void initState() {
  super.initState();

  message = '';
  isPressed = false;
}
```



StatefulWidget (7)

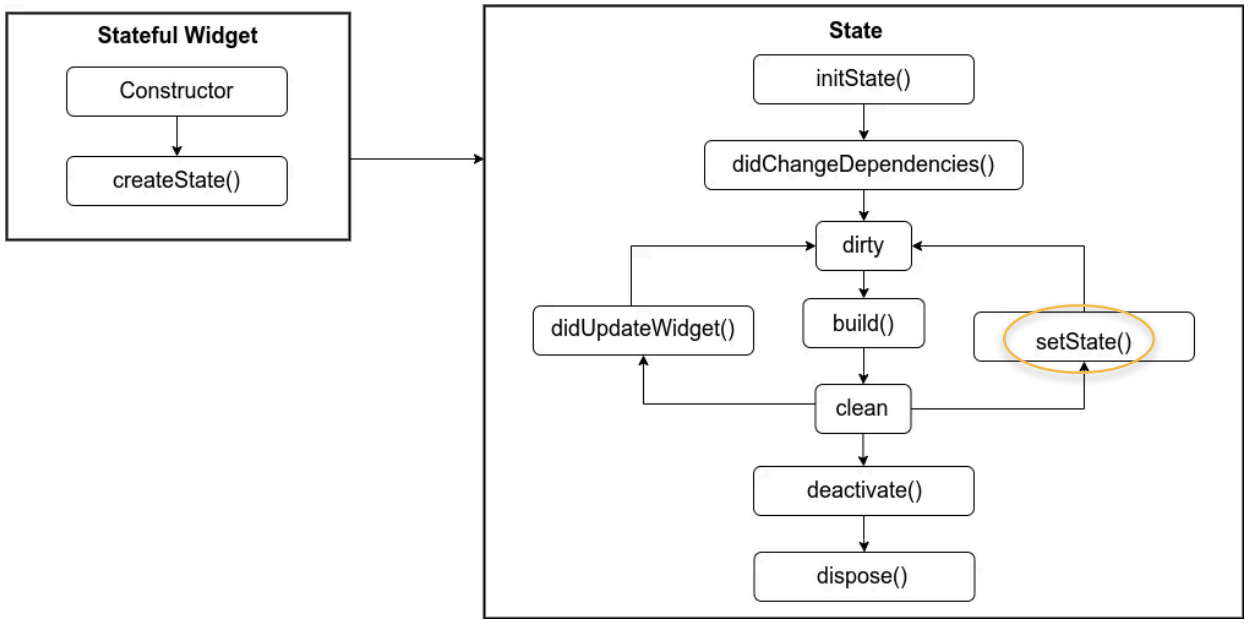
- **build()**: It is the most essential lifecycle method for both a stateless and a stateful widget. It is responsible for **describing** and **rendering** widgets




