#02 | Konsep Widget di Flutter

| **Summary** | Modul kedua ini berisi konsep widget pada flutter, rendering, composability, immutability, widget tree. |
| --- | --- |
| **URL** | 02-konsep-widget-flutter |
| **Category** | Mobile Flutter |
| **Environment** | Flutter, mobile, android, iOS, web, desktop |
| **Status** | Published |
| **Feedback Link** | https://github.com/jti-polinema/flutter-codelab/issues |
| **Author** | Tim Pengajar Mobile TSA 2022 |
| **Author LDAP** | hbb-polinema |
| **Analytics Account** | G-H7507PFEJB |

[Konsep Widget di Flutter](#_ok7k5uux6)

[Apa itu Widget ?](#_5hnrehwfd5fx)

[Apa yang Anda akan pelajari](#_21yzqg98x7h6)

[Apa yang Anda perlu persiapkan](#_hs6hf8vnv2d6)

[Pengetahuan yang Anda harus dimiliki](#_319h1e1sjzdz)

[Widgets, widgets, everywhere](#_jdmzwkcs72o)

[Flutter rendering](#_zid1uwvx8dga)

[Flutter – rendering by itself](#_bwo0af2iwk90)

[Composability](#_q6selyhmjlab)

[Composition vs inheritance](#_2vy0wczdnhpm)

[Immutability](#_vwx8i5py6zhb)

[Everything is a widget](#_9y6uexmy9v0x)

[The widget tree](#_5sgy79scn2bj)

[Tugas Mandiri](#_qqpq7vxs7rkc)

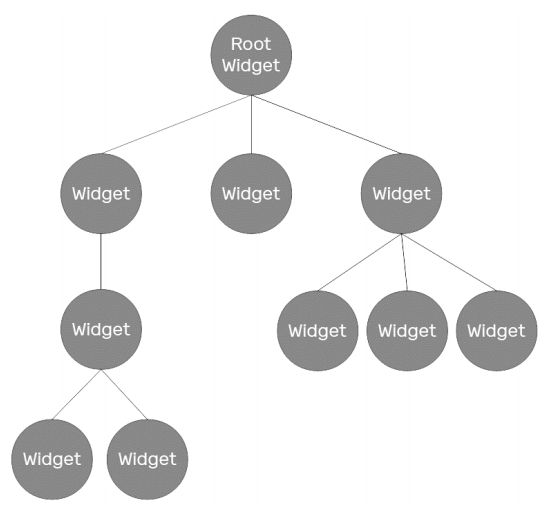
[Selamat](#_s22qmcj4zu61)

[Apa selanjutnya?](#_9mqifffit2ew)

[Referensi](#_an95hoqxrlhg)

# Konsep Widget di Flutter

Duration: 30:00



**Terakhir diperbarui:** 09 Agustus 2022

**Penulis:** Tim Pengajar Mobile

## Apa itu Widget ?

Gaya pengembangan aplikasi menggunakan flutter sedikit berbeda dengan gaya pengembangan aplikasi pada umumnya, dimana UI pada flutter dibuat menggunakan Widget.

[](https://www.youtube.com/watch?v=b_sQ9bMltGU)

**Widget adalah** sebuah konsep dimana UI dapat dianggap sebagai sebuah balok LEGO, sebuah bentuk baru dapat disusun dari beberapa balok dan masing masing kumpulan balok dapat dikombinasikan dengan kumpulan balok lain sehingga membentuk sebuah bentuk baru yang lebih kompleks.

Flutter menggunakan widget ini sebagai balok dasar pembangunan aplikasi. Widget dapat disusun dan dikombinasikan dalam satu layar, sama halnya dengan xml pada pemrograman android native, widget dapat disusun dalam bentuk tree dimana satu widget menjadi parent dan widget lain menjadi child. Masing masing widget dapat diberikan konfigurasi sesuai dengan kebutuhan aplikasi.

**Flutter memiliki dua jenis widget yaitu StatelessWidget dan StatefulWidget.** Stateless widget digunakan ketika value (state/konfigurasi) dari widget tersebut tidak pernah berubah, dan StatefullWidget digunakan ketika value (state/konfigurasi) dari widget dapat berubah. Baik StatelesWidget maupun StatefullWidget sama-sama memiliki sebuah method bernama “build” yang memiliki BuildContext untuk mengatur posisi widget di dalam widget tree, detail mengenai widget dan bagaimana membuatnya akan dibahas pada bab selanjutnya.

