ThemeData and ColorScheme

- Easy coloring your app by defining a ColorScheme with seed color
- Override some colors in the ColorScheme
- Set properties for all sliders or switches or buttons in an app
- Enable your app to use light or dark mode

ThemeData and ColorScheme

Till now we used:

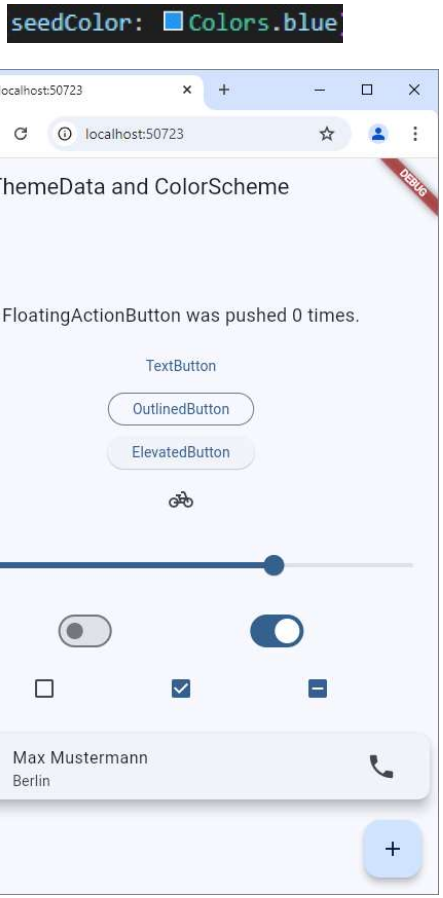
```
@override
Widget build(BuildContext context) {
  return const MaterialApp(
    home: Scaffold(
```

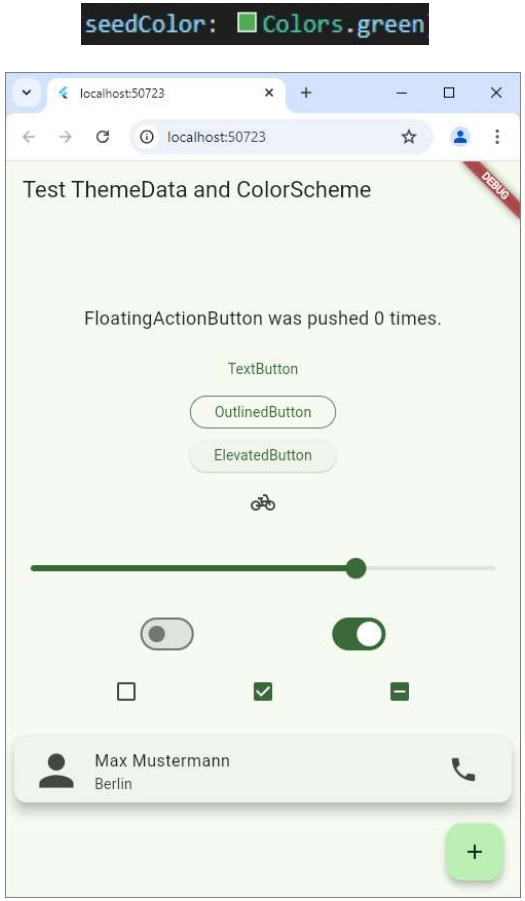
For easy color management we can use:

