Mobile Programming Advanced

FLUTTER

Jeremia Tarigan

RESPONSIVE DAN ADAPTIVE UI

Topik

RESPONSIVE UI

Responsive UI

Responsive UI adalah pendekatan dalam pengembangan aplikasi yang memastikan tampilan dan pengalaman pengguna tetap optimal pada berbagai ukuran layar dan perangkat. Dalam Flutter, konsep responsive UI sangat penting karena memungkinkan aplikasi untuk tampil dan berfungsi dengan baik di berbagai perangkat, mulai dari ponsel kecil hingga tablet besar.

Responsive UI

Konsep Responsive UI

Responsive UI menyesuaikan tata letak dan elemen UI berdasarkan ukuran dan orientasi layar perangkat. Tujuannya adalah untuk memberikan pengalaman pengguna yang konsisten dan optimal di berbagai perangkat.

Teknik Membangun Responsive UI di Flutter

- Media Query
- LayoutBuilder
- Flexible dan Expanded
- AspectRatio

Responsive UI

Media Query

Di Flutter, MediaQuery adalah sebuah widget yang memberikan informasi tentang ukuran dan orientasi layar perangkat serta informasi lain terkait media yang sedang digunakan oleh aplikasi. Widget ini berguna untuk membuat tata letak responsif yang dapat menyesuaikan diri dengan berbagai ukuran layar dan orientasi.

Responsive UI

Fungsi Utama MediaQuery:

- Mengakses Informasi Media: MediaQuery memungkinkan Anda untuk mengakses berbagai informasi media seperti lebar dan tinggi layar, serta informasi lainnya seperti piksel per inci (pixel density) dan orientasi (portrait atau landscape).

- Responsif terhadap Perangkat: Dengan menggunakan MediaQuery, Anda dapat membuat tata letak aplikasi yang responsif. Misalnya, Anda bisa menentukan ukuran widget berdasarkan ukuran layar perangkat, atau menyesuaikan penempatan elemen-elemen UI berdasarkan orientasi layar.

- Penggunaan dalam Widget Tree: MediaQuery biasanya digunakan sebagai parent widget di mana informasi media yang diberikan dapat diakses oleh anak-anak widget di dalamnya. Ini memungkinkan anak-anak widget untuk menyesuaikan diri dengan kondisi layar tanpa perlu informasi langsung dari BuildContext.

Responsive UI

Contoh Penggunaan MediaQuery:

MediaQuery.of(context) digunakan untuk mendapatkan objek MediaQueryData yang berisi informasi tentang media.
MediaQuery.of(context).size.width digunakan untuk mengakses lebar layar perangkat.
Container diatur untuk memiliki lebar 80% dari lebar layar menggunakan MediaQuery.of(context).size.width * 0.8.

```dart
import 'package:flutter/material.dart';
void main() {
  runApp(MyApp());
}
class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('MediaQuery Example'),
        ),
        body: Center(
          child: Container(
            width: MediaQuery.of(context).size.width *
                0.5, // Menggunakan 80% lebar layar
            height: 200.0,
            color: Colors.blue,
            child: Center(
              child: Text(
                'Responsive Container',
                style: TextStyle(color: Colors.white, fontSize: 20.0),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

Responsive UI

Layout Builder

LayoutBuilder adalah sebuah widget di Flutter yang sangat berguna untuk membangun tata letak (layout) yang responsif dan dinamis berdasarkan ukuran widget induknya. Widget ini memungkinkan Anda untuk membuat tata letak yang berbeda tergantung pada lebar dan tinggi yang tersedia untuk widget tersebut.

Fungsi Utama Layout Builder

- Menggunakan Constraints: LayoutBuilder memberikan akses ke BoxConstraints dari widget induknya melalui parameter builder-nya. Constraints ini mencakup informasi seperti maxWidth, minWidth, maxHeight, dan minHeight.
- Responsif terhadap Ukuran: Dengan menggunakan informasi constraints yang tersedia, Anda dapat membuat keputusan terkait tata letak widget yang akan dibuat. Misalnya, Anda dapat memilih tata letak yang berbeda untuk layar yang lebih sempit atau lebih lebar.

Responsive UI

Fungsi Utama Layout Builder :

- Mendukung Tata Letak Dinamis: Memungkinkan pengembangan tata letak yang dinamis berdasarkan perubahan ukuran layar atau perangkat, sehingga aplikasi dapat menyesuaikan diri dengan baik terhadap berbagai resolusi layar.
- Flexibilitas dalam Pengembangan: LayoutBuilder memberikan fleksibilitas tambahan dalam pengaturan tata letak karena memungkinkan Anda untuk menyesuaikan elemen-elemen UI secara dinamis berdasarkan batas-batas (constraints) yang diberikan.

Responsive UI

Contoh Penggunaan Layout Builder

Dalam contoh ini, LayoutBuilder ditempatkan di dalam widget Center di dalam body dari Scaffold.
Di dalam builder dari LayoutBuilder, kita menggunakan BoxConstraints untuk memeriksa maxWidth dari constraints.
Berdasarkan nilai maxWidth, kita membangun tata letak yang berbeda:
Jika maxWidth kurang dari 600, kita menampilkan tata letak yang lebih sempit dengan latar belakang biru.
Jika maxWidth 600 atau lebih, kita menampilkan tata letak yang lebih lebar dengan latar belakang hijau.

