## **Button**

Kali ini kita akan belajar menggunakan widget button. Widget button ini adalah widget yang dapat menerima trigger sentuhan atau dapat melakukan suatu fungsi ketika disentuh, widget-widget button tersebut antara lain:

### **ElevatedButton**

ElevatedButton merupakan bagian dari Material Design widget dari Flutter. Untuk menggunakan ElevatedButton caranya seperti berikut:

* ElevatedButton(
* child: const Text("Tombol"),
* onPressed: () {
* // Aksi ketika button diklik
* },
* ),

Pada kode di atas ElevatedButton memiliki 2 parameter yaitu onPressed dan child. Parameter *onPressed* merupakan sebuah *function event* ketika tombol ditekan dan sebenarnya ada *event* lain seperti *onLongPress* dan *onHighlightChanged*. Parameter *child* diisi oleh widget pada umumnya.

### **TextButton**

TextButton merupakan widget button yang memiliki tampilan yang polos selayaknya Text. TextButton umumnya digunakan pada toolbars, dialog, atau bersama komponen button lain. Contoh kode dari TextButton adalah seperti berikut:

* TextButton(
* child: const Text('Text Button'),
* onPressed: () {
* // Aksi ketika button diklik
* },
* ),

Sama halnya ElevatedButton, TextButton juga memiliki parameter onPressed dan child.

### **OutlinedButton**

OutlinedButton juga merupakan bagian dari material design yang menyediakan tampilan TextButton dengan tambahan outline. OutlinedButton umumnya digunakan untuk tombol atau aksi yang penting, tetapi bukan aksi utama dalam aplikasi.

Berikut ini adalah contoh widget OutlinedButton:

* OutlinedButton(
* child: const Text('Outlined Button'),
* onPressed: () {
* // Aksi ketika button diklik
* },
* ),

Tampilan OutlinedButton sendiri akan seperti berikut:

### **IconButton**

IconButton merupakan widget button dengan icon. Tak seperti widget tombol lainnya, widget IconButton ini tidak memiliki child. Perhatikan kode di bawah ini:

* IconButton(
* icon: const Icon(Icons.volume\_up),
* tooltip: 'Increase volume by 10',
* onPressed: () {
* // Aksi ketika button diklik
* },
* ),

Seperti yang kita lihat di atas, IconButton tidak menggunakan child untuk isi (*content*) melainkan menggunakan parameter icon dan *tooltip* (penunjuk) untuk memberikan hint pada tombol.

### **DropdownButton**

DropdownButton merupakan tombol yang saat diklik, akan muncul *pop-up* daftar beberapa item yang dapat kita pilih salah satu. Berikut contoh kodenya:

* class FirstScreen extends StatefulWidget {
* const FirstScreen({Key? key}) : super(key: key);
* @override
* State<FirstScreen> createState() => \_FirstScreenState();
* }
* class \_FirstScreenState extends State<FirstScreen> {
* String? language;
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: const Text('First Screen'),
* ),
* body: DropdownButton<String>(
* items: const <DropdownMenuItem<String>>[
* DropdownMenuItem<String>(
* value: 'Dart',
* child: Text('Dart'),
* ),
* DropdownMenuItem<String>(
* value: 'Kotlin',
* child: Text('Kotlin'),
* ),
* DropdownMenuItem<String>(
* value: 'Swift',
* child: Text('Swift'),
* ),
* ],
* value: language,
* hint: const Text('Select Language'),
* onChanged: (String? value) {
* setState(() {
* language = value;
* });
* },
* ),
* );
* }
* }

Pada contoh tersebut DropdownButton tidak menggunakan child maupun children, akan tetapi menggunakan items di mana berisi list dari widget DropdownMenuItem. Pada widget DropdownMenuItem terdapat child untuk tiap itemnya dan value yang ada pada DropdownMenuItem adalah nilai dari tiap itemnya. Nantinya akan dibutuhkan parameter onChanged ketika ada perubahan atau ketika memilih salah satu dari item tersebut dan mengubah nilai language atau value dari DropdownButton tersebut. Sedangkan hint berfungsi ketika nilai value dari DropdownButton *null* atau kosong.

