**Framework Flutter**

Teknologi sudah menjadi bagian tak terpisahkan dalam hidup kita terutama *smartphone*. Menurut data dari *We Are Social*, pada bulan Januari 2020 pengguna internet di Indonesia mencapai sekitar 175,4 juta orang. Terdapat peningkatan sebesar 17% atau sekitar 25 juta pengguna dari tahun 2019. Jika dilihat dari total populasi penduduk Indonesia yang berkisar 272,1 juta jiwa, maka 64% di antaranya sudah mengakses internet. Dengan masifnya angka pengguna *smartphone* ini, kian spesifik pula *requirements* alias permintaan bagi seorang pengembang aplikasi. Termasuk *requirement* pada perangkat *mobile* dengan target platform Android dan iOS. Masing-masing platform memiliki aplikasi *native*-nya tersendiri.

Mengembangkan dua platform memiliki banyak implikasi. Ditinjau dari sisi bisnis, biaya yang perlu dikeluarkan tentunya tidak murah. Dua aplikasi berarti ada dua tipe developer yang perlu direkrut dan dua jenis pengembangan untuk dikelola. Jika ada teknologi yang memungkinkan pengembangan aplikasi di beberapa platform sekaligus hanya dengan satu codebase tentunya lebih hemat bukan? Pasti lebih efisien dari sisi biaya, waktu, dan tenaga.

Tertarik untuk mempelajari *stack* teknologi yang bisa mewujudkannya? Mari kita berkenalan dengan Flutter. Inilah *framework* populer dan powerful buatan Google, Flutter hadir untuk mengembangkan aplikasi multi-platform yang dikompilasi secara *native*.

## Apa itu Flutter

Flutter adalah SDK (Software Development Kit) yang dikembangkan oleh Google untuk membuat aplikasi yang bagus dan bisa berjalan pada berbagai platform. Flutter 2 yang merupakan versi terbaru memberikan dukungan pada Anda untuk membangun aplikasi pada sistem operasi Android, iOS, Web, Windows, Linux, dan MacOS. Dengan ini, Anda cukup sekali coding atau dikenal dengan single codebase. Flutter juga sudah digunakan oleh banyak developer maupun organisasi di seluruh dunia, selain itu Flutter bersifat open source.

### Mengapa Flutter?

Flutter berbeda dari kebanyakan SDK Cross-platform lainnya untuk membuat aplikasi mobile. Untuk menarik widget, Flutter bukan menggunakan WebView maupun widget OEM, melainkan mesin rendering berkinerja tinggi. Flutter dapat digunakan bersamaan dengan aplikasi native yang sudah ada atau digunakan secara keseluruhan untuk aplikasi baru.

Ada beberapa kelebihan dari Flutter, antara lain:

1. **Flutter memungkinkan kita untuk membuat aplikasi yang indah (beautiful)**  
   Desainer dapat dengan bebas berkreasi tanpa adanya banyak batasan dari framework. Flutter juga dapat mengontrol setiap piksel yang ada di layar, sehingga memudahkan dalam membuat animasi. Flutter menyediakan banyak komponen material design yang dapat berjalan baik pada Android, iOS, dan web.
2. **Flutter berjalan dengan sangat cepat (fast)**  
   Flutter menggunakan graphic engine bernama Skia-2D yang juga digunakan pada Chrome dan Android. Kode Flutter menggunakan bahasa Dart, yang memungkinkan untuk dikompilasi ke kode native 32-bit dan 64-bit ARM untuk iOS dan Android.
3. **Flutter sangat produktif (productive)**  
   Flutter memiliki fitur hot-reload yang memungkinkan Anda untuk melihat hasil kompilasi secara real-time. Dengan hot-reload, Anda dapat dengan mudah melihat perubahan kode pada perangkat, tanpa perlu menunggu restart dan kehilangan state.
4. **Flutter bersifat terbuka (open source)**  
   Flutter adalah proyek open source dengan lisensi BSD. Kode yang ada di Flutter berasal dari kontribusi ratusan developer dari seluruh dunia. Banyak plugin yang sudah dibuat oleh developer. Anda dapat ikut berkontribusi mengembangkan Flutter dengan melaporkan bug/issue atau ikut memperbaiki kode yang sudah ada. Source code dari Flutter dapat Anda temukan pada tautan <https://github.com/flutter/flutter>.

### Sejarah Perkembangan Flutter

Versi pertama dari Flutter bernama Sky yang diumumkan pada Dart Developer Summit 2015. Pada saat itu, Sky hanya berjalan pada sistem operasi Android.

Lalu, pada Google Developer Days di Shanghai, Google mengumumkan Flutter versi Release Preview 2 yang merupakan rilis sebelum versi stable 1.0.

Pada 3 Maret 2021 kemarin, Flutter mengumumkan versi Flutter 2 dengan pembaruan yang cukup besar, yaitu dukungan Flutter untuk platform web dan desktop (Windows, Linux, dan macOS) yang memasuki versi stable.

Sejak rilis beta pertama, Flutter sudah banyak digunakan untuk mengembangkan aplikasi mobile. Contohnya adalah perusahaan Abbey Road Studios, Alibaba, Capital One, Groupon, Hamilton, JD.com, Philips Hue, Reflectly, Tencent, dan GITS Indonesia.