```dart
class LayoutBuilderEx extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      title: 'LayoutBuilder Example',
      home: Scaffold(
        appBar: AppBar(
          title: Text('LayoutBuilder Example'),
        ),
        body: Center(
          child: LayoutBuilder(
            builder: (BuildContext context, BoxConstraints constraints) {
              if (constraints.maxWidth < 600) {
                // Jika lebar maksimum kurang dari 600, tampilkan tata letak yang lebih sempit
                return Container(
                  color: Colors.blue,
                  width: 200,
                  height: 200,
                  child: const Center(
                    child: Text(
                      'Narrow Layout',
                      style: TextStyle(color: Colors.white, fontSize: 24),
                    ),
                  ),
                );
              } else {
                // Jika lebar maksimum 600 atau lebih, tampilkan tata letak yang lebih lebar
                return Container(
                  color: Colors.green,
                  width: 400,
                  height: 400,
                  child: Center(
                    child: const Text(
                      'Wide Layout',
                      style: TextStyle(color: Colors.white, fontSize: 24),
                    ),
                  ),
                );
              }
            },
          ),
        ),
      ),
    );
  }
}
```

Responsive UI

Flexible dan Expanded

Di Flutter, "Flexible" dan "Expanded" adalah dua widget yang membantu dalam mengatur tata letak (layout) dari elemen-elemen di dalam aplikasi Flutter.

Responsive UI

Flexible

- Mendukung Tata Letak Dinamis: Memungkinkan pengembangan tata letak yang dinamis berdasarkan perubahan ukuran layar atau perangkat, sehingga aplikasi dapat menyesuaikan diri dengan baik terhadap berbagai resolusi layar.
- Flexibilitas dalam Pengembangan: LayoutBuilder memberikan fleksibilitas tambahan dalam pengaturan tata letak karena memungkinkan Anda untuk menyesuaikan elemen-elemen UI secara dinamis berdasarkan batas-batas (constraints) yang diberikan.

Responsive UI

Flexible

- Flexible digunakan di dalam Row untuk mengatur bagaimana child-widget (Container) akan mendapatkan ruang yang tersedia.
- Pada contoh pertama, terdapat dua Container di dalam sebuah Row. Container pertama memiliki flex 2, sementara Container kedua memiliki flex 1. Ini berarti Container pertama akan mendapatkan dua per tiga dari lebar Row, sedangkan Container kedua akan mendapatkan satu per tiga.
- Pada contoh kedua, urutan flex diubah (Container pertama memiliki flex 1 dan Container kedua memiliki flex 2), sehingga distribusi ruang di Row berubah: Container pertama mendapatkan satu per tiga, sementara Container kedua mendapatkan dua per tiga.

```dart
class FlexibleEx extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Flexible Example'),
        ),
        body: Column(
          children: <Widget>[
            Container(
              height: 100,
              color: Colors.blue,
              child: Row(
                children: <Widget>[
                  Flexible(
                    flex: 2,
                    child: Container(
                      color: Colors.green,
                    ),
                  ),
                  Flexible(
                    flex: 1,
                    child: Container(
                      color: Colors.red,
                    ),
                  ),
                ],
              ),
            ),
            Container(
              height: 100,
              color: Colors.yellow,
              child: Row(
                children: <Widget>[
                  Flexible(
                    flex: 1,
                    child: Container(
                      color: Colors.orange,
                    ),
                  ),
                  Flexible(
                    flex: 2,
                    child: Container(
                      color: Colors.purple,
                    ),
                  ),
                ],
              ),
            ),
```

Expanded

- Widget Expanded mirip dengan Flexible, tetapi dengan flex yang telah ditetapkan menjadi 1 secara otomatis. Ini berarti Expanded widget mengambil semua ruang yang tersedia di dalam parent widget-nya, memperluas child widget-nya sesuai dengan ruang yang tersedia. Expanded biasanya digunakan sebagai child dari Row, Column, atau Flex untuk menjamin bahwa widget tertentu akan mengambil sebanyak mungkin ruang yang tersedia dalam arah utama dari widget parent-nya.

Responsive UI

Expanded

- Column adalah widget yang mengatur anak-anaknya dalam satu kolom vertikal.
- Di dalam Column, terdapat tiga Container dengan warna biru, merah, dan hijau.
- Widget Expanded digunakan di antara dua Container (warna merah) untuk memungkinkan Container tersebut mengisi sisa ruang vertikal yang tersedia dalam Column.
- Expanded mengambil sebanyak mungkin ruang dalam arah utama dari parent-nya (vertikal dalam kasus ini).
- Hal ini membuat Container warna merah memperluas dirinya untuk mengisi sisa ruang di antara Container biru dan hijau.