Selengkapnya tentang berbagai widget Button, bacalah pada tautan berikut:

* [Button Material Components](https://flutter.dev/docs/development/ui/widgets/material#Buttons)
* [ElevatedButton Class](https://api.flutter.dev/flutter/material/ElevatedButton-class.html)
* [TextButton Class](https://api.flutter.dev/flutter/material/TextButton-class.html)
* [OutlinedButton](https://api.flutter.dev/flutter/material/OutlinedButton-class.html)
* [IconButton Class](https://api.flutter.dev/flutter/material/IconButton-class.html)
* [DropdownButton Class](https://api.flutter.dev/flutter/material/DropdownButton-class.html)

## **Input Widget**

Salah satu bentuk interaksi dengan pengguna adalah dengan menerima input. Ada beberapa input widget yang bisa digunakan supaya pengguna bisa berinteraksi dengan aplikasi. Perhatikan bahwa input pengguna ini berkaitan dengan state yang dapat sering berubah. Karena itu umumnya input widget akan ditempatkan di dalam StatefulWidget.

### **TextField**

TextField merupakan sebuah widget yang digunakan untuk menerima input berupa teks yang berasal dari *keyboard*. Terdapat beberapa cara yang bisa Anda gunakan untuk mendapatkan nilai dari TextField. Salah satunya adalah melalui parameter onChanged.

* String \_name = '';
* TextField(
* onChanged: (String value) {
* setState(() {
* \_name = value;
* });
* },
* )

Parameter *onChanged* berisi sebuah fungsi yang akan dipanggil setiap terjadi perubahan inputan pada TextField. Pada fungsi ini, kita dapat mengubah nilai variabel state dengan memanggil fungsi setState().

Jika Anda tidak ingin mengambil nilai setiap perubahan, tetapi hanya ketika seluruh input sudah selesai di-*submit*, Anda dapat menggunakan parameter onSubmitted seperti berikut:

* String \_name = '';
* TextField(
* onSubmitted: (String value) {
* setState(() {
* \_name = value;
* });
* },
* )

Cara lain yang bisa kita gunakan adalah dengan TextEditingController. Dengan controller, kita cukup membuat variabel TextEditingController lalu menambahkannya ke widget TextField.

* TextEditingController \_controller = TextEditingController();
* TextField(
* controller: \_controller,
* ),

Ketika menggunakan controller, pastikan untuk menghapus controller ketika halaman atau widget sudah tidak digunakan. Ini bertujuan supaya tidak menimbulkan kebocoran memori (*memory leak*).

* @override
* void dispose() {
* \_controller.dispose();
* super.dispose();
* }

Berikut ini adalah contoh penerapan widget TextField:

Untuk membuat TextField seperti di atas, Anda bisa menggunakan kode seperti berikut:

* **[onChanged](https://www.dicoding.com/academies/159/tutorials/8606?from=6504#tab1-code1)**
* [Controller](https://www.dicoding.com/academies/159/tutorials/8606?from=6504#tab1-code2)
* class \_FirstScreenState extends State<FirstScreen> {
* String \_name = '';
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: const Text('First Screen'),
* ),
* body: Padding(
* padding: const EdgeInsets.all(16.0),
* child: Column(
* children: [
* TextField(
* decoration: const InputDecoration(
* hintText: 'Write your name here...',
* labelText: 'Your Name',
* ),
* onChanged: (String value) {
* setState(() {
* \_name = value;
* });
* },
* ),
* const SizedBox(height: 20),
* ElevatedButton(
* child: const Text('Submit'),
* onPressed: () {
* showDialog(
* context: context,
* builder: (context) {
* return AlertDialog(
* content: Text('Hello, $\_name'),
* );
* });
* },
* )
* ],
* ),
* ),
* );
* }
* }

## **Switch**

Switch merupakan inputan yang mengembalikan nilai boolean true atau false. Perhatikan contoh berikut:

* class \_FirstScreenState extends State<FirstScreen> {
* bool lightOn = false;
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: const Text('First Screen'),
* ),
* body: Switch(
* value: lightOn,
* onChanged: (bool value) {
* setState(() {
* lightOn = value;
* });
* ScaffoldMessenger.of(context).showSnackBar(
* SnackBar(
* content: Text(lightOn ? 'Light On' : 'Light Off'),
* duration: Duration(seconds: 1),
* ),
* );
* },
* ),
* );
* }
* }

Pada contoh tersebut value dari Switch berupa boolean di mana ketika boolean tersebut false maka Switch akan berada pada posisi nonaktif. Switch umumnya digunakan sebagai konfigurasi on/off pada halaman setting.