[](https://www.youtube.com/watch?v=4AoFA19gbLo)

## Apa yang Anda akan pelajari

Di codelab ini Anda akan mempelajari tentang:

* Konsep widget pada flutter
* Flutter rendering
* Composability
* Immutability
* The widget tree

| **Catatan:** Materi ini diadaptasi dari Buku Flutter for Beginners dan Buku Ajar Pemrograman Mobile Flutter JTI Polinema. |
| --- |

## Apa yang Anda perlu persiapkan

* PC atau Laptop dengan spesifikasi minimum RAM 4GB Processor Core i3
* Koneksi internet
* Chrome Browser

## Pengetahuan yang Anda harus dimiliki

* Algoritma dan Pemrograman Dasar
* Pemrograman Berorientasi Objek

# Widgets, widgets, everywhere

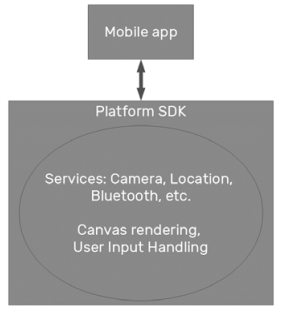
Duration: 1:30

Widget Flutter adalah bagian inti dari framework dan digunakan terus-menerus di seluruh kode. Anda akan mendengar pepatah "Semuanya adalah widget," dan itu benar di Flutter. Di bagian ini, kita akan melihat bagaimana Flutter merender antarmuka pengguna dan bagaimana Flutter menerapkan ide widget ke pengembangan aplikasi untuk membuat UI yang mengagumkan. Widget dapat dipahami sebagai representasi visual (tetapi tidak hanya itu) dari bagian-bagian aplikasi. Banyak widget disatukan untuk menyusun UI aplikasi. Anda dapat membayangkan widget di Flutter seperti bermain puzzle.

Tujuan widget adalah untuk menyediakan cara agar aplikasi Anda menjadi modular, dapat diskalakan (*scalable*), dan ekspresif dengan lebih sedikit kode dan tanpa batasan. Karakteristik utama dari UI widget di Flutter adalah *composability* dan *immutability* (kekal/tidak pernah berubah/statis).

## Flutter rendering

Salah satu aspek utama yang membuat Flutter unik adalah cara menggambar visual komponen ke layar. Pembeda utama untuk kerangka kerja yang ada adalah bagaimana aplikasi berkomunikasi dengan SDK platform, apa yang diminta SDK untuk dilakukan, dan dilakukan dengan sendirinya:

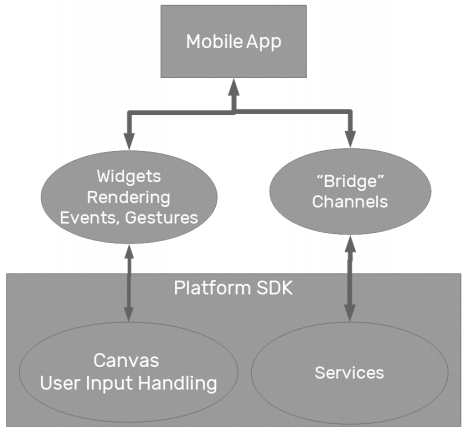


Platform SDK dapat dilihat sebagai antarmuka antara aplikasi, sistem operasi dan layanan. Setiap sistem menyediakan SDK-nya sendiri dengan kemampuan yang dimiliki dan berdasarkan bahasa pemrograman yang digunakan (yaitu, Kotlin/Java untuk Android SDK dan Swif/Objective C untuk iOS SDK).

# Flutter – rendering by itself

Duration: 30:00

Flutter memilih untuk melakukan semua pekerjaan rendering dengan sendirinya. Satu-satunya hal yang dibutuhkan dari SDK platform adalah akses ke **Services API** dan kanvas untuk menggambar UI :



Flutter memindahkan widget dan rendering ke aplikasi, di mana ia dapat disesuaikan dan ditambahkan (*extensibility*). Melalui kanvas, ia dapat menggambar apa saja dan juga dapat mengakses *event* untuk menangani masukan dan gerakan pengguna dengan sendirinya. Jembatan di Flutter dilakukan melalui platform channel.

| **Catatan:** Untuk memahami lebih baik tentang cara flutter melakukan *rendering*, silakan tonton video dari Flutter YouTube Channel di bawah ini.  https://www.youtube.com/watch?v=996ZgFRENMs |
| --- |

# Composability

Duration: 10:00

Untuk struktur widget antarmuka pengguna (UI), Flutter memilih komposisi daripada inheritance, dengan tujuan menjaga setiap widget tetap sederhana dan dengan tujuan yang jelas. Salah satu tujuan utama kerangka kerja, yaitu fleksibilitas, Flutter memungkinkan pengembang untuk membuat banyak kombinasi agar dapat mencapai hasil yang luar biasa.