```dart
class ExpandedEx extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Expanded Widget Example'),
        ),
        body: Column(
          children: <Widget>[
            Container(
              color: Colors.blue,
              height: 100,
              width: 100,
            ),
            Expanded(
              child: Container(
                color: Colors.red,
                width: 100,
              ),
            ),
            Container(
              color: Colors.green,
              height: 100,
              width: 100,
            ),
          ],
        ),
      ),
    );
  }
}
```

Responsive UI

Perbedaan Utama

- Flexible: Memungkinkan Anda untuk menentukan flex yang berbeda-beda untuk setiap child widget di dalam satu baris atau kolom. Fleksibilitas ini memungkinkan pengaturan yang lebih halus terhadap bagaimana ruang tersedia didistribusikan di antara child widget.
- Expanded: Mengambil semua ruang yang tersedia, yang setara dengan menetapkan flex: 1 pada widget Flexible. Ini berguna ketika Anda ingin widget tertentu untuk mengisi atau memperluas ruang yang tersedia sebanyak mungkin dalam satu arah.

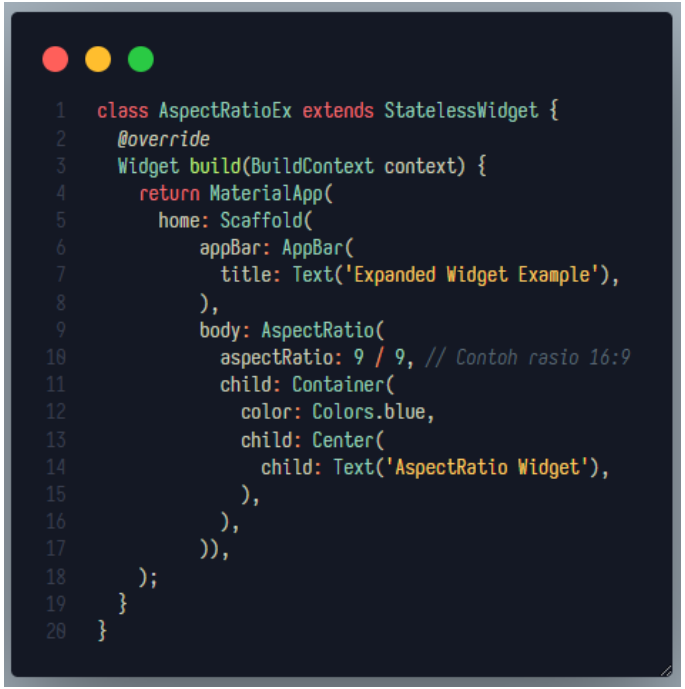
Aspect Ratio

Di Flutter, AspectRatio adalah sebuah widget yang digunakan untuk mengatur proporsi aspek dari child widget-nya. Widget ini memungkinkan Anda untuk menentukan rasio antara lebar dan tinggi dari child widget tersebut.

Responsive UI

Aspect Ratio

- Dalam contoh di atas, AspectRatio memiliki child berupa Container dengan warna biru dan teks di tengah. Properti aspectRatio diatur menjadi 16/9, yang berarti child widget akan memiliki rasio aspek 16:9, sehingga akan mengikuti proporsi ini terlepas dari ukuran layar atau ukuran widget tersebut.
- Widget AspectRatio berguna ketika Anda ingin menjaga proporsi aspek dari sebuah widget, seperti video player, gambar, atau elemen UI lainnya yang harus tetap memiliki rasio aspek tertentu untuk tampilan yang konsisten.