```dart
@override
Widget build(BuildContext context) {
  return MaterialApp(
    home: Scaffold(
      appBar: AppBar(
        title: const Text('Stateful Widget Demo'),
      ),  // AppBar
      body: Container(
        alignment: Alignment.center,
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: [
            ElevatedButton(
              onPressed: sayHello, child: const Text('Say Hello!')),
            Text(
              message,
              style: const TextStyle(fontSize: 24),
            )  // Text
          ],
        ),  // Column
      ),  // Container
    ),  // Scaffold
  );  // MaterialApp
}
```

StatefulWidget (8)

- **setState()**: **Changes** in the **state object** and the need to rebuild the necessary widget. When calling the ***setState()***, the **build** function is triggered for that state object which in turn **updates the UI**.

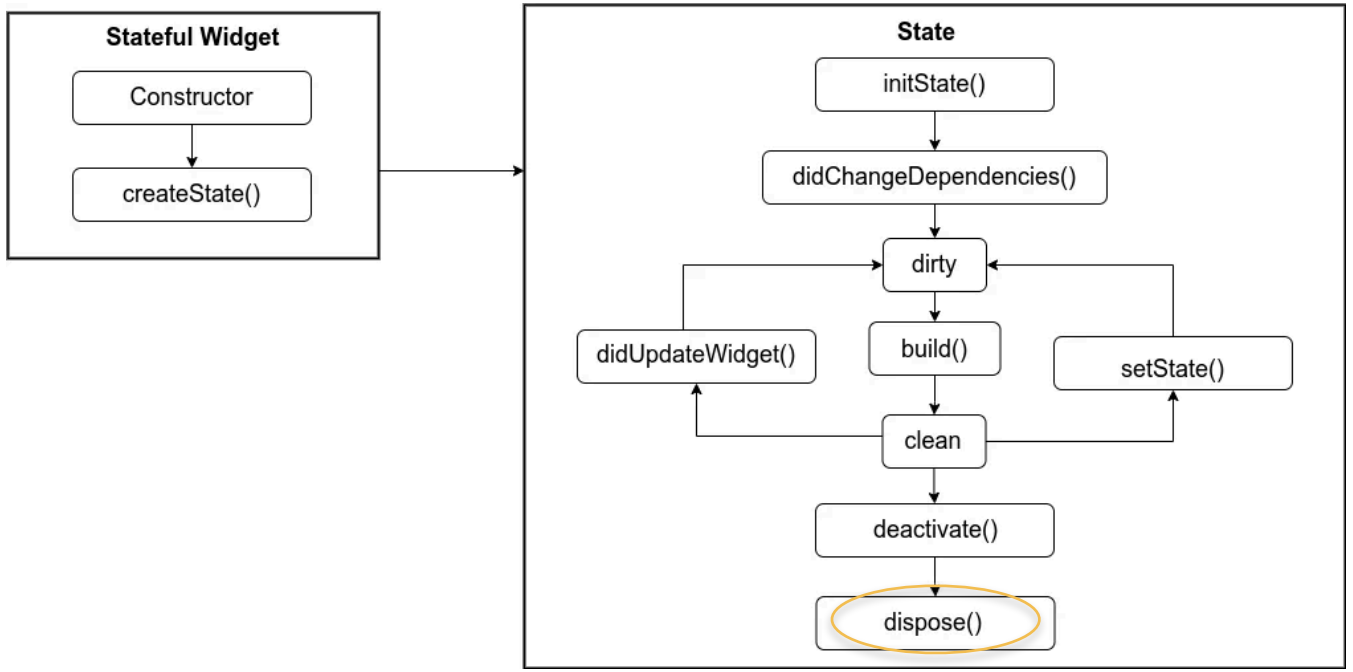



```
void sayHello() {
  if (mounted) {
    setState(() {
      message = 'Hello Flutter';
    });
  }
}
```

StatefulWidget (9)

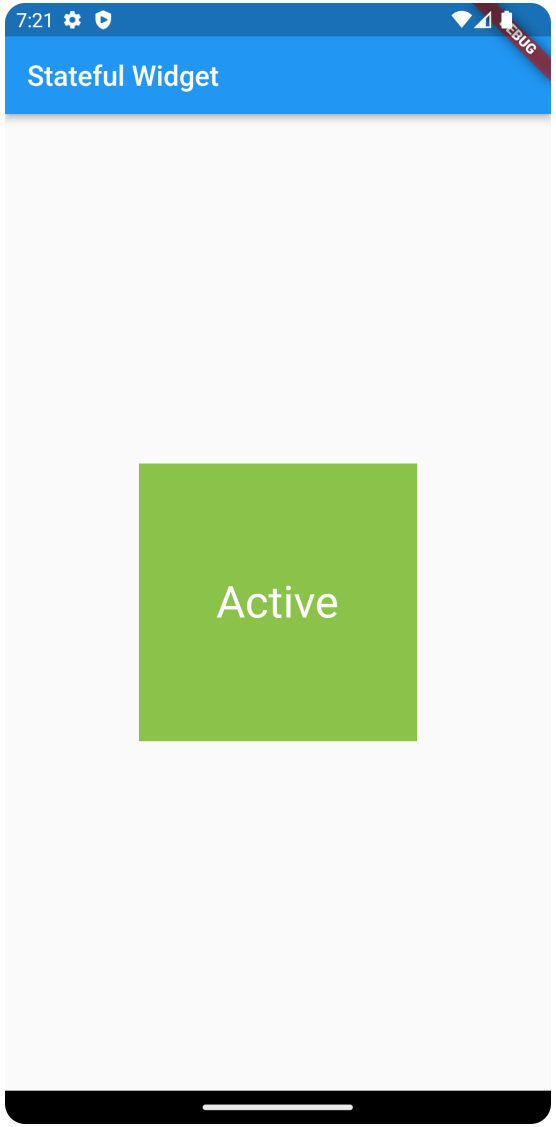
- **dispose()**: The state object's **mounted property** is set to **false** indicating that it will never be built again. Within the *dispose()* method we **release resources** held by the corresponding object.