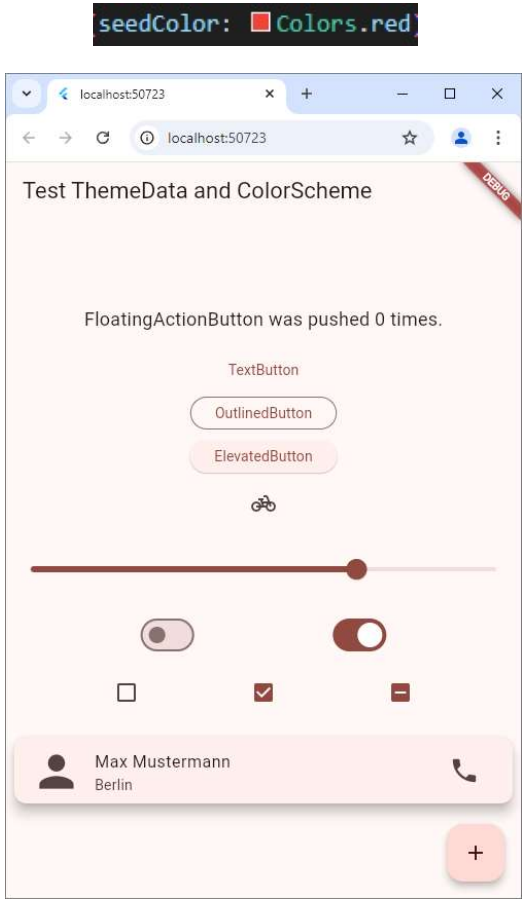
```
@override
Widget build(BuildContext context) {
  return MaterialApp(
    theme:
        ThemeData(colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue)),
    home: Scaffold(
```

```
(new) ColorScheme ColorScheme.fromSeed({
  required Color seedColor,
  Brightness brightness = Brightness.light,
  DynamicSchemeVariant dynamicSchemeVariant = Dy
  double contrastLevel = 0.0,
  Color? primary,
  Color? onPrimary,
  Color? primaryContainer,
  Color? onPrimaryContainer,
  Color? primaryFixed,
  Color? primaryFixedDim,
  Color? onPrimaryFixed,
  Color? onPrimaryFixedVariant,
  Color? secondary,
  Color? onSecondary,
  Color? secondaryContainer,
  Color? onSecondaryContainer,
  Color? secondaryFixed,
  Color? secondaryFixedDim,
  Color? onSecondaryFixed,
  Color? onSecondaryFixedVariant,
  Color? tertiary,
  Color? onTertiary,
  Color? tertiaryContainer,
  Color? onTertiaryContainer,
```

Seed Colors

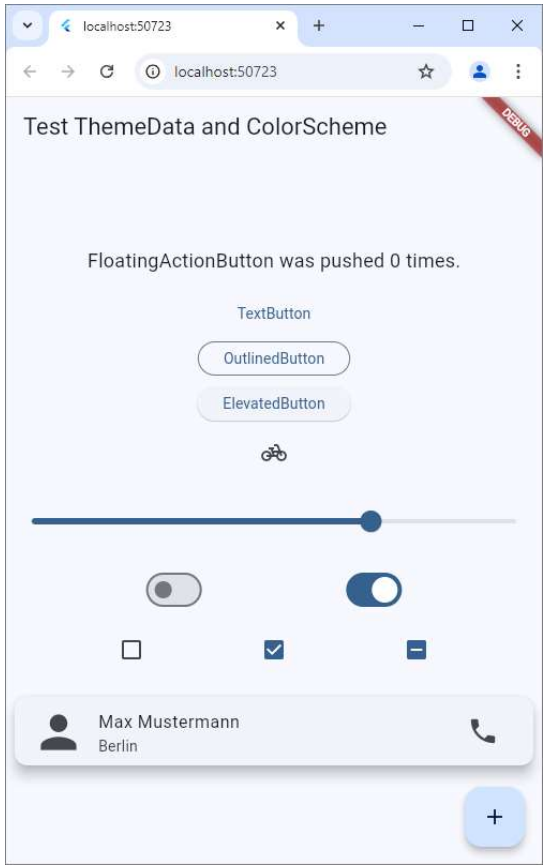

seedColor: Colors.blue
localhost:50723
localhost:50723
Test ThemeData and ColorScheme
DEBUG
FloatingActionButton was pushed 0 times.
TextButton
OutlinedButton
ElevatedButton
Max Mustermann
Berlin


seedColor: Colors.green
localhost:50723
localhost:50723
Test ThemeData and ColorScheme
DEBUG
FloatingActionButton was pushed 0 times.
TextButton
OutlinedButton
ElevatedButton
Max Mustermann
Berlin