## Composition vs inheritance

Inheritance menurunkan satu kelas dari yang lain. Misalnya, Anda mungkin memiliki kelas seperti Kendaraan, serta memiliki subkelas Mobil dan Sepeda Motor. Kelas Mobil dan Sepeda Motor akan mewarisi kemampuan kelas Kendaraan dan kemudian menambahkan spesialisasinya sendiri. Dalam hal ini, Mobil adalah Kendaraan dan Sepeda Motor adalah Kendaraan.

Komposisi mendefinisikan kelas sebagai jumlah dari bagian-bagiannya. Misalnya, Anda mungkin memiliki kelas Engine dan kelas Roda. Dalam model ini, sebuah Mobil terdiri dari Mesin, Roda empat, dan spesialisasi lainnya; Mobil memiliki Mesin dan Mobil memiliki Roda. Inheritance lebih fleksibel daripada Composition. Composition memungkinkan untuk hal-hal seperti injeksi dependensi dan modifikasi saat runtime.

# Immutability

Duration: 10:00

Flutter didasarkan pada gaya pemrograman reaktif, di mana instance widget berumur pendek dan mengubah deskripsinya (baik secara visual atau tidak) berdasarkan perubahan konfigurasi, sehingga ia bereaksi terhadap perubahan dan menyebarkan perubahan ini ke widget penyusunnya, dan seterusnya.

Widget Flutter mungkin memiliki state yang saling berelasi, dan saat state yang berelasi berubah, widget akan dibuat ulang agar sesuai dengan representasinya.

Istilah **state** dan **reaktif** diperkenalkan dalam gaya pemrograman React, yang library-nya dibuat oleh Facebook.

# Everything is a widget

Duration: 10:00

Widget Flutter ada di mana-mana dalam aplikasi. Mungkin tidak semuanya adalah widget, tapi hampir semuanya. Bahkan aplikasinya adalah widget di Flutter, dan itulah mengapa konsep ini begitu penting. Widget mewakili bagian dari UI, tetapi itu tidak berarti itu hanya sesuatu yang terlihat. Ini dapat berupa salah satu dari berikut ini:

* Elemen visual/struktural yang merupakan elemen struktural dasar, seperti Tombol atau Widget teks
* Elemen khusus tata letak yang dapat menentukan posisi, margin, atau padding, seperti widget Padding
* Sebuah elemen gaya yang dapat membantu untuk mewarnai dan tema elemen visual/struktural, seperti widget Tema
* Elemen interaksi yang membantu merespons interaksi pengguna dengan cara yang berbeda, seperti widget GestureDetector

Mari kita lihat sekilas contoh widget berikut sehingga Anda bisa memahami apa yang dimaksud. Silakan buka IDE VScode Anda dan lihat pada file lib/main.dart. Anda akan melihat bagian seperti ini:

| class MyApp extends StatelessWidget {  Widget build(BuildContext context) {  return MaterialApp(  title: 'Flutter Demo',  theme: ThemeData(  primarySwatch: Colors.blue,  ),  home: const MyHomePage(title: 'Flutter Demo Home Page'),  );  }  } |
| --- |

Tidak hanya ini contoh pertama Anda dari widget Flutter, ini juga merupakan kesempatan pertama Anda untuk melihat bahasa pemrograman Dart. Jika Anda mempunyai dasar pemograman bahasa Java, C++, Objective-C, dan sebagainya, maka itu akan terlihat relatif akrab bagi anda. Komponen kode disimpan dalam definisi Class yang berisi Fields dan methods, dengan inheritance melalui kata kunci extends.

MyApp class ini di-run sebagai aplikasi dan ia sendiri merupakan widget. Dalam hal ini, itu adalah StatelessWidget, seperti yang Anda lihat pada bagian extends. Kita akan mengeksplorasi StatelessWidgets (dan lawannya, yaitu StatefulWidget) yang nanti akan dipelajari lebih mendalam, tetapi untuk saat ini, cukup mengetahui bahwa StatelessWidget tidak memiliki state, yang juga dapat dimanfaatkan untuk menggabungkan dengan widget lain.

Salah satu poin penting yang perlu diperhatikan adalah method build. Method ini digunakan untuk memperbarui tampilan dan dipanggil ketika beberapa aktivitas eksternal terjadi – misalnya, pengguna berinteraksi dengan perangkat, beberapa data yang dikirim dari database, atau sebuah timer yang dipicu pada waktu yang telah ditentukan.