### Dart

Seperti yang telah kita ketahui, aplikasi Flutter ditulis dengan bahasa Dart. Seharusnya Anda telah mempelajari tentang fundamental Dart di kelas [Memulai Pemrograman dengan Dart](https://www.dicoding.com/academies/191). Untuk menyegarkan kembali ingatan Anda kami akan mengulas sedikit tentang Dart.

Bahasa pemrograman Dart merupakan bahasa pemrograman general-purpose yang dirancang oleh Lars Bak dan Kasper Lund. Bahasa pemrograman ini dikembangkan sebagai bahasa pemrograman aplikasi yang dapat dengan mudah dipelajari dan disebarkan.

Bahasa pemrograman besutan Google ini dapat digunakan untuk mengembangkan berbagai macam platform termasuk di dalamnya adalah web, aplikasi mobile, server, dan perangkat yang mengusung teknologi Internet of Things (IoT).

Bahasa pemrograman tersebut dapat digunakan untuk mengembangkan aplikasi yang dijalankan pada berbagai macam peramban (browser) modern. Dart juga dapat digunakan untuk mengembangkan aplikasi dari codebase tunggal menjadi aplikasi Android maupun iOS.

Bahasa pemrograman Dart dapat digunakan secara bebas oleh para developer karena bahasa ini dirilis secara open-source oleh Google di bawah lisensi BSD. Bahasa pemrograman Dart merupakan bahasa pemrograman berbasis kelas dan berorientasi terhadap objek dengan menggunakan Syntax bahasa pemrograman C.

Bahasa ini dikenalkan oleh Google sebagai pengganti bahasa pemrograman JavaScript, akan tetapi secara opsional bahasa ini dapat dikompilasi ke dalam JavaScript dengan menggunakan Dart-to-JavaScript compiler. Sedikit berbeda dengan bahasa pemrograman JavaScript yang bertipe dinamis, bahasa pemrograman Dart merupakan bahasa pemrograman bertipe statis.

Oke, sampai sini kita telah mempelajari tentang apa itu Flutter, termasuk sejarah dan kenapa kita perlu belajar mengembangkan aplikasi dengan Flutter. Di kelas ini Anda akan belajar dasar pengembangan aplikasi Flutter, seperti package, widget yang umum digunakan, cara berpindah halaman menggunakan Navigation, hingga proses build menjadi APK yang bisa diinstal pada smartphone Anda

Catatan: Materi-materi di kelas ini akan disampaikan dalam banyak opsi, mulai dari sistem operasi (OS), platform Android atau iOS, dan Integrated Development Environment (IDE). Silakan sesuaikan dengan spesifikasi hardware/software Anda dan lewatilah materi yang dirasa tidak diperlukan.

Untuk lulus dari kelas ini cukup dengan mengirimkan submission dari satu platform saja, misal Anda memilih platform Android maka tidak perlu mengirimkan versi iOS atau webnya.

Mari kita lanjut ke materi selanjutnya, yaitu instalasi Flutter.

## Instalasi Flutter

Sebelum mulai mengembangkan aplikasi menggunakan Flutter, tentunya kita perlu menyiapkan dan menginstal tools apa saja yang dibutuhkan untuk membuat aplikasi Flutter. Untuk melakukan instalasi Flutter, ada perbedaan cara di setiap sistem operasi. Simak caranya berikut ini sesuai dengan sistem operasi yang Anda gunakan:

### Persyaratan Minimum

Sebelum melakukan instalasi dan menjalankan Flutter, perangkat milik Anda harus memenuhi persyaratan minimum seperti di bawah ini:

* [**Windows**](https://www.dicoding.com/academies/159/tutorials/6446?from=6444#tab1-code1)
* [MacOS](https://www.dicoding.com/academies/159/tutorials/6446?from=6444#tab1-code2)
* [Linux](https://www.dicoding.com/academies/159/tutorials/6446?from=6444#tab1-code3)
* Sistem Operasi : Windows 7 SP1 atau lebih baru (64-bit).
* Ruang Penyimpanan : 1.64 GB dan tidak termasuk IDE dan tools lainnya.
* Flutter bergantung pada tools yang ada di dalam environment:
  + Windows Powershell 5.0 atau versi terbaru (sudah terdapat pada paket instalasi Windows 10), dapat Anda unduh pada link berikut ini <https://docs.microsoft.com/en-us/powershell/scripting/install/installing-windows-powershell>.
  + Git for Windows 2.x, dengan opsi "Use Git from the Windows Command Prompt", dapat Anda unduh pada link berikut ini <https://git-scm.com/download/win>.

Jika Git untuk Windows sudah diinstal, pastikan Anda dapat menjalankan perintah git dari Command Prompt (CMD) atau PowerShell.

## Flutter SDK

Setelah persyaratan minimum terpenuhi, Anda dapat mulai melakukan instalasi Flutter SDK. Cara untuk mendapatkan Flutter SDK sebagai berikut:

* [**Windows**](https://www.dicoding.com/academies/159/tutorials/6446?from=6444#tab2-code1)
* [MacOS](https://www.dicoding.com/academies/159/tutorials/6446?from=6444#tab2-code2)
* [Linux](https://www.dicoding.com/academies/159/tutorials/6446?from=6444#tab2-code3)