```
@override
void dispose() {
  super.dispose();
}
```



Demo

```dart
class _TapboxAState extends State<TapboxA> {
  bool _active = false;

  void _handleTap() {
    setState(() {
      _active = !_active;
    });
  }

  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(title: const Text('Stateful Widget'),),
        body: GestureDetector(
          onTap: _handleTap,
          child: Center(
            child: Container(
              width: 200, height: 200,
              decoration: BoxDecoration(
                color: _active ? Colors.lightGreen : Colors.grey,
              ),  // BoxDecoration
              child: Center(
                child: Text(
                    _active ? 'Active' : 'Inactive',
                    style: const TextStyle(fontSize: 32, color: Colors.white),
                  ),  // Text
              ),  // Center
              ),  // Container
            )  // Center
        ),  // GestureDetector
      ),  // Scaffold
    );  // MaterialApp
  }
}
```



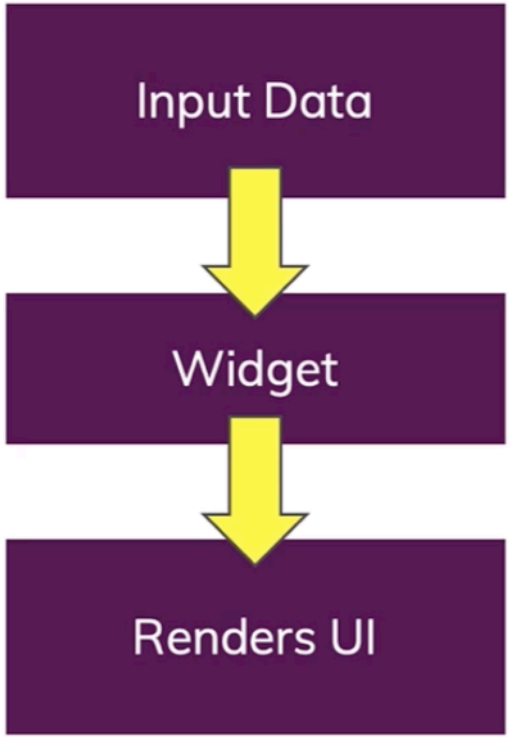
Stateless vs Stateful Widgets

Resource Software Solution

Stateless

Button tapped 0 times

Text

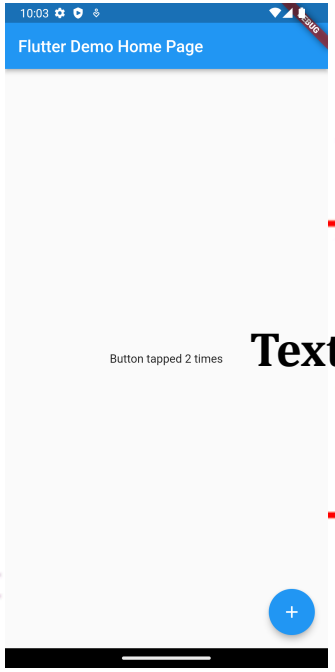


Data can change (externally)

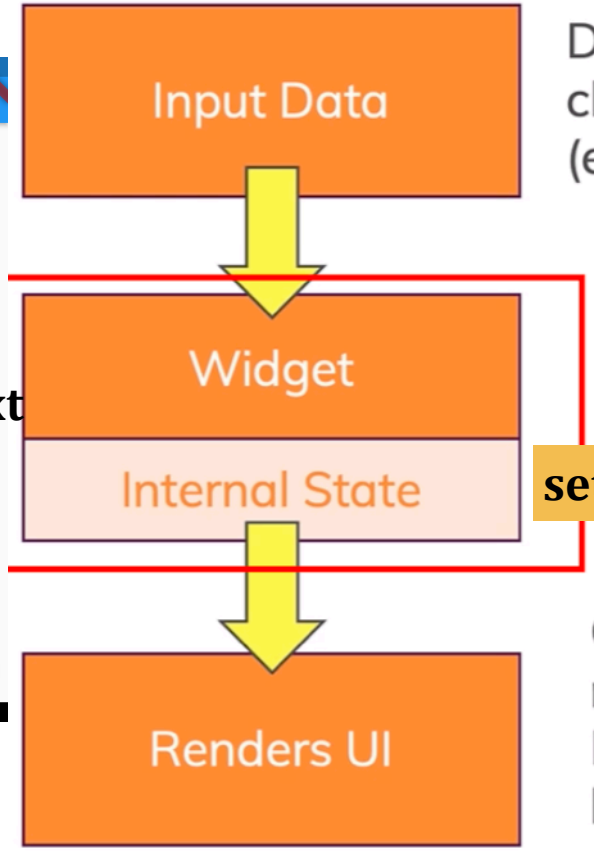
Gets (re)-rendered when Input Data changes

Stateful

Button tapped 0 times



Text



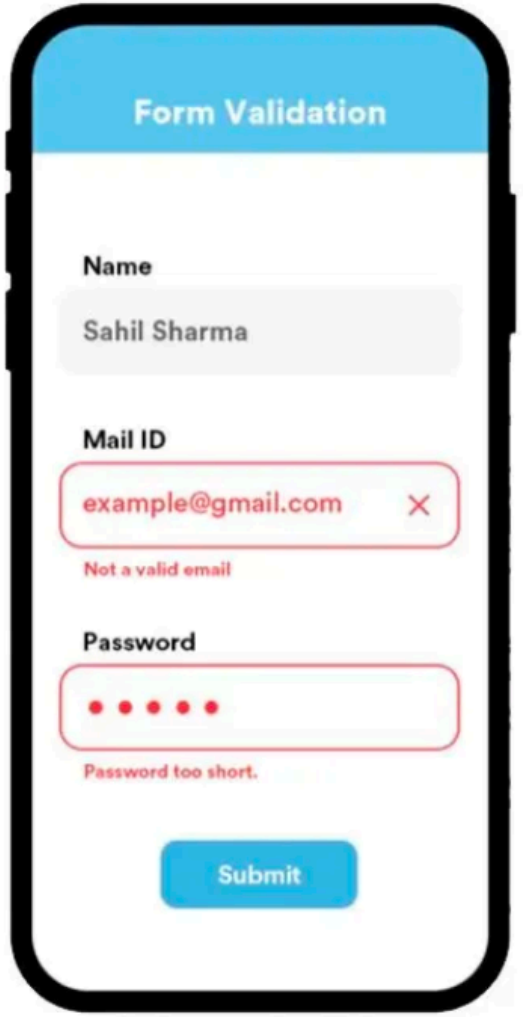
Data can change (externally)

setState() method

Gets (re)-rendered when Input Data or local State changes

Building Forms in Flutter

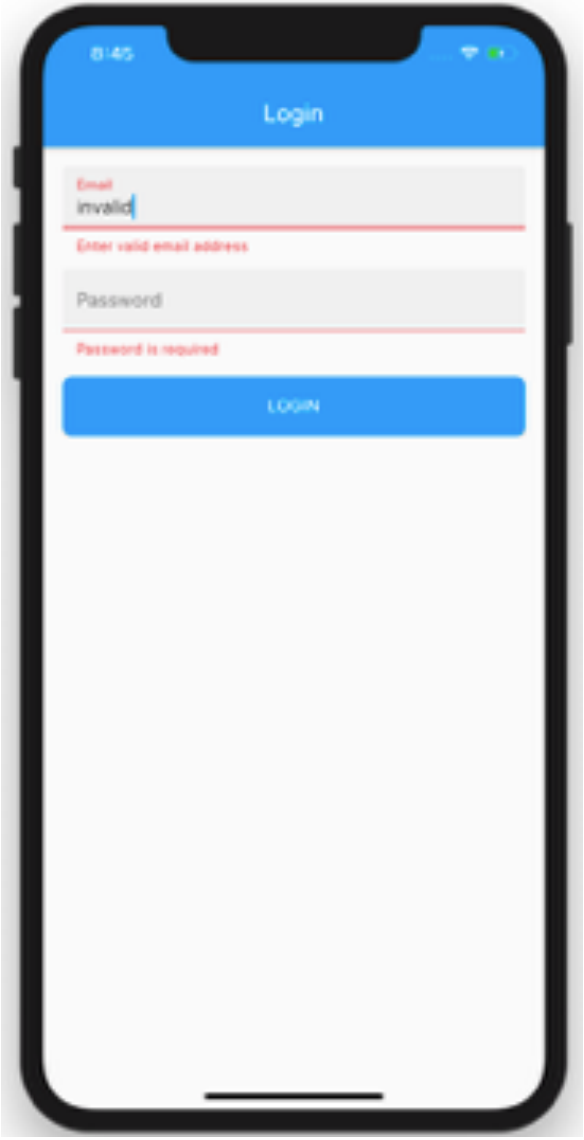
1. Build a form with validation
2. Create style
3. Focus
4. CheckboxListTile class
5. RadioListTile class
6. DropdownButton class



Build a Form with Validation (1)

- Create a Form with a GlobalKey