### **Radio**

Radio merupakan inputan yang digunakan untuk memilih salah satu dari beberapa pilihan dalam suatu kelompok. Berikut contohnya:

* class \_FirstScreenState extends State<FirstScreen> {
* String? language;
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: const Text('First Screen'),
* ),
* body: Column(
* mainAxisSize: MainAxisSize.min,
* children: <Widget>[
* ListTile(
* leading: Radio<String>(
* value: 'Dart',
* groupValue: language,
* onChanged: (String? value) {
* setState(() {
* language = value;
* showSnackbar();
* });
* },
* ),
* title: Text('Dart'),
* ),
* ListTile(
* leading: Radio<String>(
* value: 'Kotlin',
* groupValue: language,
* onChanged: (String? value) {
* setState(() {
* language = value;
* showSnackbar();
* });
* },
* ),
* title: Text('Kotlin'),
* ),
* ListTile(
* leading: Radio<String>(
* value: 'Swift',
* groupValue: language,
* onChanged: (String? value) {
* setState(() {
* language = value;
* showSnackbar();
* });
* },
* ),
* title: Text('Swift'),
* ),
* ],
* ),
* );
* }
* void showSnackbar() {
* ScaffoldMessenger.of(context).showSnackBar(
* SnackBar(
* content: Text('$language selected'),
* duration: Duration(seconds: 1),
* ),
* );
* }
* }

Pada contoh tersebut terdapat variable language yang digunakan pada groupValue tiap Radio. Language inilah yang menyimpan nilai Radio yang dipilih. Nilainya akan berubah ketika fungsi onChanged terpanggil.

### **Checkbox**

Checkbox merupakan inputan benar atau salah. Checkbox akan berisi centang jika nilainya adalah benar dan kosong jika salah. Seperti pada contoh berikut:

* class \_FirstScreenState extends State<FirstScreen> {
* bool agree = false;
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: const Text('First Screen'),
* ),
* body: ListTile(
* leading: Checkbox(
* value: agree,
* onChanged: (bool? value) {
* setState(() {
* agree = value!;
* });
* },
* ),
* title: Text('Agree / Disagree'),
* ),
* );
* }
* }

Kode di atas jika dijalankan akan tampil seperti berikut:

Ada beberapa tautan yang dapat Anda baca untuk memahami tentang widget-widget input yang ada pada Flutter, antara lain:

* [Input and selections widgets](https://flutter.dev/docs/development/ui/widgets/material#Input%20and%20selections)
* [TextField Class](https://api.flutter.dev/flutter/material/TextField-class.html)
* [Switch Class](https://api.flutter.dev/flutter/material/Switch-class.html)
* [Radio Class](https://api.flutter.dev/flutter/material/Radio-class.html)
* [Checkbox Class](https://api.flutter.dev/flutter/material/Checkbox-class.html)

## Image

Dalam pengembangan suatu aplikasi kita tidak akan lepas dari image atau gambar untuk membuat tampilan semakin menarik. Pada materi kali ini kita akan belajar bagaimana menampilkan gambar dari internet dan project asset.

