

# Checkbox

In Flutter, a Checkbox is a widget that lets the user toggle between checked (true) and unchecked (false) states. You control it using the value property and handle changes with the onChanged callback.

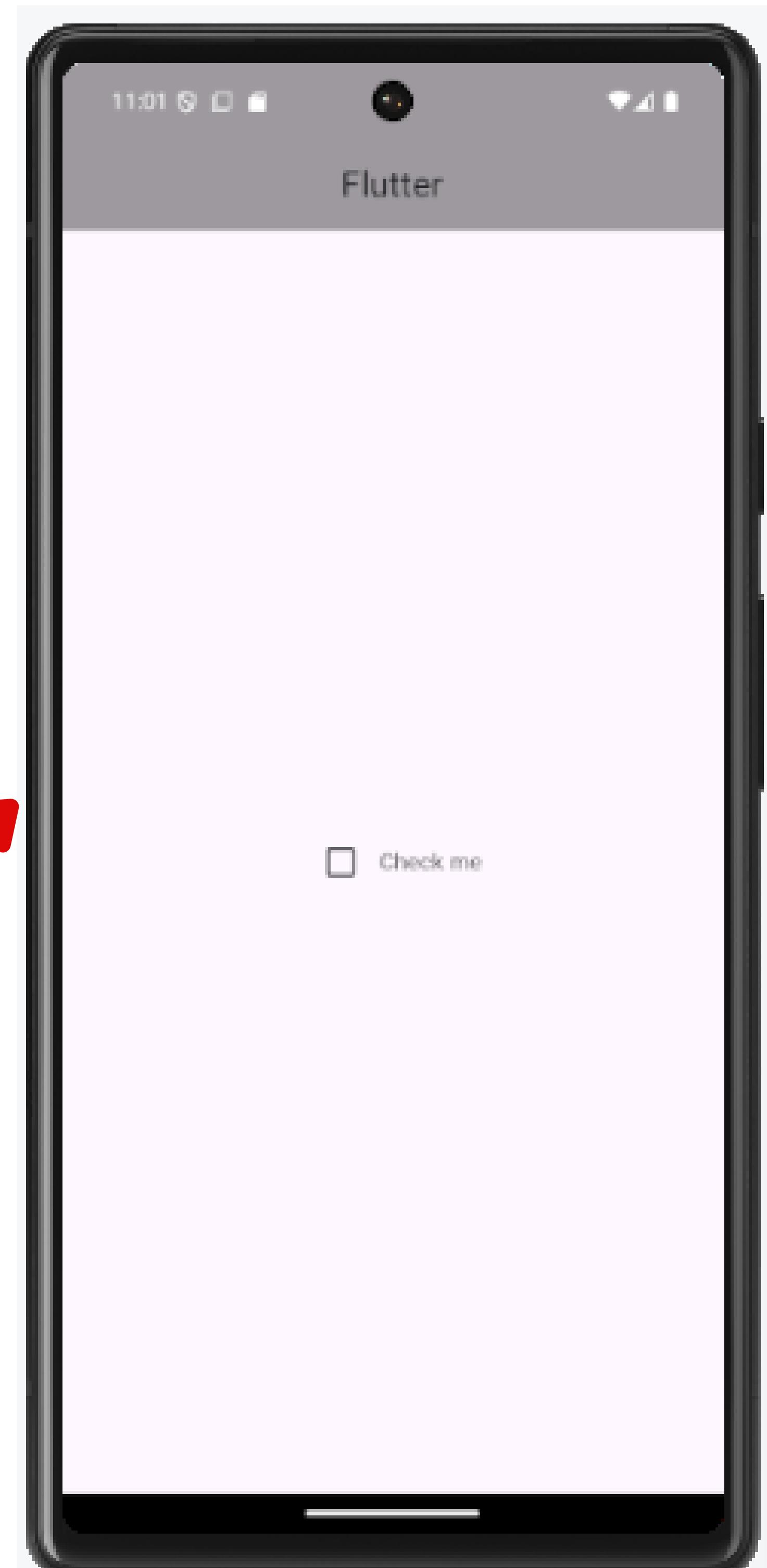
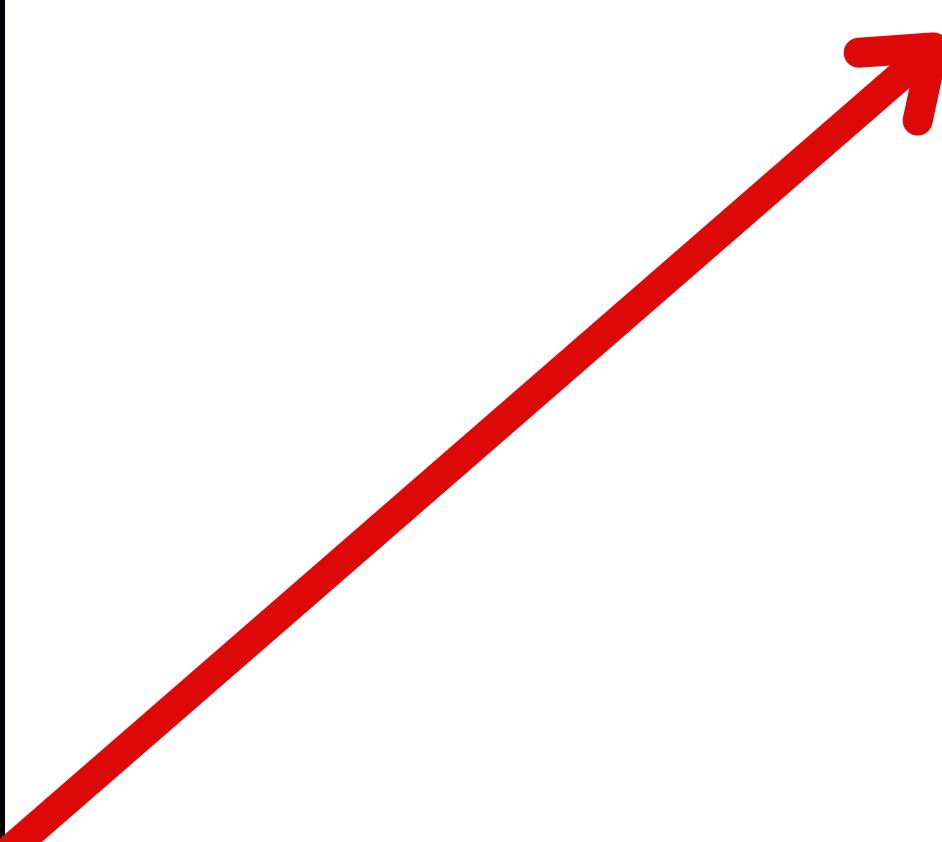
Here, a boolean variable isChecked is initialized to false. This variable will track whether the checkbox is checked or not.



```
class _MyCheckboxState extends State<MyCheckbox> {
  bool isChecked = false;

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