1. Unduh paket instalasi untuk mendapatkan versi stabil terbaru dari Flutter SDK di alamat web <https://flutter.dev/docs/development/tools/sdk/releases>. Ambil versi terbaru pada stable channel sesuai sistem operasi yang digunakan.
2. Ekstrak berkas zip dan tempatkan folder flutter pada lokasi instalasi yang diinginkan untuk Flutter SDK. Misalnya C:\Development, jangan pasang Flutter di direktori seperti C:\Program Files atau yang membutuhkan hak istimewa seperti administrator.
3. Temukan berkas flutter\_console.bat di dalam direktori flutter tersebut. Mulai dengan klik dua kali atau jalankan script tersebut dan Anda sekarang siap untuk menjalankan perintah Flutter di Flutter Console.
4. Tampilan dari flutter\_console.bat seperti di bawah ini:

### Update Path

Langkah selanjutnya kita akan melakukan update path supaya perintah Flutter bisa digunakan pada command prompt/terminal. Berikut langkah-langkahnya:

* [**Windows**](https://www.dicoding.com/academies/159/tutorials/6446?from=6444#tab3-code1)
* [MacOS](https://www.dicoding.com/academies/159/tutorials/6446?from=6444#tab3-code2)
* [Linux](https://www.dicoding.com/academies/159/tutorials/6446?from=6444#tab3-code3)

1. Dari bar pencarian di Start menu, ketik ‘env’ dan pilih Edit Environment Variable untuk akun Anda.
2. Klik pada tombol Environment Variables.
3. Di bawah User variabel periksa apakah ada entri yang disebut PATH, jika ada maka pilih lalu edit, jika tidak ada maka buat baru dengan nama variabel Path.
4. Edit atau tambahkan value-nya dengan direktori Flutter SDK.
   * Jika terdapat entri, tambahkan path lengkap ke flutter\bin menggunakan tanda titik koma (;) sebagai pemisah dari nilai yang ada (jika menggunakan mode edit satu baris).
   * Jika entri tidak ditemukan, buat user variabel baru dan beri nama Path dan beri nilai flutter\bin sebagai nilainya.

Perhatikan bahwa Anda harus menutup dan membuka kembali semua jendela konsol yang ada agar perubahan dapat terlihat.

### Flutter Doctor

Flutter doctor adalah perintah untuk mengecek kelengkapan framework flutter yang Anda gunakan, seperti versi flutter yang digunakan, Android SDK yang digunakan, iOS SDK yang digunakan (hanya pada MacOS), perangkat yang sudah terhubung, dan semacamnya.  Periksa kembali dan pastikan dependensi untuk pengaturan sudah lengkap. Jalankan perintah berikut untuk membuka flutter doctor:

1. flutter doctor

Perintah ini memeriksa environment Anda dan menampilkan laporan ke jendela terminal. Pada Flutter SDK sudah terdapat Dart SDK, jadi Anda tidak perlu menginstal Dart secara terpisah. Periksa output dengan cermat untuk perangkat lunak lain yang mungkin perlu Anda instal atau melakukan sesuatu lebih lanjut (ditunjukkan dalam teks tebal).

Contoh :

|  |
| --- |
| [-] Android toolchain - develop for Android devices  • Android SDK at D:\Android\sdk  \*\*✗ Android SDK is missing command line tools; download from https://goo.gl/XxQghQ\*\*  • Try re-installing or updating your Android SDK,      visit https://flutter.io/setup/#android-setup for detailed instructions. |

Bagian tersebut menjelaskan cara menyelesaikan proses instalasi Flutter SDK. Setelah memasang dependensi yang hilang, jalankan perintah flutter doctor lagi untuk memverifikasi bahwa Anda telah mengatur semuanya dengan benar.

**Project Wizard**

Setelah berhasil menginstal Flutter SDK dan IDE artinya peralatan yang kita butuhkan telah siap. Bagian ini akan menjelaskan langkah-langkah untuk membuat aplikasi Flutter baru, mulai dari template, cara aplikasi, dan menggunakan "Hot Reload" setelah Anda melakukan perubahan kode aplikasi.

Pilih alat pengembangan (IDE) favorit Anda untuk menulis baris perintah, membangun, dan menjalankan aplikasi Flutter.

* [Android Studio/IntelliJ](https://www.dicoding.com/academies/159/tutorials/8586?from=6451#tab1-code1)
* [**Visual Studio Code**](https://www.dicoding.com/academies/159/tutorials/8586?from=6451#tab1-code2)
* [Terminal & Editor](https://www.dicoding.com/academies/159/tutorials/8586?from=6451#tab1-code3)

1. Buka Visual Studio Code.
2. Aktifkan Command Line dengan cara klik **View>Command Pallete…**
3. Ketik “**Flutter”** lalu pilih **Flutter: New Project**.
4. Masukkan **nama project**, dan tekan **Enter**.
5. Buat atau pilih direktori untuk folder project baru.
6. Tunggu beberapa saat sampai VSCode selesai menyiapkan project.

Perintah di atas membuat direktori project Flutter yang berisi aplikasi demonstrasi sederhana menggunakan Material Components.

**Tips**: Kode Dart untuk aplikasi Anda ada di lib/main.dart. Untuk deskripsi lebih lanjut tentang apa yang dilakukan setiap blok kode, lihat komentar yang ada di dalam file tersebut.

## Menjalankan Emulator Android atau iOS

Sebelumnya kita telah berhasil menginstal flutter SDK, IDE, dan membuat project baru berdasarkan IDE yang digunakan. Sekarang kita akan mulai belajar cara menjalankan project Android menggunakan emulator atau device, maupun project iOS dengan menggunakan simulator.