### **Image.network**

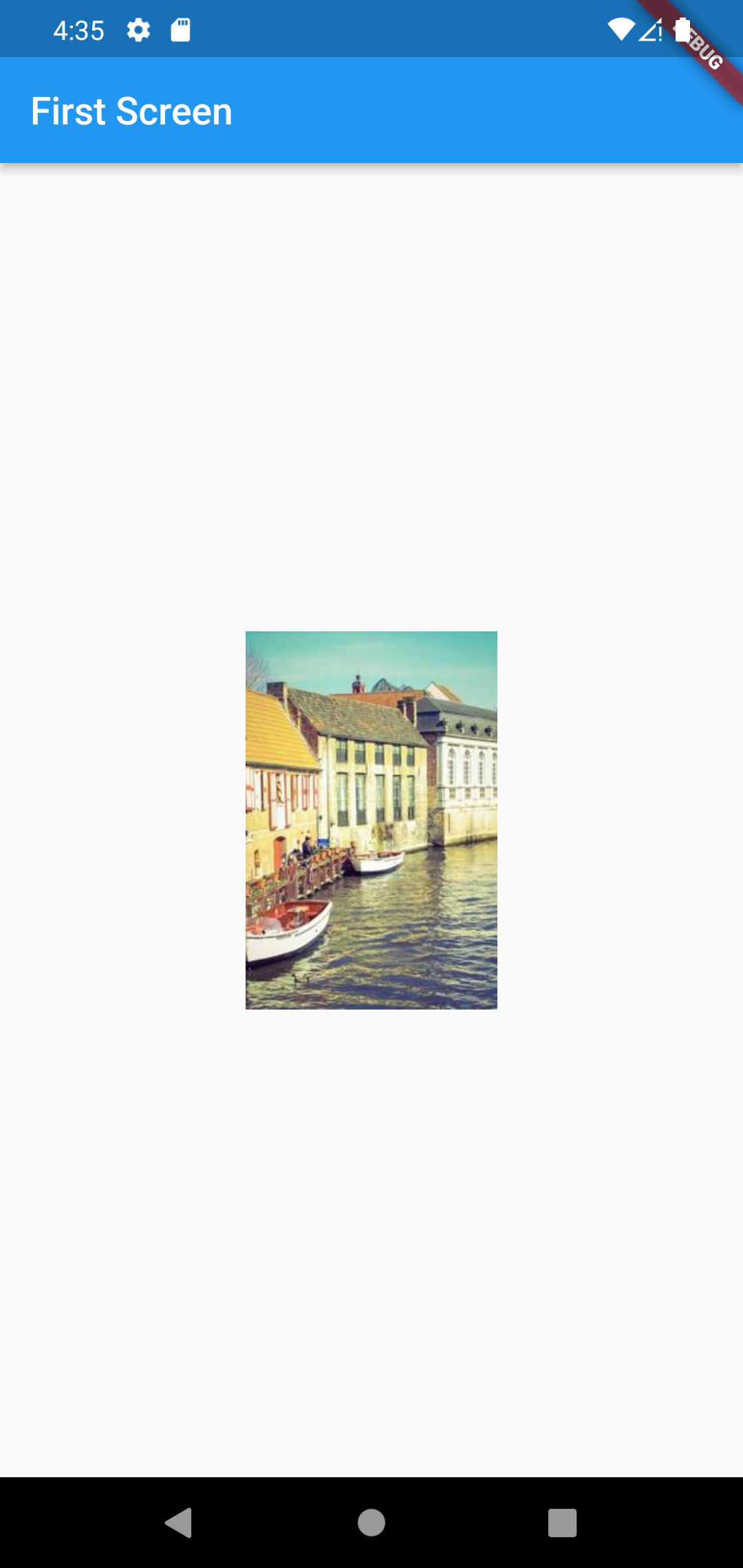
Untuk menampilkan gambar yang bersumber dari internet, kita akan menggunakan method Image.network. Cara penulisan method ini sebagai berikut:

* Image.network(url)

Method ini cukup menambahkan URL gambar dari internet dan kita pun dapat menambahkan width dan height juga. Di bawah ini adalah contoh penggunaan Image.network:

* class FirstScreen extends StatelessWidget {
* const FirstScreen({Key? key}) : super(key: key);
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: const Text('First Screen'),
* ),
* body: Center(
* child: Image.network(
* 'https://picsum.photos/200/300',
* width: 200,
* height: 200,
* ),
* ),
* );
* }
* }

Pada kode di atas kita panggil method Image.network dengan url *https://picsum.photos/200/300* lalu beri *width* dan *height* masing-masing 200. Sehingga hasilnya seperti berikut:



### **Image.asset**

Selain melalui internet, kita juga dapat menampilkan gambar yang bersumber dari asset *project*.. Asset di sini berupa gambar-gambar yang nantinya didaftarkan pada *project*. Untuk mendaftarkan asset gambar pada project kita harus menambahkannya pada berkas **pubspec.yaml**.

Pertama kita harus menambahkan terlebih dahulu gambar yang akan didaftarkan ke dalam folder project kita. Saat ini Flutter mendukung beberapa jenis format gambar, seperti JPEG, PNG, GIF, Animated GIF, WebP, Animated WebP, BMP, dan WBMP. Di luar format tersebut, Flutter akan memanfaatkan API dari masing-masing platform. Jika platform native mendukung format gambar yang digunakan, maka gambar tersebut akan bisa di-render oleh Flutter.

Pada contoh berikut kita menambahkan folder **images/** pada folder project.