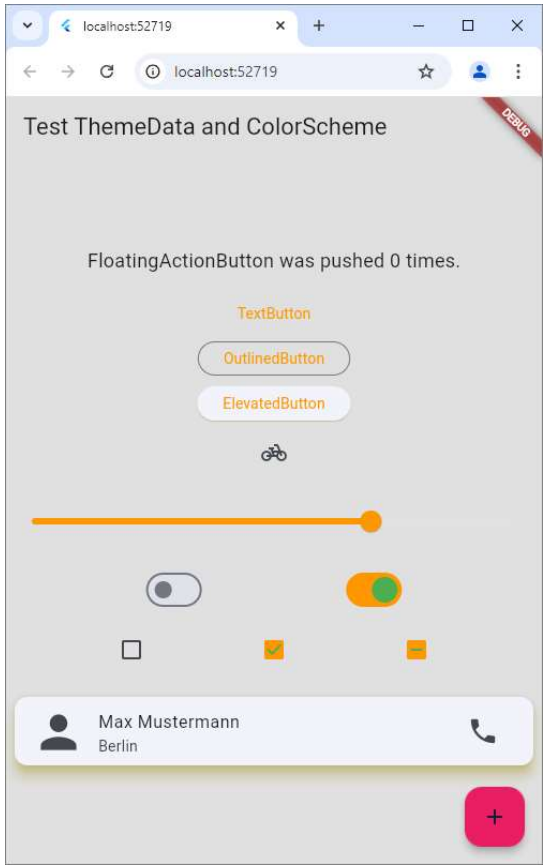

seedColor: Colors.red
localhost:50723
localhost:50723
Test ThemeData and ColorScheme
DEBUG
FloatingActionButton was pushed 0 times.
TextButton
OutlinedButton
ElevatedButton
Max Mustermann
Berlin

Override certain colors in the ColorScheme

```
colorScheme: ColorScheme.fromSeed(seedColor: Colors.blue),
```




```
colorScheme: ColorScheme.fromSeed(
  seedColor: Colors.blue,
  primary: Colors.orange,
  primaryContainer: Colors.pink,
  onPrimary: Colors.green,
  surface: Colors.grey.shade300,
  shadow: Colors.yellow,
), // ColorScheme.fromSeed
```

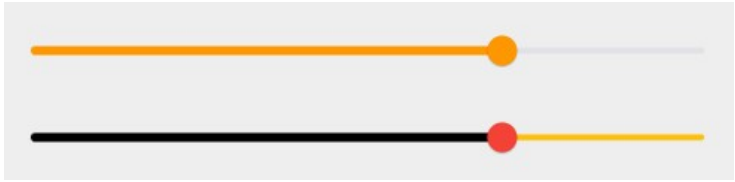


Explicitly defined colors override ColorScheme

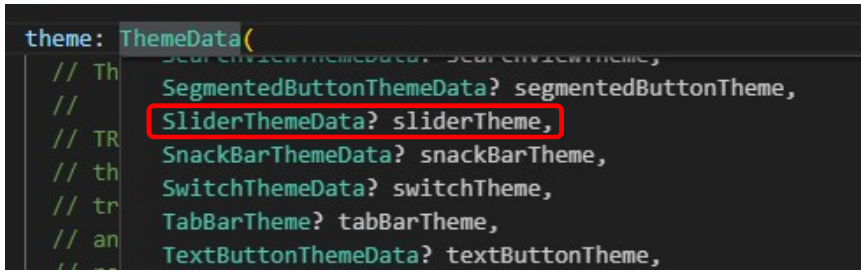
```
Slider(
    value: sliderValue,
    onChanged: (value) {
      setState(() {
        sliderValue = value;
      });
    }), // Slider

const SizedBox(height: 10),

Slider(
    activeColor: Colors.black,
    thumbColor: Colors.red,
    inactiveColor: Colors.amber,
    value: sliderValue,
    onChanged: (value) {
      setState(() {
        sliderValue = value;
      });
    }), // Slider
```



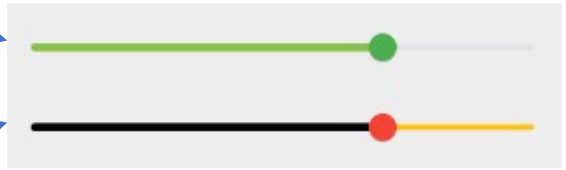
Set properties for all sliders in the app