### Menjalankan di Android

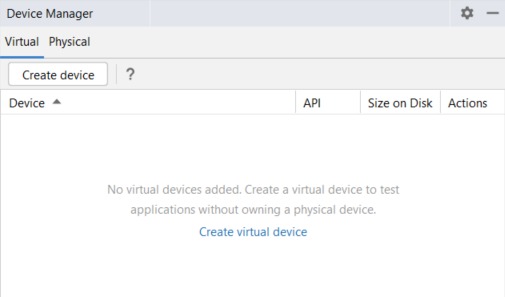
Kita akan menggunakan project Flutter yang telah dibuat sebelumnya untuk dijalankan di emulator atau device Android. Ada beberapa hal yang perlu diperhatikan untuk dapat menjalankan project-nya, uraiannya adalah sebagai berikut:

#### Android Emulator

Sebelum menggunakan emulator, Anda perlu memastikan beberapa syarat.

1. Virtualization  
   Untuk menjalankan emulator di dalam Android Studio, pastikan aspek virtualization. Sistem Anda harus memenuhi persyaratan, yakni ketentuan prosesor dan sistem operasi dari laptop/PC yang Anda gunakan.
2. Processor
   1. Intel: Jika Anda menggunakan processor Intel, maka pastikan bahwa processor tersebut mendukung Intel VT-X, Intel EM64T (Intel 64), dan Execute Disable (XD) Bit functionality.  
      Processor Intel mampu menjalankan emulator di sistem operasi Windows, Linux, mau pun MacOS.
   2. AMD: Jika Anda menggunakan AMD, maka pastikan bahwa ia mendukung AMD Virtualization (AMD-V) dan Supplemental Streaming SIMD Extensions 3 atau yang biasa disebut dengan SSSE3.  
      Processor AMD hanya bisa menjalankan emulator di sistem operasi Linux.

#### Persiapan Running menggunakan Emulator

1. Untuk menjalankan aplikasi, klik menu **Tools**, lalu pilih **Device Manager**.  
   [](https://www.dicoding.com/academies/159/tutorials/8591?from=8586)
2. Untuk membuat emulator baru, klik tombol **Create device** atau **Create virtual device**.
3. Halaman Select Hardware akan muncul.

Jika Anda ingin membuat spesifikasi hardware (perangkat keras) sendiri, pilih **New Hardware Profile**. Anda dapat menentukan konfigurasi hardware sesuai dengan kebutuhan Anda. Ingatlah untuk menggunakan konfigurasi emulator yang sesuai dengan kemampuan laptop atau komputer yang Anda gunakan. Setelah itu klik **Next**.

1. Halaman System Image akan muncul.  
   Pada halaman ini Anda dapat memilih versi Android dari emulator yang akan Anda buat. Pilih versi yang sudah diunduh, misal **Pie**. Lihat tombol download di sebelah kanan. Anda perlu mengunduh terlebih dahulu jika ingin menggunakannya. Setelah itu klik **Next**.
2. Halaman Verify Configuration akan muncul.  
   Pada halaman ini Anda bisa mengubah konfigurasi hardware yang telah ditentukan sebelumnya. Tekan tombol **Show Advanced Setting** pada bagian kiri bawah. Jika sudah selesai, klik **Finish**. Pengaturan emulator sudah selesai dan bisa langsung dijalankan. Anda dapat membuka emulatornya dengan menekan tombol hijau yang ada di sebelah kanan.

#### Android Device

Selain menggunakan Android Emulator, Anda juga dapat melakukan run atau debugging menggunakan perangkat smartphone asli. Menjalankan aplikasi menggunakan perangkat fisik memiliki beberapa kelebihan jika dibandingkan dengan emulator, yaitu:

* Lebih cepat,
* Lebih ringan, dan
* Lebih mudah.

Dengan menggunakan peranti smartphone asli, kita dapat memastikan bahwa aplikasi kita berjalan dengan wajar ketika sudah sampai di tangan pengguna. Kendala dari pendekatan ini adalah beragamnya model peranti yang ada di pasaran. Namun, pembahasan mengenai hal tersebut tidak tercakup dalam kelas ini.

Mari ikuti langkah-langkah untuk menjalankan proses run atau debugging. Tampilan dari langkah berikut bisa dipastikan akan berbeda dengan peranti yang Anda pakai. Akan tetapi secara garis besar langkahnya akan sama.

1. Pastikan peranti yang akan dipakai sesuai dengan target SDK atau paling tidak mendukung versi SDK terendah yang digunakan aplikasi.
2. Buka **setting** dan masuk ke dalam menu **About**. Pada halaman menu ini, Anda perlu menemukan informasi tentang Build number.
3. Kemudian tekan **Build number**sebanyak 7 kali.
4. Kembali ke menu **setting** di awal dan akan muncul menu baru di bawah about yaitu **Developer Options**.
5. Masuk ke dalam menu **Developer Options** dan pastikan opsi **USB Debugging Mode** sudah dalam keadaan **On**.
6. Setelah menyelesaikan pengaturan pada peranti, peranti pun dapat dihubungkan dengan laptop atau komputer yang Anda pakai.

**Catatan:** Beberapa vendor smartphone memiliki sistem operasi yang unik. Tampilan setting dan letak opsi bisa jadi berbeda dengan gambar di atas. Beberapa vendor juga mengharuskan Anda untuk mengunduh driver khusus sebelum bisa menghubungkannya ke Android Studio. Kami sarankan untuk mengunjungi website atau membaca petunjuk yang sesuai dengan vendor dari peranti Anda.

