#02 | Pengantar Bahasa Pemrograman Dart - Bagian 1

| **Summary** | Mahasiswa mampu menjelaskan evolusi dart, cara kerjanya, struktur bahasanya, operator dart, dan main function |
| --- | --- |
| **URL** | 02-pengantar-bahasa-pemrograman-dart-bag-1 |
| **Category** | Mobile Flutter |
| **Environment** | web |
| **Status** | Published |
| **Feedback Link** | https://github.com/jti-polinema/flutter-codelab/issues |
| **Author** | Habibie Ed Dien |
| **Author LDAP** | hbb-polinema |
| **Analytics Account** | G-H7507PFEJB |

[Pengantar Bahasa Pemrograman Dart - Bagian 1](#_ok7k5uux6)

[Apa yang Anda akan pelajari](#_21yzqg98x7h6)

[Apa yang Anda perlu persiapkan](#_hs6hf8vnv2d6)

[Pengetahuan yang Anda harus dimiliki](#_319h1e1sjzdz)

[Getting started with Dart](#_bwo0af2iwk90)

[The evolution of Dart](#_jdmzwkcs72o)

[How Dart works](#_ot43tirxvs6w)

[Dart VM and JavaScript compilation](#_v7doc7dqxt9q)

[Introducing the structure of the Dart language](#_3tnx3743juyf)

[Object orientation](#_v1othgyd5k2b)

[Dart operators](#_kkswhgfku9sn)

[Arithmetic operators](#_feyb33cnek4g)

[Increment and decrement operators](#_tugpij91egy6)

[Equality and relational operators](#_hqv2482uv2h5)

[Logical operators](#_cx845autmvr4)

[Hands-on with Dart](#_li2h8jf4xyp6)

[DartPad](#_dnogxbo334la)

[Hello world Dart style](#_sz55qqr012g9)

[Main function](#_8prajj6fgk36)

[Tugas Praktikum](#_dz35nqtk0wjb)

[Soal 1](#_dyi35l9d12kz)

[Soal 2](#_vei0y249v2vb)

[Soal 3](#_ijl9dnn6pxt)

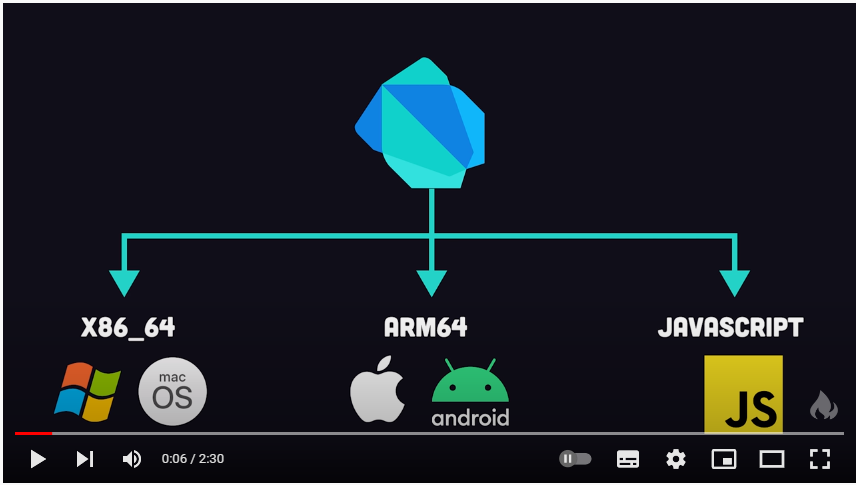
[Soal 4](#_6s11epie77iq)

[Selamat!](#_s22qmcj4zu61)

[Apa selanjutnya?](#_9mqifffit2ew)

[Referensi](#_an95hoqxrlhg)

# Pengantar Bahasa Pemrograman Dart - Bagian 1



**Terakhir diperbarui:** 04 September 2023

**Penulis:** Habibie Ed Dien

Bahasa Dart adalah inti dari framework Flutter. Kerangka kerja modern seperti Flutter membutuhkan bahasa modern tingkat tinggi agar bisa memberikan pengalaman terbaik kepada pengembang, serta memungkinkan untuk membuat aplikasi seluler yang luar biasa. Memahami Dart adalah dasar untuk bekerja dengan Flutter; pengembang perlu mengetahui asal-usul bahasa Dart, bagaimana komunitas mengerjakannya, kelebihannya, dan mengapa itu adalah bahasa pemrograman yang dipilih untuk Flutter.