```
  Center(
    child: Row(
      mainAxisAlignment: MainAxisAlignment.center,
      children: [
        Checkbox(
          value: isChecked,
          onChanged: (bool? value) {
            setState(() {
              isChecked = value ?? false;
            });
          },
        ), // Checkbox
        const Text('Check me'),
      ],
    ), // Row
  ), // Center
```



- **Initializes State:** Inside the widget's state, a boolean variable (`isChecked`) is defined to track whether the checkbox is checked or not. Initially, it's set to false (unchecked).
- **Builds the UI:**
  - The app uses a **Scaffold**, which provides a standard app structure.
  - An app bar is added at the top with a title ("Flutter") and a custom background color.
  - The main content (body) is centered on the screen and consists of a horizontal row containing a checkbox and a label ("Check me").
- **Handles User Interaction:** When the checkbox is toggled, the app updates the `isChecked` variable to reflect the new state (checked or unchecked). The `setState` method is called to rebuild the UI with the updated state.
- **UI Updates:** As the state changes (when the checkbox is checked or unchecked), the UI updates automatically to show the current state of the checkbox.

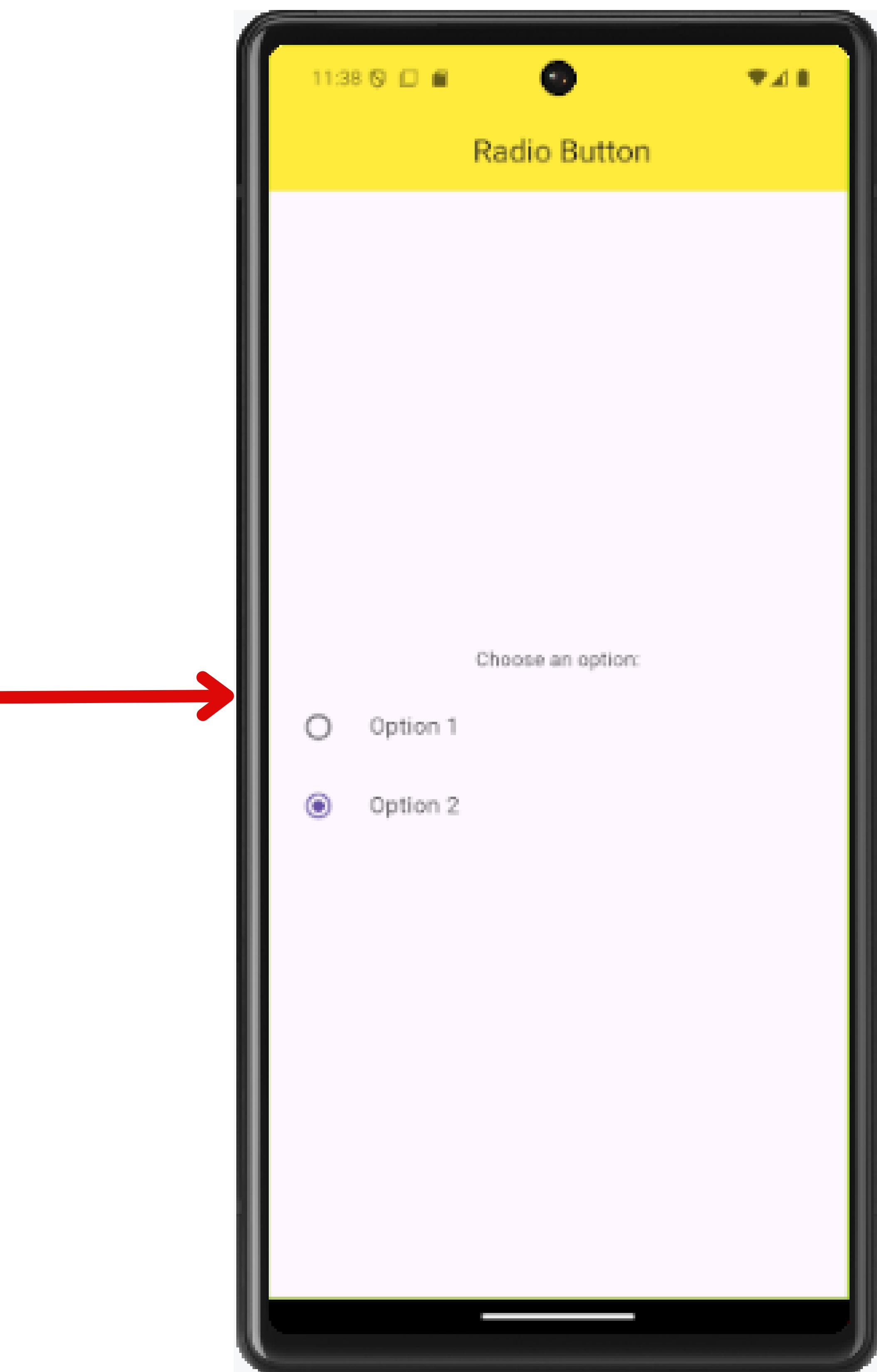
# Radio Button

A Radio Button in Flutter is a UI component that allows users to select one option from a set of choices. Unlike checkboxes, only one radio button in a group can be selected at a time.

This line declares an integer variable `_selectedValue` and initializes it with a value of 1. This variable will store the currently selected value of the radio button. It is used to control which radio button is selected in the UI.