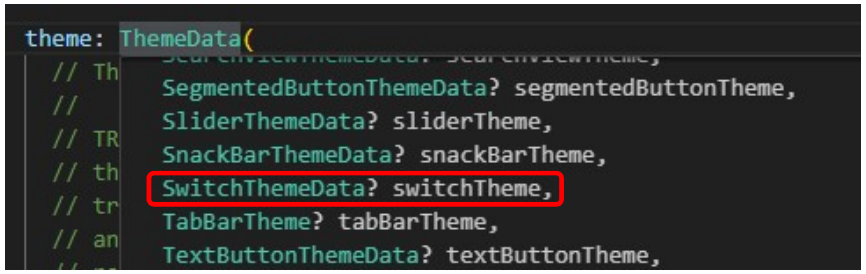

```
SliderThemeData SliderThemeData({
  double? trackHeight,
  Color? activeTrackColor,
  Color? inactiveTrackColor,
  Color? secondaryActiveTrackColor,
  Color? disabledActiveTrackColor,
  Color? disabledInactiveTrackColor,
  Color? disabledSecondaryActiveTrackColor,
  Color? activeTickMarkColor,
  Color? inactiveTickMarkColor,
  Color? disabledActiveTickMarkColor,
  Color? disabledInactiveTickMarkColor,
  Color? thumbColor,
```

```
theme: ThemeData(
  sliderTheme: const SliderThemeData(
      activeTrackColor: Colors.lightGreen,
      thumbColor: Colors.green), // SliderThemeData
  colorScheme: ColorScheme.fromSeed(
```

```
Slider(
    value: sliderValue,
    onChanged: (value) {
      setState(() {
        sliderValue = value;
      });
    }), // Slider
const SizedBox(height: 10),
Slider(
    activeColor: Colors.black,
    thumbColor: Colors.red,
    inactiveColor: Colors.amber,
    value: sliderValue,
```

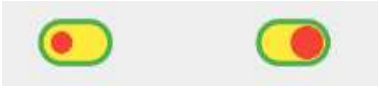


Same for the switches



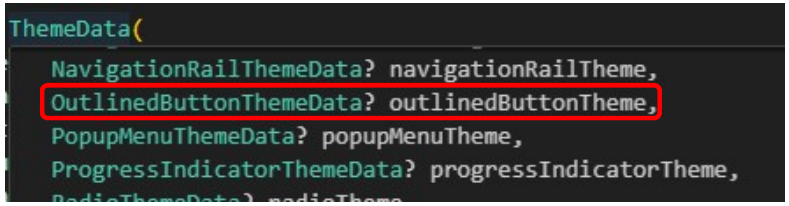

```
SwitchThemeData SwitchThemeData({
  WidgetStateProperty<Color?>? thumbColor,
  WidgetStateProperty<Color?>? trackColor,
  WidgetStateProperty<Color?>? trackOutlineColor,
  WidgetStateProperty<double?>? trackOutlineWidth,
  MaterialTapTargetSize? materialTapTargetSize,
  WidgetStateProperty<MouseCursor?>? mouseCursor,
  WidgetStateProperty<Color?>? overlayColor,
  double? splashRadius,
  WidgetStateProperty<Icon?>? thumbIcon,
})
```

```
theme: ThemeData(
  switchTheme: const SwitchThemeData(
    thumbColor: WidgetStatePropertyAll(Colors.red),
    trackColor: WidgetStatePropertyAll(Colors.yellow),
    trackOutlineColor: WidgetStatePropertyAll(Colors.green),
    trackOutlineWidth: WidgetStatePropertyAll(4),
  ), // SwitchThemeData
  colorScheme: ColorScheme.fromSeed(
```



With a **WidgetStateProperty** you can define different properties dependent if the widget is e.g. pressed, hovered, disabled …
WidgetStatePropertyAll sets the same value for all these states.