### Menjalankan di iOS

Kita akan menggunakan project Flutter yang telah dibuat sebelumnya untuk dijalankan di iOS simulator. Ada beberapa hal yang perlu diperhatikan untuk dapat menjalankan project-nya, uraiannya adalah sebagai berikut:

#### iOS Simulator

iOS Simulator atau Simulator adalah aplikasi yang digunakan untuk menjalankan aplikasi iOS. Simulator merupakan aplikasi bawaan dari XCode.

Selain dari Simulator, sebenarnya kita bisa menggunakan device iphone asli. Akan tetapi untuk menjalankan di device asli kita memerlukan akun Apple developer. Jadi untuk menyederhanakan masalah, di modul ini hanya akan diajarkan dengan media Simulator saja.

Ikuti langkah berikut ini untuk membuka Simulator:

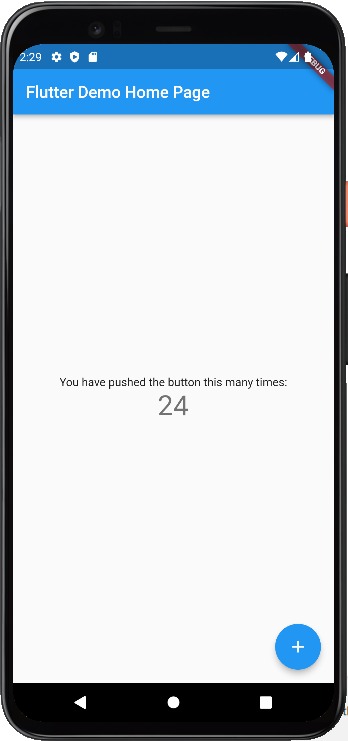
1. Buka terminal dan jalankan kode berikut ini:
   1. open -a Simulator
2. Aplikasi simulator akan muncul, contohnya seperti gambar di bawah ini:
3. Pastikan Simulator yang Anda jalankan adalah 64-bit (Iphone 5 ke atas), jika belum maka ganti konfigurasinya dengan mengakses lewat Device atau **File -> Open Devices (versi terbaru)**.

**Catatan:** Contoh di atas diambil dengan MacOS Catalina. Jika Anda menggunakan versi di bawahnya ada kemungkinan perbedaan tampilan. Jika Anda menemui masalah silakan bertanya di forum diskusi.

**Menjalankan Project Flutter**

Setelah mengetahui persiapan sebelum menjalankan *project* di Android dan di iOS, baik itu di *emulator* maupun *device*. Berikut cara untuk menjalankannya di beberapa IDE yang sudah dijelaskan sebelumnya.

* [Android Studio/IntelliJ](https://www.dicoding.com/academies/159/tutorials/8596?from=8591#tab1-code1)
* [**Visual Studio Code**](https://www.dicoding.com/academies/159/tutorials/8596?from=8591#tab1-code2)
* [Terminal & Editor](https://www.dicoding.com/academies/159/tutorials/8596?from=8591#tab1-code3)

1. Temukan **Status Bar** pada bagian paling bawah jendela VSCode seperti di bawah ini:
2. Pilih perangkat di area **Device Selector.**  
   1. Jika tidak ada perangkat yang tersedia dan Anda ingin menggunakan simulator perangkat, klik **No Device** dan luncurkan simulator.
   2. Untuk menyiapkan perangkat fisik (smartphone pribadi), ikuti petunjuk khusus perangkat pada saat melakukan **Install Flutter SDK** untuk OS Anda.
3. Aktifkan **Command Line** dengan cara pilih **Run > Start Debugging** atau **tekan F5**, atau pada kode void main akan muncul **run dan debug**, atau pada tab Run di bagian kiri.
4. Tunggu hingga aplikasi diluncurkan dan lihat progress dalam tampilan **Debug Console**.
5. Setelah build aplikasi selesai, Anda akan melihat aplikasi starter di perangkat Anda.  
   [](https://www.dicoding.com/academies/159/tutorials/8596?from=8591)

Selamat Anda telah berhasil menjalankan aplikasi Flutter pertama Anda!

**Menjalankan di Web**

Selain platform mobile, Flutter juga bisa berjalan pada web browser. Dalam proses pengembangan, untuk keperluan debugging kita perlu menggunakan [Google Chrome](https://www.google.com/chrome/) sebagai browser. Sejak Flutter versi 2.0 Flutter Web sudah memasuki versi stable sehingga bisa digunakan secara langsung. Jika menggunakan versi di bawahnya, Anda perlu mengaktifkan channel beta terlebih dahulu. Silakan ikuti langkahnya pada blog [berikut](https://www.dicoding.com/blog/ingin-bikin-web-menggunakan-flutter-yuk-kenalan-dengan-hummingbird/).

Untuk menambahkan dukungan web pada project Anda, jalankan perintah berikut melalui terminal dari lokasi project:

1. flutter config --enable-web
2. flutter create .

Setelah konfigurasi dan proses re-create berhasil, folder web akan ditambahkan ke dalam folder project Anda. Di dalam folder ini akan berisi berkas html dan konfigurasi web lainnya. Anda bisa mencoba menjalankan aplikasi web Flutter ini dengan memilih target device chrome pada IDE yang Anda gunakan.