```
class _RadioButtonCenterState extends State<RadioButtonCenter> {
    // Variable to store the selected radio button value
    int _selectedValue = 1;
```

```
@override
Widget build(BuildContext context) {
    return Center(
        child: Column(
            mainAxisSize: MainAxisSize.min,
            children: [
                const Text('Choose an option:'),
                const SizedBox(height: 10),
                RadioListTile<int>(
                    title: const Text('Option 1'),
                    value: 1,
                    groupValue: _selectedValue,
                    onChanged: (int? value) {
                        setState(() {
                            _selectedValue = value!;
                        });
                    },
                ), // RadioListTile
                RadioListTile<int>(
                    title: const Text('Option 2'),
                    value: 2,
                    groupValue: _selectedValue,
                    onChanged: (int? value) {
                        setState(() {
                            _selectedValue = value!;
                        });
                    },
                ), // RadioListTile
            ],
        ),
    );
}
```



- **Build Method:** Describes the layout of the widget.
- **Scaffold:** Provides a basic visual structure for the app.
- **AppBar:** Adds a top bar with a title and styling.
- **Center:** Aligns its child widget in the middle of the screen.
- **Column:** Arranges its child widgets vertically.
- **Text:** Displays a prompt above the radio buttons.
- **SizedBox:** Adds vertical space between the prompt and the radio buttons.
- **RadioListTile:** Creates a radio button with a label.
- **Value:** Assigns a unique identifier to each radio button.
- **GroupValue:** Links radio buttons to the selected value.
- **OnChanged:** Updates the selected value when a radio button is clicked.
- **Completion:** Ends the layout definitions for the Column, Center, and Scaffold.

# Slider

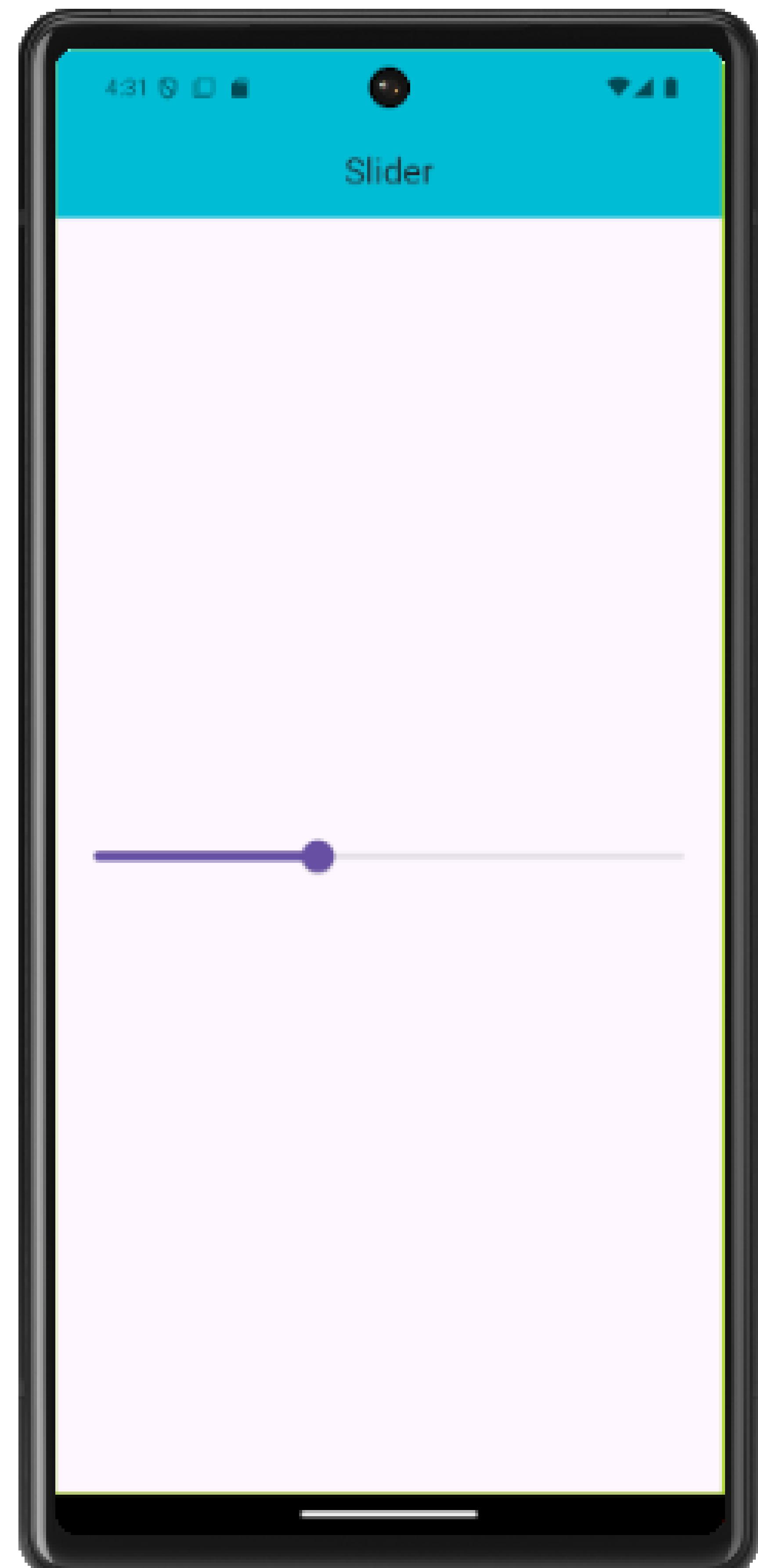
A Slider in Flutter is a widget that lets users pick a value by sliding a thumb across a line. It's great for things like adjusting volume or brightness. You can set the minimum and maximum values, and the slider moves between them.

\_sliderValue is a private variable that holds the current value of the slider, initialized to 10.0. The underscore \_ before the variable name makes it private, meaning it can only be accessed within this class.