Same for the OutlinedButtons

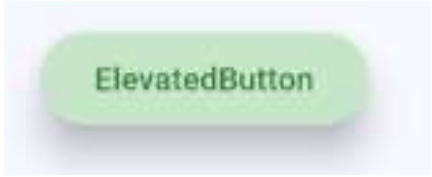



```
OutlinedButtonThemeData OutlinedButtonThemeData({ButtonStyle? style})
```

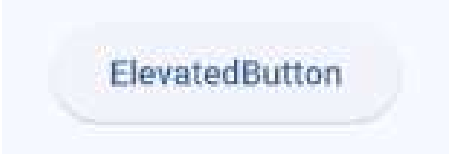
```
outlinedButtonTheme: OutlinedButtonThemeData(
  style: OutlinedButton.styleFrom(
    backgroundColor: Colors.green.shade100,
    side: const BorderSide(color: Colors.green, width: 2),
  ),
), // OutlinedButtonThemeData
```

Exercise

Define ThemeData in such a way, that all ElevatedButtons in your application have some green shape like



The default display of ElevatedButtons for seedColor blue is:



Solution

```
return MaterialApp(
  theme: ThemeData(
    elevatedButtonTheme: ElevatedButtonThemeData(
      style: ElevatedButton.styleFrom(
          backgroundColor: Colors.green.shade100,
          foregroundColor: Colors.green.shade800,
          elevation: 15),
    ), // ElevatedButtonThemeData
```

Alternative or Add-On to using ThemeData

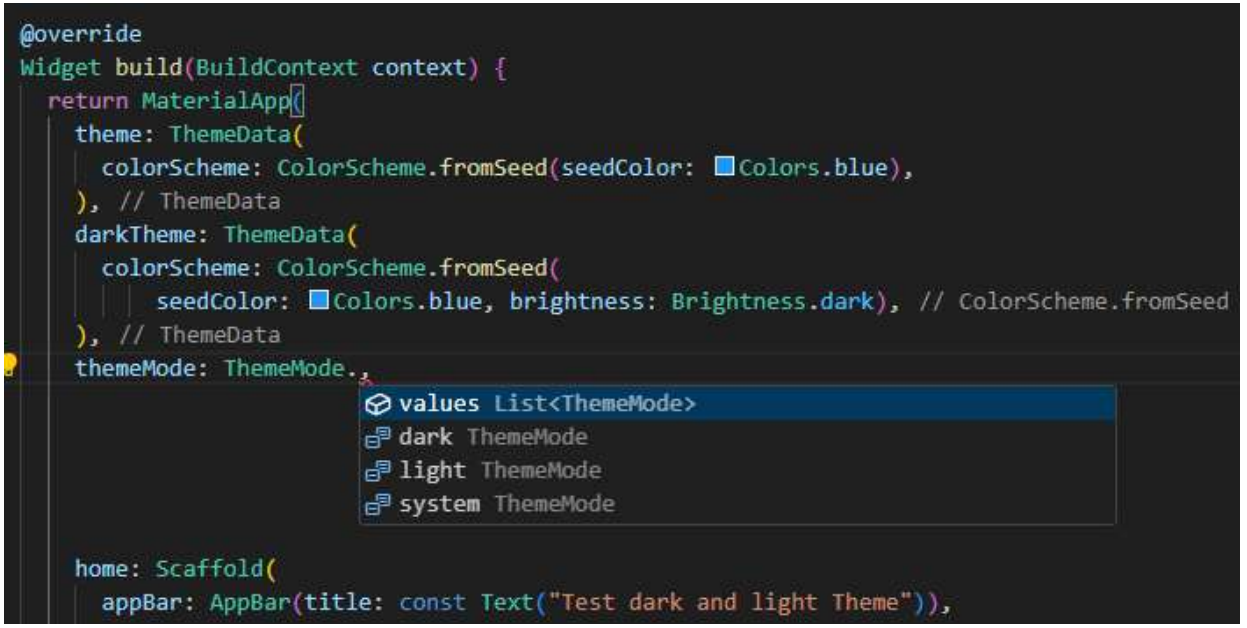
In case you often want to use Ok / Cancel buttons in your app:

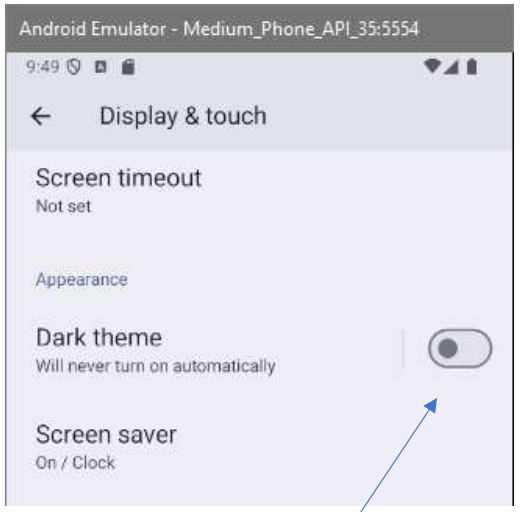

```
class OkButton extends OutlinedButton {
  OkButton({super.key, required super.onPressed})
      : super(
            style: OutlinedButton.styleFrom(
                backgroundColor: Colors.green, foregroundColor: Colors.white),
            child: const Text("Ok"));
}
```

```
class CancelButton extends OutlinedButton {
  CancelButton({super.key, required super.onPressed})
      : super(
            style: OutlinedButton.styleFrom(
                backgroundColor: Colors.red, foregroundColor: Colors.white),
            child: const Text("Cancel"));
}
```

```
Row(
  mainAxisAlignment: MainAxisAlignment.spaceEvenly,
  children: [
    OkButton(onPressed: onButtonPressed),
    CancelButton(onPressed: onButtonPressed),
  ],
), // Row
```

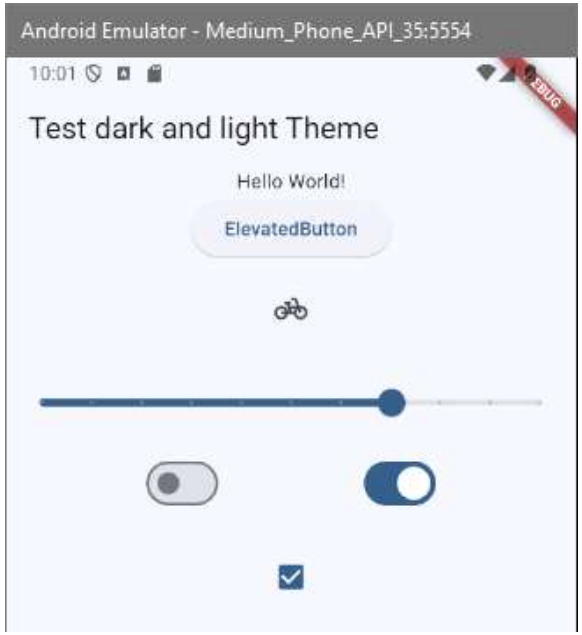
Light and Dark Theme



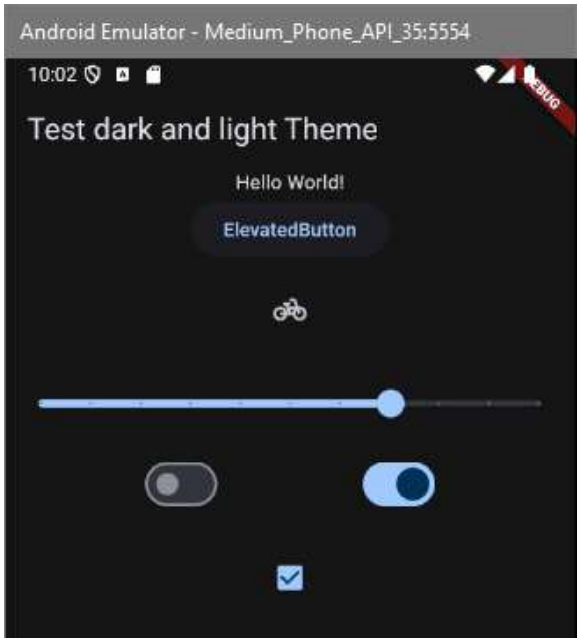


Default ThemeMode is “**ThemeMode.system**”, that means:
use light or dark theme depending on what the user has defined in his Settings.

Light and Dark Theme for seed color blue



light theme



dark theme