```
final _formKey = GlobalKey<FormState>();

Form(
  key: _formKey,
  child: null,
)
```

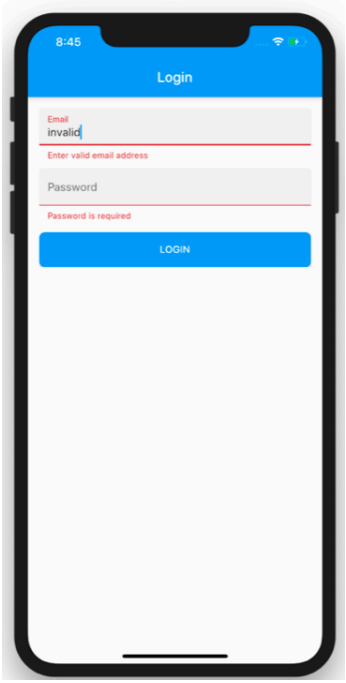


The **_formKey** was set to a global key **holding the form state**, and it's passed to the Form widget as a key. This key will allow the **form to validate** all its descendant input fields.

Build a Form with Validation (2)

- Add a **TextFormField** with validation logic

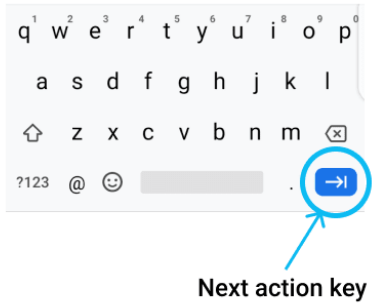
```
final _formKey = GlobalKey<FormState>();

Form(
  key: _formKey,
   child: Column(
     children: <Widget>[
       TextFormField(
         keyboardType: TextInputType.emailAddress,
         textInputAction: TextInputAction.next,
       ),
       TextFormField(
         keyboardType: TextInputType.text,
         textInputAction: TextInputAction.done,
         obscureText: true,
       ),
     ],
   ),
)
```

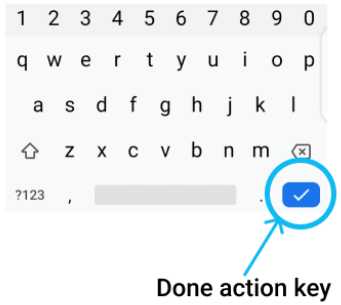


keyboardType: the type of soft keyboard to pop up when the text field gains focus, passing an appropriate text input type value to this property will add to a better user experience.

textInputAction: its value determines the icon to appear as the keyboard's action key.

obscureText: it takes a boolean value (with false as default) that determines whether or not the data supplied by the user in the text field is to be obscured.





Build a Form with Validation (3)

Resource Software Solution

- Form Submission

```
final TextEditingController _emailController = TextEditingController();
final TextEditingController _passwordController = TextEditingController();


_submitForm() {
  if (_formKey.currentState!.validate()) {
    final user = {
     'email': _emailController.text,
      'password': _passwordController.text,
    };