Masukkan berkas gambar yang ingin Anda gunakan ke dalam folder **image**. Sebagai contoh kita menggunakan gambar bernama [**android.png**](https://github.com/dicodingacademy/assets/raw/main/flutter_fundamental_academy/android.png).

Setelah menambahkan gambar pada *project*, saatnya kita mendaftarkan gambar tersebut pada **pubspec.yaml**.

Di dalam berkas **pubspec.yaml**, kita bisa mendaftarkan aset gambar pada bagian flutter seperti di bawah ini:

* ...
* flutter:
* uses-material-design: true
* # To add assets to your application, add an assets section, like this:
* # assets:
* # - images/a\_dot\_burr.jpeg
* # - images/a\_dot\_ham.jpeg
* ...

Daftarkan asset gambar seperti berikut:

* ...
* flutter:
* uses-material-design: true
* assets:
* - images/android.png
* ...

Hapus juga tanda pagar (#) atau komentar yang tidak diperlukan. Perhatikan pula indentasi kodenya. assets: berada sejajar dengan uses-material-design: yaitu berjarak 2 spasi dari ujung dan berada di dalam flutter: sedangkan - images/android.png berada di dalam assets: dan berjarak 4 spasi dari ujung.

Pada contoh di atas kita telah menambahkan asset yang berisi lokasi gambar atau aset yang ingin kita gunakan. Karena kita menambahkan gambar **android.png** pada folder *images*, maka lokasi gambar tersebut adalah **images/android.png**.

Apabila ada banyak gambar yang kita masukkan ke dalam lokasi folder, dibandingkan menuliskan lokasi gambar satu per satu, kita bisa langsung menuliskan folder **images/** seperti berikut:

* ...
* flutter:
* uses-material-design: true
* assets:
* - images/
* ...

Setelah menambahkan *assets*, kita harus me-*refresh* **pubspec.yaml** dengan cara *save file* pubspec.yaml bila menggunakan Visual Studio Code atau menekan 'Packages get' yang ada di pojok kanan atas untuk Android Studio.

Setelah kita menambahkan asset ke dalam pubspec.yaml kita perlu melakukan full restart agar asset yang baru dapat digunakan dalam aplikasi.

Kita telah mendaftarkan suatu asset. Sekarang kita akan panggil asset tersebut pada kode kita dengan method Image.asset. Cara penulisannya seperti berikut:

* Image.asset(lokasi\_asset)

Contoh dalam kodenya akan seperti berikut:

* class FirstScreen extends StatelessWidget {
* const FirstScreen({Key? key}) : super(key: key);
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* appBar: AppBar(
* title: const Text('First Screen'),
* ),
* body: Center(
* child: Image.asset('images/android.png', width: 200, height: 200),
* ),
* );
* }
* }

Jika kita jalankan aplikasi Flutter, maka gambar akan tampil seperti berikut:



Untuk mempelajari widget Image lebih lanjut, Anda dapat membaca dokumentasinya pada tautan [Image Class](https://api.flutter.dev/flutter/widgets/Image-class.html).

## **Font**

Dalam pengembangan suatu aplikasi, seorang *User Interface* desainer dapat menggunakan font berbeda dengan *default* font yang ada. Sebagai pengembang aplikasi kita diharuskan menambahkan font pada aplikasi yang dirancang oleh desainer agar sesuai dengan desain User Interface.

Pada pembelajaran kali ini kita akan belajar bagaimana menambahkan *font* pada Flutter. Sebelum kita memulai pembelajaran, kita akan mengunduh font yang ada di internet atau menggunakan *font* yang telah dimiliki. Pada contoh ini kita akan mengunduh salah satu *font* dari [Google Fonts](https://fonts.google.com/) yaitu [Oswald](https://fonts.google.com/specimen/Oswald).

### **Menambahkan Font ke Project**

Setelah mengunduh font, langkah selanjutnya kita akan memasukkan *file-file* font tersebut ke folder *project*. Pada contoh ini kita akan membuat folder fonts pada *project* kita, dan masukkan *file-file* font yang telah diunduh, seperti berikut:

### **Mendaftarkan Font di pubspec.yaml**

Sama halnya dengan gambar, kita perlu mendaftarkan font pada berkas **pubspec.yaml** sebagai asset seperti berikut:

* flutter:
* uses-material-design: true
* assets:
* - images/
* **fonts:**
* **- family: Oswald**
* **fonts:**
* **- asset: fonts/Oswald/Oswald-Regular.ttf**

Sama halnya dengan gambar, font ada dalam bagian flutter. Untuk mendaftarkan font, kita membuat bagian fonts yang ada dalam blok flutter.