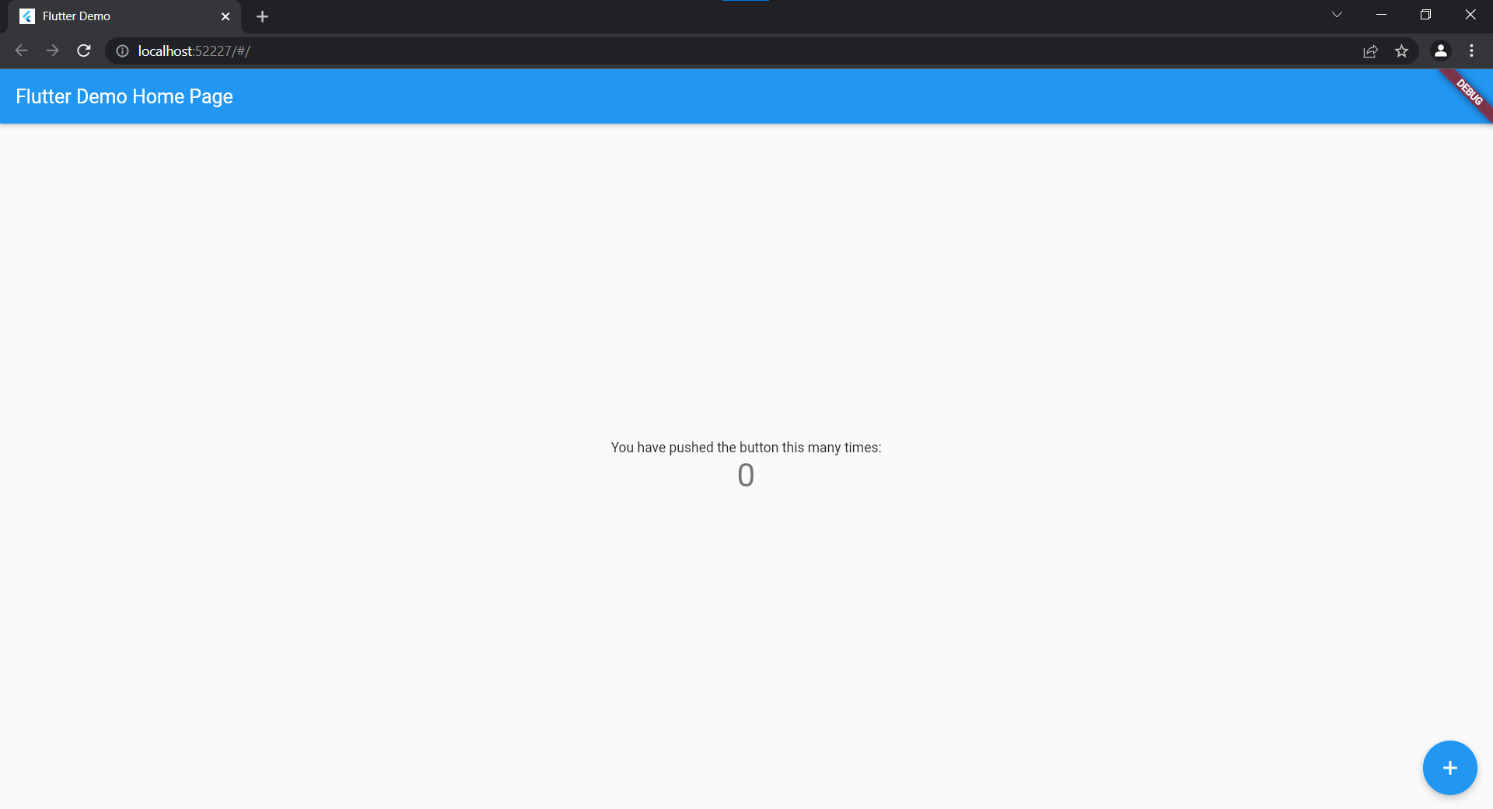
Android Studio:

Visual Studio Code:

Ketika target device Chrome sudah terpilih seperti di atas, Anda dapat menjalankan aplikasi dengan cara yang sama seperti ketika menjalankannya pada platform mobile. Jika Anda ingin menjalankan melalui terminal, maka perintahnya adalah seperti ini:

1. flutter run -d chrome

Setelah proses build selesai, jendela browser akan muncul dan menampilkan aplikasi Anda:

[](https://www.dicoding.com/academies/159/tutorials/16758?from=8596)

Pada Flutter Web, perlu diperhatikan bahwa Flutter memiliki dua jenis renderer yang berbeda, keduanya antara lain:

* **HTML renderer**  
  Renderer ini menggunakan kombinasi elemen HTML, CSS, Canvas, dan SVG. Jenis renderer ini memiliki ukuran unduhan yang lebih kecil.
* **CanvasKit renderer**  
  Renderer ini bekerja dengan cara yang sama dengan platform mobile atau desktop. CanvasKit renderer memiliki performa yang lebih tinggi, tetapi akan menambahkan ukuran hingga sekitar 2 MB.

Anda dapat menentukan renderer yang digunakan dengan menambahkan parameter pada *command line*, contohnya seperti berikut:

1. flutter run -d chrome --web-renderer html
2. flutter run -d chrome --web-renderer canvaskit

Jika Anda tidak mendefinisikan parameter --web-renderer, maka renderer akan menggunakan mode auto (*default*). Opsi ini akan menggunakan HTML renderer ketika aplikasi berjalan di browser mobile dan menggunakan CanvasKit saat aplikasi berjalan di browser desktop.