```
class _SliderDemoState extends State<SliderDemo> {  
  // This variable holds the current value of the slider  
  double _sliderValue = 10.0;  
  
  @override  
  Widget build(BuildContext context) {
```

```
    Slider(  
      value: _sliderValue,  
      min: 0.0,  
      max: 100.0,  
      divisions: 100,  
      label: _sliderValue.toStringAsFixed(0),  
      onChanged: (double newValue) {  
        setState(() {  
          _sliderValue = newValue;  
        });  
      },  
    ), // Slider
```

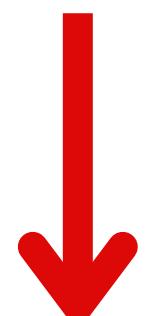


- **Slider**: A widget that allows the user to select a value from a range by sliding a thumb along a track.
- **value: \_sliderValue**: Sets the current value of the slider to the value of `_sliderValue`.
- **min: 0.0 and max: 100.0**: Define the minimum and maximum values the slider can have.
- **divisions: 100**: Divides the slider into 100 discrete intervals.
- **label: `_sliderValue.toStringAsFixed(0)`**: Displays the current value of the slider as a label, formatted to 0 decimal places.
- **onChanged: (double newValue) {}**: This is a callback function that gets triggered whenever the slider value changes. Inside this function, `setState()` is called to update the `_sliderValue` with the new value (`newValue`). `setState()` tells Flutter to rebuild the widget to reflect the updated state.

# Dropdown button

A **DropdownButton** in Flutter is a widget that allows users to select an item from a list of options presented in a dropdown menu.

`_selectedItem`: This is a private variable that holds the currently selected item from the dropdown. It is initialized to 'Item 1'. The underscore `_` before the variable name makes it private to this class.



```
class DropdownButtonExampleState extends State<DropdownButtonExample> {
  String? _selectedItem = 'Item 1'; // Initial selected value

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

```
DropdownButton(
  value: _selectedItem, // Currently selected value
  items: ['Item 1', 'Item 2', 'Item 3', 'Item 4']
    .map<DropdownMenuItem<String>>((String value) {
  return DropdownMenuItem<String>(
    value: value,
    child: Text(value),
  ); // DropdownMenuItem
}).toList(),
onChanged: (String? newValue) {
  setState(() {
    _selectedItem = newValue; // Update the selected value
  });
},
), // DropdownButton
```



- **DropdownButton**: A widget that displays a dropdown menu with a list of items.
- **value: \_selectedItem**: Sets the currently selected value of the dropdown to `_selectedItem`.
- **items**: This is a list of `DropdownMenuItem<String>` widgets created from a list of strings (`['Item 1', 'Item 2', 'Item 3', 'Item 4']`). The map function converts each string into a `DropdownMenuItem<String>` with a value and a `Text` widget displaying the string.
- **onChanged**: This callback is triggered when the user selects a new item from the dropdown. It calls `setState()` to update `_selectedItem` with the newly selected value, causing the widget to rebuild and reflect the updated selection.