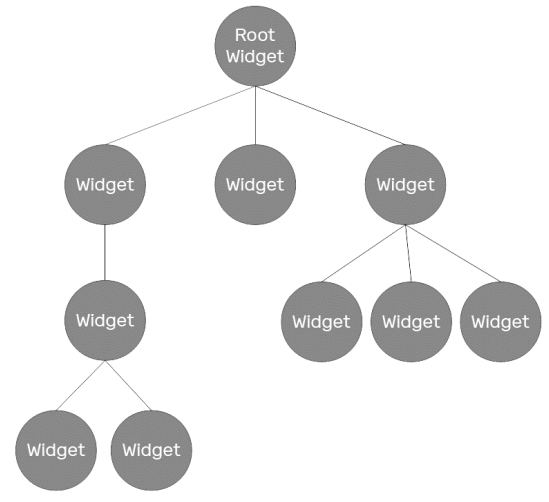
Dalam method build ini, widget MyApp hanya mengembalikan widget lain, yaitu MaterialApp, yang sebenarnya ia juga memiliki method build yang mungkin juga dapat me-return beberapa widget. Pada akhirnya, Anda akan mendapatkan widget yang akan merender grafis pada tampilan.

Widget adalah blok bangunan dasar dari sebuah antarmuka. Untuk membangun UI dengan benar, Flutter mengatur widget di dalam bentuk sebuah widget tree.

# The widget tree

Duration: 10:00

Ini adalah konsep penting lainnya dalam layout Flutter. Di situlah widget menjadi hidup. widget tree ini adalah representasi logika dari semua widget UI. Ini dihitung selama **layout** (pengukuran dan info struktural) dan digunakan selama **rendering (**frame to screen**)** dan **hit testing** (interaksi sentuh), dan inilah hal terbaik yang dilakukan Flutter. Dengan menggunakan banyak algoritma optimasi, ia mencoba untuk memanipulasi pohon sesedikit mungkin, mengurangi jumlah total pekerjaan yang dihabiskan untuk melakukan rendering, yang bertujuan untuk efisiensi yang lebih besar:



Widget direpresentasikan tree sebagai node. Setiap widget mungkin memiliki state yang saling terkait dengan lainnya; setiap terjadi perubahan pada state, itu akan menghasilkan penyusunan ulang pada widget dan child yang terlibat didalamnya.

Seperti yang Anda lihat, struktur tree's child itu tidak statis, dan telah ditentukan oleh masing-masing widget-nya. Relasi children di dalam widget membentuk sebuah pohon UI; hal itu terjadi karena dibuat dengan komposisi.

Widget tree tidak bekerja sendiri di dalam sebuah framework. Ia dibantu dengan elemen tree; sebuah tree memiliki relasi dengan tree widget yang merepresentasikan susunan widget pada layar, jadi setiap widget akan memiliki elemen yang sesuai di dalam elemen tree setelah dilakukan proses build.

Tree elemen memiliki tugas penting di Flutter. Ini membantu untuk memetakan elemen pada layar ke bentuk tree widget. Ini juga menentukan bagaimana rebuilding widget dilakukan ketika skenario update. Ketika sebuah widget berubah dan perlu dibangun ulang, ini akan menyebabkan pembaruan yang sesuai pada elemen. Elemen ini menyimpan jenis widget yang sesuai beserta referensinya pada elemen children. Ketika dilakukan perubahan posisi, misalnya, widget, elemen akan melakukan pemeriksaan pada jenis widget baru yang sesuai, dan jika ada yang sesuai, itu akan memperbarui sendiri dengan deskripsi widget yang baru.

Element tree dapat dianggap sebagai pra-render auxiliary tree ke bentuk widget tree.

Jika Anda memerlukan informasi lebih lanjut tentang itu, Anda dapat memeriksa dokumen resmi di <https://api.flutter.dev/flutter/widgets/Element-class.html>

# Tugas Mandiri

Duration: 90:00

1. Carilah kode aplikasi yang dibangun dengan Flutter di GitHub, lalu buatlah satu diagram tree yang menjelaskan salah satu file kode Dart!
2. Jelaskan diagram yang telah Anda buat berdasarkan jawaban dari soal nomor satu!

Kumpulkan dalam bentuk file PDF ke LMS yang telah disediakan.

# Selamat

Duration: 0:00

Selamat Anda telah menyelesaikan Codelab ini sebagai pengetahuan dasar tentang pemahaman widget pada framework flutter.

Pada codelab berikutnya, Anda akan melakukan mode debug, release, profile, dan running sesuai target platform. Anda juga akan mempelajari konsep dari file pubspec.yaml termasuk mengelola aset dan library dependencies.

## Apa selanjutnya?

Silakan cek beberapa sumber belajar lainnya…

* [Flutter for Beginners Second Edition](https://drive.google.com/file/d/1hJbctx-_O3B6kUeG70bJD5xfbnsWwBXY/view?usp=sharing)
* [FlutLab is a modern Flutter online IDE](https://flutlab.io/)

## Referensi

* <https://www.youtube.com/flutterdev>
* Buku Ajar Pemrograman Mobile Dengan Flutter, Jurusan Teknologi Informasi, Politeknik Negeri Malang. 2021.