    // If the form passes validation, display a Snackbar.
    Scaffold.of(context).showSnackBar(SnackBar(
      content:Text('Login Successful')));
  }
}
```

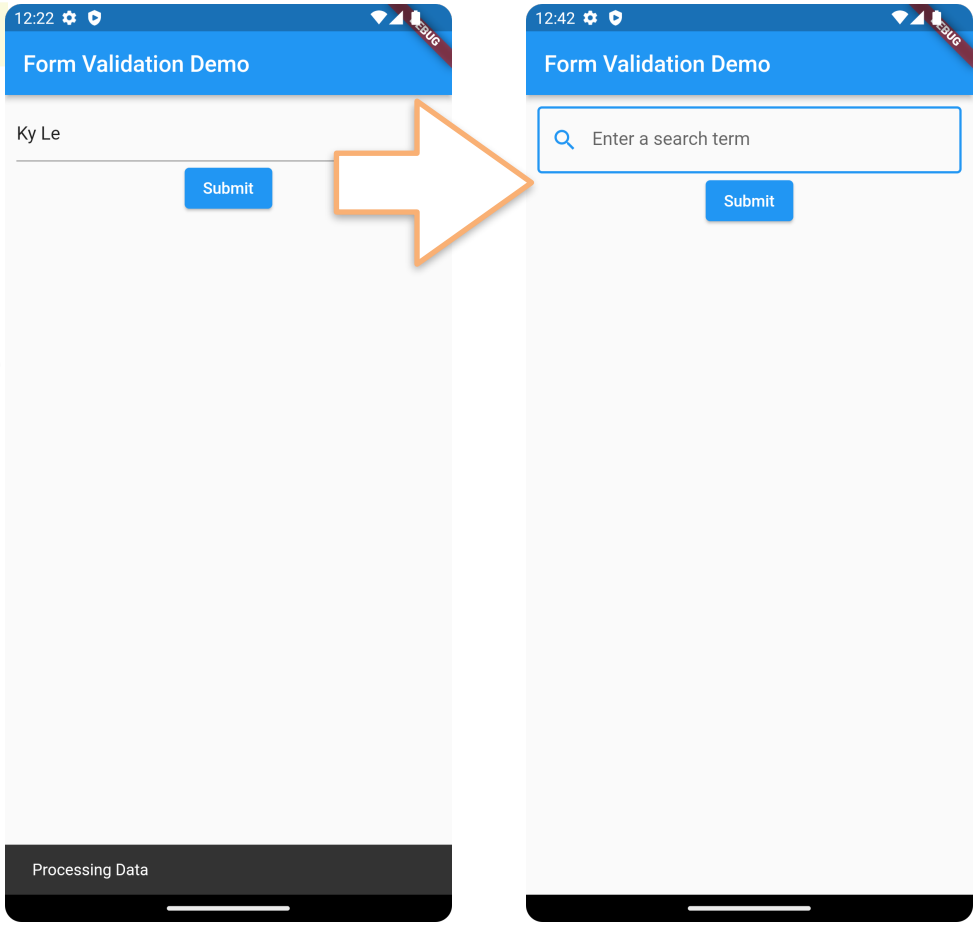
- Non-Empty Validation of Input Fields

```
Form(
  key: _formKey,
  child: Column(
    children: <Widget>[
      TextFormField(
        keyboardType: TextInputType.emailAddress,
        textInputAction: TextInputAction.next,
        controller: _emailController,
        validator: (value) {
                if (value == null || value.isEmpty) {
                  return 'Please enter your email';
                }
                return null;
              }
      ),
       ElevatedButton(
        onPressed: _submitForm,
        child: Text('Login'),
       ),
    ],
  ),
)
```

Create Style

- By default, a TextField is decorated with an underline. You can add a **label**, **icon**, **inline hint text**, and **error text** by supplying an **InputDecoration** as the **decoration** property of the TextField.

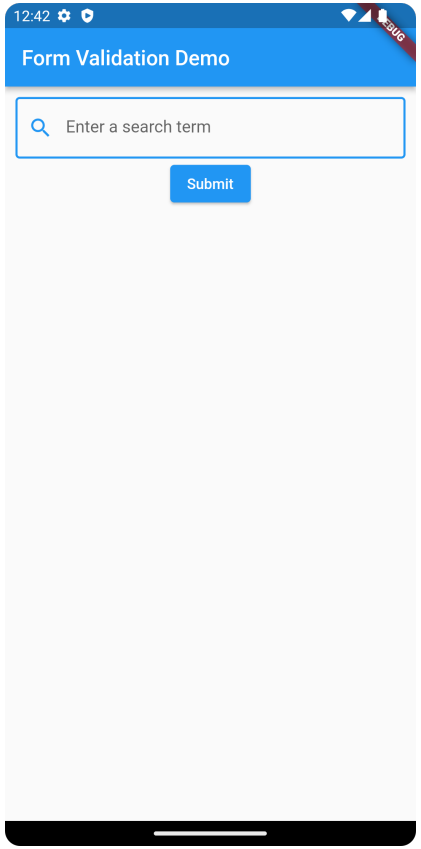
```
TextFormField(
  decoration: const InputDecoration(
      border: OutlineInputBorder(),
      hintText: 'Enter a search term',
      prefixIcon: Icon(Icons.search)),  // InputDecoration
  // The validator receives the text that the user has entered.
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter some text';
    }
    return null;
  },
),  // TextFormField
```



Focus

- When a text field is **selected** and **accepting input**, it is said to have “focus.”
- Focus a text field as soon as it’s visible