## Hot reload

Fitur **hot reload** pada Flutter membantu Anda untuk melakukan percobaan, membangun UI, menambahkan fitur, serta memperbaiki bug dengan cepat dan mudah. Hot reload berfungsi dengan menyuntikkan berkas kode yang diperbarui ke dalam **Dart Virtual Machine** (VM) yang sedang berjalan. Setelah VM memperbarui class dengan versi field dan fungsi yang baru, framework Flutter secara otomatis membangun kembali widget. Hal ini memungkinkan Anda dengan cepat melihat efek dari perubahan Anda.

Lebih lanjut, berikut ini cara melakukan hot reload pada aplikasi Flutter:

1. Jalankan aplikasi Flutter. Hanya aplikasi Flutter dalam **mode debug** yang dapat melakukan **hot reload**.
2. Ubah beberapa kode kecil dalam salah satu file Dart di project Anda. Jika perubahan yang terjadi banyak atau perubahan berada di dalam initState atau main, Anda harus melakukan hot restart.
3. Jika Anda menggunakan IDE/editor yang didukung Flutter (Android Studio/IntelliJ/Visual Studio Code), pilih Save/Save All (cmd+s/ctrl+s) atau klik tombol **Hot Reload** pada toolbar di setiap IDE.
4. Jika Anda menjalankan aplikasi di command line menggunakan **flutter run**, masukkan perintah **r** di jendela terminal.

Setelah operasi hot reload berhasil, Anda akan melihat pesan di konsol seperti ini:

|  |
| --- |
| Performing hot reload...  Reloaded 1 of 448 libraries in 978ms. |

Hot reload menyebabkan semua widget yang ada dibangun kembali. Hanya kode yang terlibat dalam rebuilding widget yang dieksekusi ulang secara otomatis. Seperti contoh fungsi main dan initState tidak akan dijalankan kembali jika menggunakan **hot reload**.

Apa bedanya hot reload, hot restart, dan full restart?

* **Hot reload** memuat perubahan kode ke dalam VM dan menjalankan kembali metode build yang ada di dalam widget. Ini akan membangun kembali widget-widget yang ada, dan mempertahankan status terakhir aplikasi.
* **Hot restart** memuat perubahan kode ke dalam VM dan mengulang kembali aplikasi dari awal dan akan kehilangan state aplikasi (kembali ke nilai awal).
* **Full restart** mengkompilasi kode dari awal, tidak ada pintasan keyboard untuk ini, Anda harus menghentikan dan menjalankan kembali project (klik stop kemudian run kembali).

Flutter web saat ini hanya mendukung **hot restart**.

* [Android Studio/IntelliJ](https://www.dicoding.com/academies/159/tutorials/8601?from=16758#tab1-code1)
* [**Visual Studio Code**](https://www.dicoding.com/academies/159/tutorials/8601?from=16758#tab1-code2)
* [Terminal & Editor](https://www.dicoding.com/academies/159/tutorials/8601?from=16758#tab1-code3)

Flutter menawarkan siklus pengembangan yang cepat dengan Hot Reload. Fitur tersebut memiliki kemampuan memuat ulang kode aplikasi yang sedang berjalan tanpa memulai ulang atau kehilangan status aplikasi. Caranya, Buat perubahan ke sumber aplikasi, beritahu IDE atau alat baris perintah Anda bahwa Anda ingin melakukan Hot Reload, dan lihat perubahan di simulator, emulator, atau perangkat Anda.

1. Buka **lib/main.dart**.
2. Ubah string 'You have ***pushed*** the button this many times' menjadi 'You have ***clicked*** the button this many times'.  
   **Penting:** Jangan hentikan aplikasi Anda. Biarkan aplikasi Anda berjalan.
3. Simpan perubahan baris perintah dengan cara: **Pilih File > Save All**, atau klik **Hot Reload (tombol petir kuning)**.

Anda akan langsung melihat string yang sudah diperbarui di aplikasi yang sedang berjalan.

### Kasus spesial

Bagian berikutnya menjelaskan situasi umum di mana kode yang dimodifikasi tidak akan berjalan lagi setelah hot reload. Dalam beberapa kasus, perubahan kecil pada kode Dart akan memungkinkan Anda untuk terus menggunakan hot reload untuk aplikasi Anda. Dalam beberapa kasus, kita perlu melakukan hot restart atau full restart:

* **Aplikasi mati atau keluar**  
  Hot reload akan berhenti jika aplikasi terhenti atau keluar, misalnya jika aplikasi terlalu lama berada di background.
* **Compilation errors**  
  Ketika perubahan kode melihatkan kesalahan kompilasi, hot reload selalu menghasilkan pesan kesalahan yang seperti di bawah ini:  
  Hot reload was rejected:'/Users/obiwan/Library/Developer/CoreSimulator/Devices/AC94F0FF-16F7-46C8-B4BF-218B73C547AC/data/Containers/Data/Application/4F72B076-42AD-44A4-A7CF-57D9F93E895E/tmp/ios\_testWIDYdS/ios\_test/lib/main.dart': warning: line 16 pos 38: unbalanced '{' opens hereWidget build(BuildContext context) {^'/Users/obiwan/Library/Developer/CoreSimulator/Devices/AC94F0FF-16F7-46C8-B4BF-218B73C547AC/data/Containers/Data/Application/4F72B076-42AD-44A4-A7CF-57D9F93E895E/tmp/ios\_testWIDYdS/ios\_test/lib/main.dart': error: line 33 pos 5: unbalanced ')');^  
  Dalam situasi ini cukup perbaiki kesalahan pada baris kode Dart yang ditentukan untuk tetap menggunakan hot reload.
* **CupertinoTabView’s builder**  
  Hot reload tidak akan berjalan ketika ada perubahan pada builder di CupertinoTabView. Untuk lebih lanjutnya silakan baca <https://github.com/flutter/flutter/issues/43574>.
* **Tipe enum**  
  Hot reload tidak akan bekerja ketika tipe enum diubah menjadi kelas reguler atau sebaliknya.
  1. // Contoh sebelum:
  2. enum Level { beginner, intermediate, advanced }

