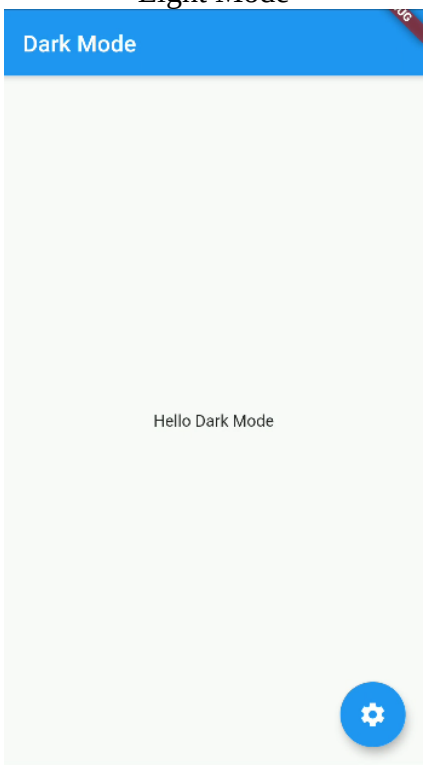
State Management

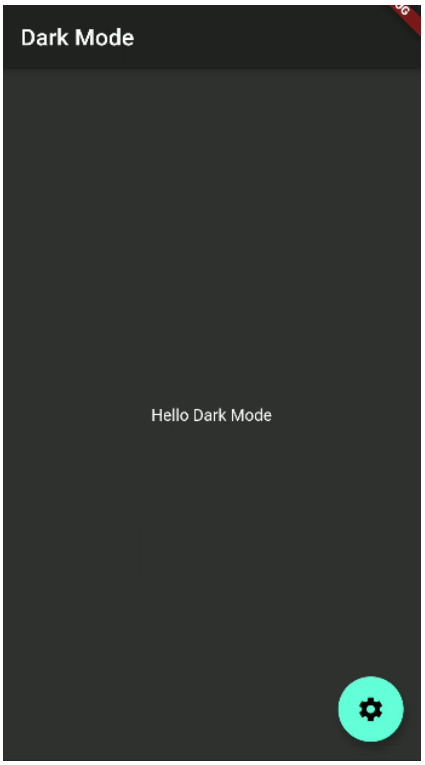
State Management adalah sebuah desain dalam coding di mana kita dapat memisahkan antara logic dan view, Tujuannya adalah agar logic dapat kembali digunakan. Pada kotlin atau java biasanya kita mengenal istilah **MVVM** atau **MVP**, pada web **MVC**, maka di Flutter ada **BLoC**, **Provider**, **Redux**, atau **MobX**.

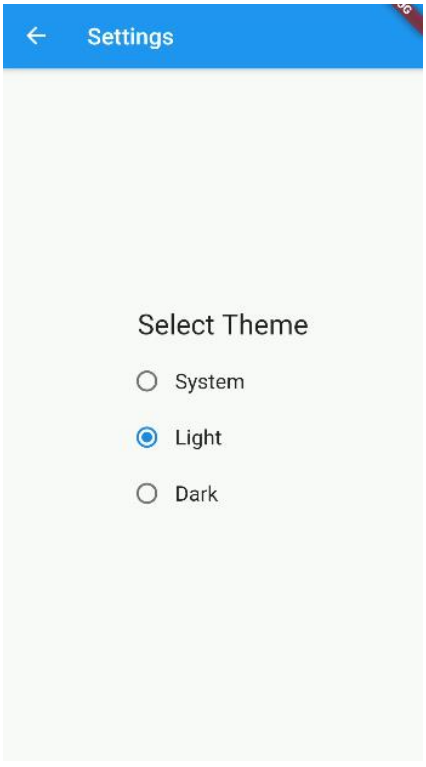
Kali ini, kita akan menggunakan salah satunya yakni **Provider**. Contohnya adalah untuk mengubah tema warna tampilan aplikasi seperti contoh di bawah:

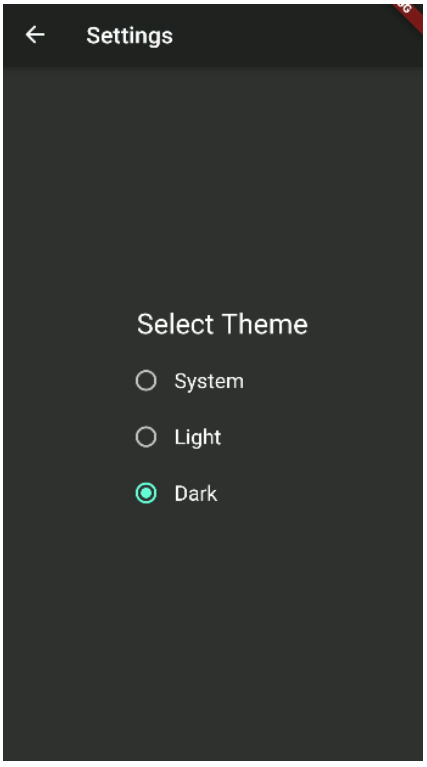
Light Mode



Dark Mode







Untuk menggunakan package **Provider**, pastikan Anda memasukkannya dalam **pubspec.yaml**