```dart
class AspectRatioEx extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
          appBar: AppBar(
            title: Text('Expanded Widget Example'),
          ),
          body: AspectRatio(
            aspectRatio: 9 / 9, // Contoh rasio 16:9
            child: Container(
              color: Colors.blue,
              child: Center(
                child: Text('AspectRatio Widget'),
              ),
            ),
          )),
    );
  }
}
```

Topik

ADAPTIVE UI

ADAPTIVE UI

Adaptive UI adalah pendekatan dalam pengembangan aplikasi yang memungkinkan tampilan dan perilaku UI untuk beradaptasi secara dinamis dengan kondisi atau karakteristik tertentu dari perangkat atau platform yang digunakan pengguna. Dalam konteks Flutter, adaptive UI sering kali mengacu pada kemampuan aplikasi untuk menyesuaikan diri dengan platform spesifik seperti Android atau iOS, serta karakteristik perangkat seperti ukuran layar, mode orientasi (portrait atau landscape), atau bahkan preferensi pengguna seperti tema gelap atau terang.

Adaptive UI

Ciri Ciri Adaptive UI

- Platform-specific UI Elements: Menggunakan widget dan komponen UI yang spesifik untuk platform tertentu, misalnya menggunakan Cupertino untuk iOS dan Material untuk Android.
- Respons terhadap Orientasi: Aplikasi dapat menyesuaikan tata letaknya secara otomatis saat perangkat beralih dari mode potret ke mode lanskap atau sebaliknya.
- Respons terhadap Ukuran Layar: Mengatur tata letak, ukuran teks, dan elemen UI lainnya agar sesuai dengan ukuran layar perangkat yang berbeda.
- Tema dan Warna: Memungkinkan aplikasi untuk mengubah tema atau warna berdasarkan preferensi pengguna atau pengaturan sistem.

Adaptive UI

Teknik Membangun Adaptive UI di Flutter:

- Platform Detection
- Platform-specific Layouts
- Respons terhadap Orientasi

Responsive UI

Platform Detection

- Theme.of(context).platform digunakan untuk mendapatkan platform saat ini dari tema yang digunakan dalam aplikasi Anda.
- CupertinoButton digunakan untuk menampilkan tombol gaya iOS.
- ElevatedButton digunakan untuk menampilkan tombol gaya Material (Android).

```dart
class PlatformDetectionEx extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    TargetPlatform platform = Theme.of(context).platform;
    return MaterialApp(
        home: Scaffold(
      body: Center(
        child: platform == TargetPlatform.iOS
            ? CupertinoButton(
                child: Text('Ini tombol iOS'),
                onPressed: () {},
              )
            : ElevatedButton(
                child: Text('Ini tombol Android'),
                onPressed: () {},
              ),
      ),
    ));
  }
}
```

Responsive UI

Platform-specific Layouts

- Theme.of(context).platform digunakan untuk mendapatkan platform saat ini dari tema yang digunakan dalam aplikasi.
- TargetPlatform digunakan untuk menyimpan platform saat ini yang didapatkan dari tema aplikasi.

```dart
class PlatformApecificLayoutsEx extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    TargetPlatform platform = Theme.of(context).platform;
    return MaterialApp(
        home: Scaffold(
      appBar: AppBar(
        title: Text('Adaptive Layout'),
      ),
      body: Center(
        child: LayoutBuilder(
          builder: (context, constraints) {
            if (constraints.maxWidth < 600) {
              // Tampilan untuk layar ponsel
              return Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Icon(Icons.phone_android, size: 100),
                  Text('Aplikasi Android'),
                ],
              );
            } else {
              // Tampilan untuk layar tablet atau lebih besar
              return Column(
                mainAxisAlignment: MainAxisAlignment.center,
                children: [
                  Icon(Icons.tablet_android, size: 100),
                  Text('Aplikasi Android Tablet'),
                ],
              );
            }
          },
        ),
      ),
    ));
  }
}
```

Responsive UI

Respons terhadap Orientasi

- OrientationBuilder adalah widget yang memungkinkan kita untuk membangun tampilan yang berbeda berdasarkan orientasi perangkat (potret atau lanskap).
- builder callback dari OrientationBuilder menerima dua parameter: context (BuildContex) dan orientation (Orientation).
- Pada contoh ini, builder akan menentukan tampilan berdasarkan nilai orientation yang diterima. Jika orientation adalah Orientation.portrait, maka akan menampilkan teks "Mode Potret". Jika orientation adalah Orientation.landscape, maka akan menampilkan teks "Mode Lanskap".

```dart
class AdaptiveOrientationLayoutsEx extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: Text('Adaptive Orientation'),
        ),
        body: OrientationBuilder(
          builder: (context, orientation) {
            return Center(
              child: orientation == Orientation.portrait
                  ? Text('Mode Potret')
                  : Text('Mode Lanskap'),
            );
          },
        ),
      ),
    );
  }
}
```

Adaptive UI

Manfaat Adaptive UI

- Pengalaman Pengguna yang Konsisten: Memastikan aplikasi terlihat dan berperilaku sesuai dengan harapan pengguna terlepas dari perangkat yang mereka gunakan.
- Fleksibilitas dalam Pengembangan: Memungkinkan pengembang untuk fokus pada pengoptimalan UI berdasarkan karakteristik unik dari setiap platform atau perangkat.
- Peningkatan Retensi Pengguna: Menyediakan pengalaman yang lebih menyenangkan dan mudah digunakan bagi pengguna dari berbagai latar belakang teknologi.

TERIMA KASIH....