  5. // Contoh sesudah:
  6. class Level {
  7. Level(this.id);
  8. int id;
  9. }
* **Perubahan font**  
  Hot reload mendukung sebagian besar perubahan aset. Namun, jika mengubah font, Anda harus menggunakan hot restart.
* **Generic types**
  1. Hot reload tidak akan bekerja ketika pendeklarasian tipe generik dimodifikasi.
  2. // Contoh sebelum:
  3. class A<X> {
  4. X ax;
  5. }

  8. // Contoh sesudah:
  9. class A<X, Y> {
  10. X ax;
  11. Y ay;
  12. }
* **Native code**  
  Jika Anda mengubah kode native tiap platform (seperti Java, Kotlin, Swift, atau Objective-C), Anda harus melakukan full restart (stop dan run kembali project) untuk melihat perubahan.
* **State sebelumnya dikombinasikan dengan kode baru**  
  Fitur hot reload pada Flutter kadang-kadang digambarkan sebagai stateful hot reload yang mempertahankan status aplikasi Anda. Ini memungkinkan Anda untuk melihat efek perubahan terbaru saja tanpa membuang kondisi saat ini. Misalnya, jika aplikasi Anda mengharuskan pengguna untuk login, Anda dapat memodifikasi dan melakukan hot reload beberapa halaman di bawah hierarki navigasi, tanpa memasukkan kembali kredensial login Anda. Saat disimpan, maka hasil output akan sesuai dengan yang kita inginkan atau kita ubah.  
    
  Jika perubahan kode memengaruhi kondisi aplikasi Anda, seperti data atau fungsi yang seharusnya berjalan dengan baik tiba-tiba berhenti berfungsi. Maka disarankan untuk menggunakan Hot Restart.  
    
  Seperti Kode di bawah ini :
  1. class MyWidget extends StatelessWidget {
  2. Widget build(BuildContext context) {
  3. return GestureDetector(onTap: () => print('T'));
  4. }
  5. }

Diubah menjadi:

* 1. class MyWidget extends StatefulWidget {
  2. @override
  3. State createState() => MyWidgetState();
  4. }
  6. class MyWidgetState extends State { /\*...\*/ }

Kemudian tekan tombol hot reload, maka console akan menampilkan output seperti berikut:  
MyWidget is not a subtype of StatelessWidget  
Dalam situasi ini, full restart diperlukan untuk melihat aplikasi yang diperbarui.

Selamat Anda telah menyelesaikan modul pertama ini. Semoga tetap semangat ya. Apalagi kini Anda telah membuat project dan menjalankan aplikasi Flutter pertama Anda. Wow! Pada modul selanjutnya Anda telah siap untuk membuat aplikasi Anda sendiri. Sudah siap? Jaga semangat Anda dan sampai jumpa di modul berikutnya ya!

**Rangkuman Materi**

Selamat Anda telah menyelesaikan modul pertama ini. Semoga tetap semangat ya. Apalagi kini Anda telah membuat project dan menjalankan aplikasi Flutter pertama Anda. Wow! Mari kita ingat kembali apa saja yang telah dibahas pada modul ini:

* Flutter adalah SDK (Software Development Kit) yang dikembangkan oleh Google untuk membuat aplikasi yang bagus dan bisa berjalan pada berbagai platform.
* Flutter 2 yang merupakan versi terbaru memberikan dukungan pada Anda untuk membangun aplikasi pada sistem operasi Android, iOS, Web, Windows, Linux, dan MacOS.
* Flutter menawarkan empat karakteristik atau kelebihan, antara lain: beautiful, fast, productive, dan open.
* Flutter ditulis menggunakan bahasa Dart. Dart merupakan bahasa pemrograman berbasis kelas dan berorientasi terhadap objek dengan menggunakan Syntax bahasa pemrograman C.
* Untuk bisa mengembangkan aplikasi menggunakan Flutter, kita perlu menyiapkan dan menginstal Flutter SDK. Tools ini tersedia pada sistem operasi Windows, macOS, dan Linux.
* Flutter memiliki dukungan terhadap beberapa IDE populer, seperti Android Studio, IntelliJ IDEA, dan Visual Studio Code.
* Kita dapat menguji coba dan menjalankan aplikasi Flutter menggunakan perangkat fisik, Android Emulator, iOS Simulator, atau web browser.
* Flutter Web memiliki dua jenis renderer yang berbeda, yaitu HTML renderer dan CanvasKit renderer. Jika tidak mendefinisikan renderer apa yang digunakan, maka mode auto yang akan dipakai.
* Flutter memiliki fitur yang sangat berguna untuk meningkatkan produktivitas pengembangan, yaitu hot reload. Fitur ini akan secara langsung mengaplikasikan perubahan kode tanpa harus melakukan proses build ulang.

Pada modul selanjutnya Anda telah siap untuk membuat aplikasi Anda sendiri. Sudah siap? Jaga semangat Anda dan sampai jumpa di modul berikutnya ya!