Untuk mendaftarkan font *Oswald* kita tuliskan *Oswald* pada bagian *family* yang nantinya akan menjadi nama font yang kita panggil pada kode dart. Lalu dalam *family* kita masukkan fonts yang di dalamnya terdapat asset yang nanti akan mengarah pada *file* font.ttf.Contoh di atas kita menambahkan asset *fonts/oswald/Oswald-Regular.ttf*.

### **Menggunakan Font pada Kode**

Setelah kita mendaftarkan font pada **pubspec.yaml** kita akan gunakan font tersebut pada kode kita. Seperti contoh di bawah ini kita akan menggunakan font pada widget **Text**:

* Text(
* 'Custom Font',
* style: TextStyle(
* **fontFamily: 'Oswald',**
* fontSize: 30,
* ),
* ),

Pada kode di atas kita menambahkan fontFamily pada TextStyle. Kita cukup panggil nama font family yang telah kita daftarkan pada **pubspec.yaml**. Hasilnya akan seperti berikut:

Tulisan "**Custom Font**" akan berubah menjadi font **Oswald** sesuai dengan yang telah kita daftarkan.

Jangan lupa! Setelah kita menambahkan *package* atau pun asset ke dalam **pubspec.yaml** kita perlu melakukan **full restart** agar asset yang baru dapat digunakan dalam aplikasi.

### **Mengubah Font Default**

Selain kita dapat mengubah font family pada satu per satu *widget Text*, kita dapat membuat font yang kita daftarkan menjadi *default*. Caranya dengan menambahkan parameter fontFamily pada kelas ThemeData yang ada pada *parameter* *theme* di **MaterialApp** seperti berikut:

* class MyApp extends StatelessWidget {
* const MyApp({Key? key}) : super(key: key);
* @override
* Widget build(BuildContext context) {
* return MaterialApp(
* title: 'Flutter Demo',
* theme: ThemeData(
* **fontFamily: 'Oswald',**
* primarySwatch: Colors.blue,
* ),
* home: FirstScreen(),
* );
* }
* }

## **ListView**

Pada Codelab kedua kita telah menggunakan dan menyinggung sedikit tentang widget **ListView**. Widget ini digunakan untuk menampilkan beberapa item dalam bentuk baris atau kolom dan bisa di-*scroll*.

Cara penggunaan *ListView* ini mirip dengan **Column** atau **Row** di mana Anda memasukkan widget yang ingin disusun sebagai **children** dari **ListView**.

* class ScrollingScreen extends StatelessWidget {
* const ScrollingScreen({Key? key}) : super(key: key);
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* body: ListView(
* children: <Widget>[
* Container(
* height: 250,
* decoration: BoxDecoration(
* color: Colors.grey,
* border: Border.all(color: Colors.black),
* ),
* child: const Center(
* child: Text(
* '1',
* style: TextStyle(fontSize: 50),
* ),
* ),
* ),
* Container(
* height: 250,
* decoration: BoxDecoration(
* color: Colors.grey,
* border: Border.all(color: Colors.black),
* ),
* child: const Center(
* child: Text(
* '2',
* style: TextStyle(fontSize: 50),
* ),
* ),
* ),
* Container(
* height: 250,
* decoration: BoxDecoration(
* color: Colors.grey,
* border: Border.all(color: Colors.black),
* ),
* child: const Center(
* child: Text(
* '3',
* style: TextStyle(fontSize: 50),
* ),
* ),
* ),
* Container(
* height: 250,
* decoration: BoxDecoration(
* color: Colors.grey,
* border: Border.all(color: Colors.black),
* ),
* child: const Center(
* child: Text(
* '4',
* style: TextStyle(fontSize: 50),
* ),
* ),
* ),
* ],
* ),
* );
* }
* }

Ketika dijalankan, aplikasi akan menjadi seperti berikut:

20210425152905aa85dcf49bf24cb127274cbc16ea35cd.gif

### **Menampilkan Item Secara Dinamis**

Selain memasukkan widget satu per satu ke dalam **children** dari **ListView**, Anda juga dapat menampilkan *list* secara dinamis. Ini sangat berguna ketika Anda memiliki banyak item dengan jumlah yang tidak menentu.