```dart
TextFormField(
  autofocus: true,
  decoration: const InputDecoration(
      border: OutlineInputBorder(),
      hintText: 'Enter a search term',
      prefixIcon: Icon(Icons.search)),  // InputDecoration
  // The validator receives the text that the user has entered.
  validator: (value) {
    if (value == null || value.isEmpty) {
      return 'Please enter some text';
    }
    return null;
  },
  controller: myController,
),  // TextFormField
```

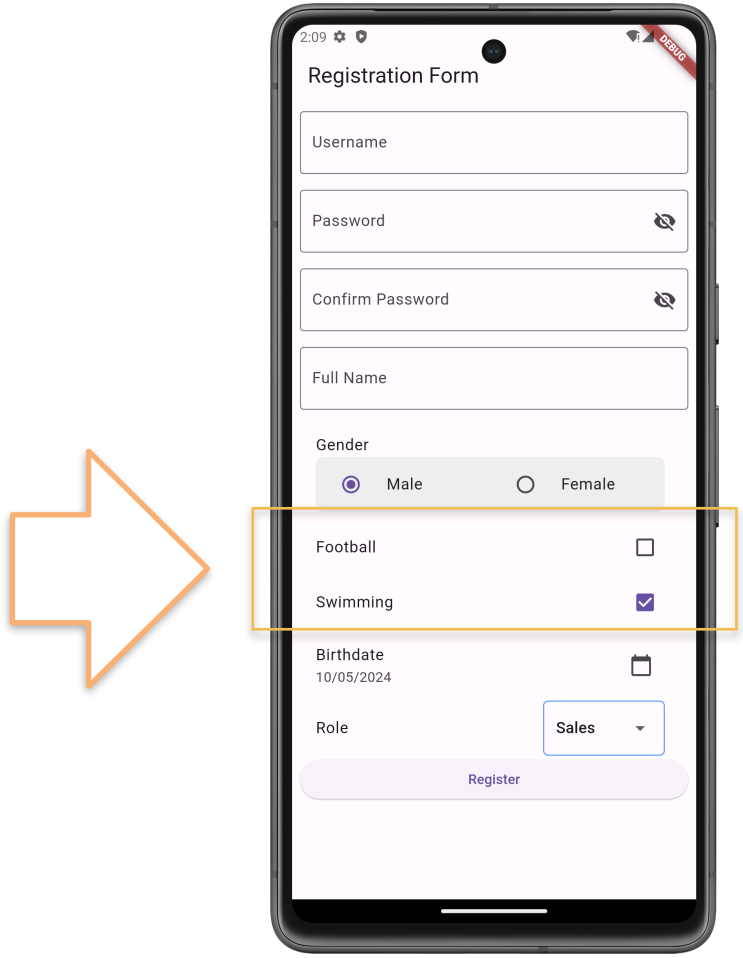


CheckboxListTile class

- A **ListTile** with a **Checkbox.**
- Below is the basic example of how to create a **CheckboxListTile**.

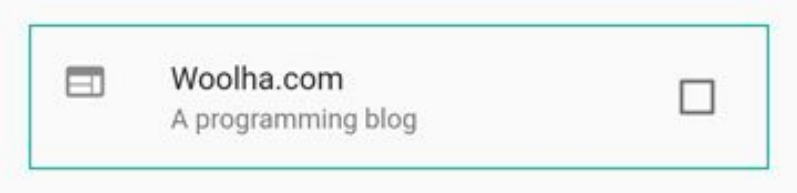
```
bool _isChecked = false;
```

```
final List<String> _hobbies = [];