1. Main

```
lib > main.dart > MyApp > build
```

```dart
import 'package:flutter/material.dart';
import 'package:provider/provider.dart';
import 'settings_screen.dart';
import 'theme_manager.dart';

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

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return ChangeNotifierProvider<ThemeManager>(
      create: (context) => ThemeManager(),
      child: MaterialAppWithTheme(),
    ); // ChangeNotifierProvider
  }
}

class MaterialAppWithTheme extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    final themeManager = Provider.of<ThemeManager>(context);
    return MaterialApp(
      home: StartScreen(),
      theme: ThemeData.light(),
      darkTheme: ThemeData.dark(),
      themeMode: themeManager.themeMode,
    ); // MaterialApp
  }
}
```

```dart
class StartScreen extends StatelessWidget {
  StartScreen({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dark Mode"),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text("Hello Dark Mode"),
          ], // <Widget>[]
        ), // Column
      ), // Center
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => SettingsScreen(),
            ), // MaterialPageRoute
          );
        },
        tooltip: 'Settings',
        child: Icon(Icons.settings),
      ), // FloatingActionButton
    ); // Scaffold
  }
}
```

```
lib > main.dart > MyApp > build
```

```dart
      darkTheme: ThemeData.dark(),
      themeMode: themeManager.themeMode,
    ); // MaterialApp
  }
}

class StartScreen extends StatelessWidget {
  StartScreen({Key key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Dark Mode"),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisAlignment: MainAxisAlignment.center,
          children: <Widget>[
            Text("Hello Dark Mode"),
          ], // <Widget>[]
        ), // Column
      ), // Center
      floatingActionButton: FloatingActionButton(
        onPressed: () {
          Navigator.push(
            context,
            MaterialPageRoute(
              builder: (context) => SettingsScreen(),
            ), // MaterialPageRoute
          );
        },
        tooltip: 'Settings',
        child: Icon(Icons.settings),
      ), // FloatingActionButton
    ); // Scaffold
  }
}
```

2. Settings_screen

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

class _SettingsScreenState extends State<SettingsScreen> {
  ThemeMode _groupValue;
  ThemeManager _themeManager;

  @override
  void initState() {
    super.initState();
    _themeManager = Provider.of<ThemeManager>(context, listen: false);
    _groupValue = _themeManager.themeMode;
  }

  void _updateTheme(ThemeMode themeMode) {
    _themeManager.themeMode = themeMode;
  }

  @override
  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text('Settings'),
      ), // AppBar
      body: Center(
        child: Column(
          mainAxisSize: MainAxisSize.max,
          mainAxisAlignment: MainAxisAlignment.center,
          crossAxisAlignment: CrossAxisAlignment.start,
          children: <Widget>[
            Padding(
              padding: EdgeInsets.only(left: 16.0, bottom: 10.0),
              child: Text("Select Theme", style: TextStyle(fontSize: 24.0),),
            ), // Padding
            Row(
              mainAxisAlignment: MainAxisAlignment.start,
              mainAxisSize: MainAxisSize.min,
              children: <Widget>[
                Radio(
                  onChanged: (val) => setState(() {
                    _groupValue = val;
                    _updateTheme(val);
                  }),
                  value: ThemeMode.system,
                  groupValue: _groupValue,
                ), // Radio
                GestureDetector(
                  onTap: () => setState(() {
                    var val = ThemeMode.system;
                    _groupValue = val;
                    _updateTheme(val);
                  }),
                  child: Text(
                    "System",
                    style: TextStyle(fontSize: 18.0),
                  ), // Text
                ), // GestureDetector
              ], // <Widget>[]
            ), // Row
            Row(
              mainAxisAlignment: MainAxisAlignment.start,
              mainAxisSize: MainAxisSize.min,
              children: <Widget>[
                Radio(
                  onChanged: (val) => setState(() {
                    _groupValue = val;
                    _updateTheme(val);
                  }),
                  value: ThemeMode.light,
                  groupValue: _groupValue,
                ), // Radio
                GestureDetector(
                  onTap: () => setState(() {
                    var val = ThemeMode.light;
                    _groupValue = val;
                    _updateTheme(val);
                  }),
                  child: Text(
                    "Light",
                    style: TextStyle(fontSize: 18.0),
                  ), // Text
                ), // GestureDetector
              ], // <Widget>[]
            ), // Row
            Row(
              mainAxisAlignment: MainAxisAlignment.start,
              mainAxisSize: MainAxisSize.min,
              children: <Widget>[
                Radio(
                  onChanged: (val) => setState(() {
                    _groupValue = val;
                    _updateTheme(val);
                  }),
                  value: ThemeMode.dark,
                  groupValue: _groupValue,
                ), // Radio
                GestureDetector(
                  onTap: () => setState(() {
                    var val = ThemeMode.dark;
                    _groupValue = val;
                    _updateTheme(val);
                  }),
                  child: Text(
                    "Dark",
                    style: TextStyle(fontSize: 18.0),
                  ), // Text
                ), // GestureDetector
              ], // <Widget>[]
            ), // Row
          ], // <Widget>[]
        ), // Column
      ), // Center
    ); // Scaffold
  }
}

class SettingsScreen extends StatefulWidget {
  @override
  _SettingsScreenState createState() => _SettingsScreenState();
}
```

3. Theme_manager

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

class ThemeManager with ChangeNotifier {
  ThemeMode _themeMode;
  ThemeMode defaultThemeMode = ThemeMode.system;

  ThemeManager() {
    _getThemeModeFromSharedPrefs();
  }

  get themeMode {
    return _themeMode;
  }

  set themeMode(ThemeMode themeMode)  {
    _themeMode = themeMode;
    _saveThemeModeInSharedPrefs(themeMode);
    notifyListeners();
  }

  void _getThemeModeFromSharedPrefs() async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    String themeModeFromPrefs = prefs.getString('themeMode');

    _themeMode = ThemeMode.values.firstWhere(
      (element) => themeModeFromPrefs == element.toString(),
      orElse: () => defaultThemeMode
    );
    notifyListeners();
  }

  _saveThemeModeInSharedPrefs(ThemeMode themeMode) async {
    SharedPreferences prefs = await SharedPreferences.getInstance();
    prefs.setString('themeMode', themeMode.toString());
  }
}

```