Misalnya kita ingin menampilkan daftar angka dari 1 sampai 10.

* final List<int> numberList = const <int>[1, 2, 3, 4, 5, 6, 7, 8, 9, 10];

Caranya, masukkan variabel atau *list* Anda sebagai *children* lalu panggil fungsi map(). Fungsi map ini berguna untuk memetakan atau mengubah setiap item di dalam list menjadi objek yang kita inginkan. Fungsi map ini membutuhkan satu buah parameter berupa fungsi atau lambda.

* class ScrollingScreen extends StatelessWidget {
* const ScrollingScreen({Key? key}) : super(key: key);
* final List<int> numberList = const <int>[1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* body: ListView(
* children: numberList.map((number) {}),
* ),
* );
* }
* }

Karena parameter children ini membutuhkan nilai berupa *list* widget, maka kita perlu mengembalikan setiap item dari **numberList** menjadi **widget** yang akan ditampilkan. Ubah fungsi **lambda** Anda menjadi seperti berikut:

* class ScrollingScreen extends StatelessWidget {
* final List<int> numberList = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10];
* @override
* Widget build(BuildContext context) {
* return Scaffold(
* body: ListView(
* children: **numberList.map((number) {**
* **return Container(**
* **height: 250,**
* **decoration: BoxDecoration(**
* **color: Colors.grey,**
* **border: Border.all(color: Colors.black),**
* **),**
* **child: Center(**
* **child: Text(**
* **'$number', // Ditampilkan sesuai item**
* **style: const TextStyle(fontSize: 50),**
* **),**
* **),**
* **);**
* **}).toList(),**
* ),
* );
* }
* }

Perhatikan di akhir kita perlu mengembalikan fungsi map menjadi objek *List* lagi dengan fungsi .toList(). Lakukan **hot reload** pada aplikasi Anda untuk melihat hasil perubahan.

### **Menggunakan ListView.builder**

Selain mengisi parameter children dari ListView seperti sebelumnya, kita juga bisa memanfaatkan method ListView.builder. ListView.builder lebih cocok digunakan pada ListView dengan jumlah item yang cukup besar. Ini karena Flutter hanya akan merender tampilan item yang terlihat di layar dan tidak me-*render* seluruh item ListView di awal.

ListView.builder memerlukan dua parameter yaitu itemBuilder dan itemCount. Parameter itemBuilder merupakan fungsi yang mengembalikan widget untuk ditampilkan. Sedangkan itemCount kita isi dengan jumlah seluruh item yang ingin ditampilkan.

Berikut ini adalah contoh kode penggunaan ListView.builder:

* ListView.builder(
* itemCount: numberList.length,
* itemBuilder: (BuildContext context, int index) {
* return Container(
* height: 250,
* decoration: BoxDecoration(
* color: Colors.grey,
* border: Border.all(color: Colors.black),
* ),
* child: Center(
* child: Text(
* '${numberList[index]}',
* style: const TextStyle(fontSize: 50),
* ),
* ),
* );
* },
* ),

### 

### **ListView.separated**

Cara lain untuk membuat ListView adalah dengan metode ListView.separated. ListView jenis ini akan menampilkan daftar item yang dipisahkan dengan separator. Penggunaan ListView.separated mirip dengan builder, yang membedakan adalah terdapat satu parameter tambahan wajib yaitu separatorBuilder yang mengembalikan Widget yang akan berperan sebagai separator.

Berikut ini adalah contoh kode dari ListView.separated:

* ListView.separated(
* itemCount: numberList.length,
* itemBuilder: (BuildContext context, int index) {
* return Container(
* height: 250,
* decoration: BoxDecoration(
* color: Colors.grey,
* border: Border.all(color: Colors.black),
* ),
* child: Center(
* child: Text(
* '${numberList[index]}',
* style: const TextStyle(fontSize: 50),
* ),
* ),
* );
* },
* separatorBuilder: (BuildContext context, int index) {
* return const Divider();
* },
* ),

Jika kode di atas dijalankan, maka tampilan aplikasi adalah seperti ini:

Apabila Anda ingin memperdalam dan mempermudah pemahaman terhadap materi ini, silakan menyimak video pembahasan berikut.

Untuk mempelajari fitur ListView selengkapnya dapat Anda baca pada tautan dokumentasi [ListView Class](https://api.flutter.dev/flutter/widgets/ListView-class.html).