CheckboxListTile(
        title: const Text('Swimming'),
        value: _hobbies.contains('Swimming'),
        onChanged: (bool? value) {
          setState(() {
            if (value!) {
              _hobbies.add('Swimming');
            } else {
              _hobbies.remove('Swimming');
            }
          });
        },
      )
```



CheckboxListTile class

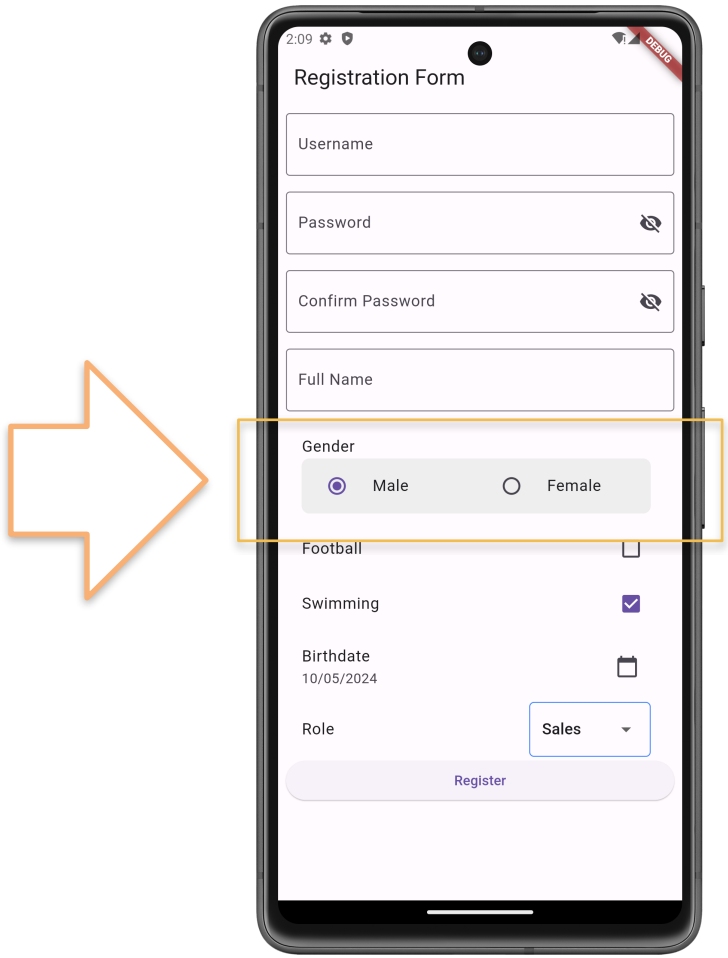
```dart
Container(
  decoration: BoxDecoration(border: Border.all(color: Colors.teal)),
  child: CheckboxListTile(
    title: const Text('Woolha.com'),
    subtitle: const Text('A programming blog'),
    secondary: const Icon(Icons.web),
    value: _isChecked,
    onChanged: (bool value) {
      setState(() {
        _isChecked = value;
      });
    },
  ),
),
```



RadioListTile class

- A **ListTile** with a **Radio.** In other words, a radio button with a label.

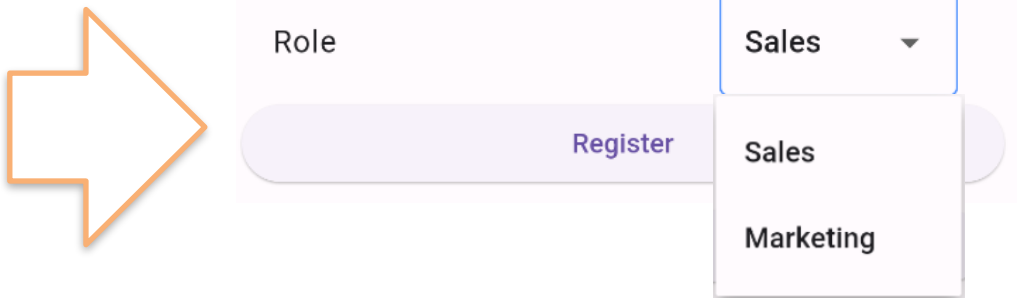
```
String _gender = 'Male';

RadioListTile(
                title: const Text('Male'),
                value: 'male',
                groupValue: _gender,
                onChanged: (value) {
                  setState(() {
                    _gender = value!;
                  });
                },
              ),
RadioListTile(
                title: const Text('Female'),
                value: 'female',
                groupValue: _gender,

              )
```



DropdownButton class

- A dropdown button lets the user select from a number of items.

```
String _role = 'Sales';

DropdownButtonHideUnderline(
            child: DropdownButton<String>(
              value: _role,
              onChanged: (String? newValue) {
                setState(() {
                  _role = newValue!;
                });
              },
              items: <String>['Sales', 'Marketing']
                  .map<DropdownMenuItem<String>>((String value) {
                return DropdownMenuItem<String>(
                  value: value,
                  child: Text(value),
                );
              }).toList(),
            ),
          )
```



Refs: https://www.darttutorial.org/dart-tutorial/dart-map-method/

*Keeping up those **inspiration** and the **enthusiasm** in the **learning path**. Let confidence to bring it into **your career path** for getting gain the **success** as your expectation.*

Thank you

Contact

- **Name: R2S Academy**
- **Email: daotao@r2s.edu.vn**
- **Phone/Zalo: 0919 365 363**
- **FB: https://www.facebook.com/r2s.tuyendung**
- **Website: www.r2s.edu.vn**

Questions and Answers