Dalam codelab ini, Anda akan meninjau dasar-dasar bahasa Dart dan mengidentifikasi sumber daya yang dapat membantu Anda dalam perjalanan Flutter Anda. Anda akan mempelajari tipe data dan operator bawaan Dart serta bagaimana Dart bekerja dengan pemrograman berorientasi objek (OOP). Dengan memahami bahasa Dart, Anda akan merasa nyaman untuk bereksperimen dengan Dart dan dapat memperluas pengetahuan Anda.

## Apa yang Anda akan pelajari

Di codelab ini Anda akan mempelajari tentang:

* Geeting started with Dart
* The evolution of Dart
* How Dart works
* Introducing the structure of the Dart language
* Hands-on with Dart

| **Catatan:** Materi ini diadaptasi dari Buku Flutter for Beginners - second edition dan Buku Ajar Pemrograman Mobile Flutter JTI Polinema serta sumber referensi lainnya. |
| --- |

## 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

# Getting started with Dart

Duration: 15:00

Dart bertujuan untuk menggabungkan kelebihan-kelebihan dari sebagian besar bahasa tingkat tinggi dengan fitur-fitur bahasa pemrograman terkini, antara lain sebagai berikut:

* **Productive tooling**: merupakan fitur kakas (*tool*) untuk menganalisis kode, plugin IDE, dan ekosistem paket yang besar.
* **Garbage collection**: untuk mengelola atau menangani dealokasi memori (terutama memori yang ditempati oleh objek yang tidak lagi digunakan).
* **Type annotations (opsional)**: untuk keamanan dan konsistensi dalam mengontrol semua data dalam aplikasi.
* **Statically typed**: Meskipun *type annotations* bersifat opsional, Dart tetap aman karena menggunakan fitur *type-safe* dan *type inference* untuk menganalisis *types* saat *runtime*. Fitur ini penting untuk menemukan *bug* selama kompilasi kode.
* **Portability**: bahasa Dart tidak hanya untuk web (yang dapat diterjemahkan ke JavaScript) tetapi juga dapat dikompilasi secara *native* ke kode **Advanced RISC Machines (ARM)** dan x86.

Semua pengembangan framework Flutter melibatkan pengetahuan/fitur mendalam dengan bahasa Dart; Kode aplikasi, kode plugin, dan manajemen dependensi semuanya menggunakan bahasa Dart beserta fitur-fiturnya. Memiliki pemahaman dasar yang kuat tentang Dart akan memudahkan Anda untuk menjadi lebih produktif dengan Flutter dan akan membuat Anda merasa nyaman dalam pengembangan Flutter. Ayo kita pelajari bahasa Dart lebih rinci, dimulai dari mana Dart berasal.

# The evolution of Dart

Duration: 10:00

Diluncurkan pada tahun 2011, Dart telah berkembang sejak saat itu. Dart merilis versi stabilnya pada tahun 2013, dengan perubahan besar termasuk dalam rilis Dart 2.0 menjelang akhir 2018, yang dapat diuraikan sebagai berikut:

* Awalnya berfokus pada pengembangan web, dengan tujuan utama menggantikan JavaScript, sekarang telah fokus pada mobile development, termasuk framework Flutter.
* **Mencoba memecahkan masalah pada JavaScript**: JavaScript tidak menyediakan ketahanan seperti banyak bahasa pemrograman lainnya, sehingga Dart ingin menjadi penerus daripada JavaScript.
* **Menawarkan performa terbaik dan alat yang lebih baik untuk proyek berskala besar:** Dart memiliki perkakas yang modern dan stabil yang telah disediakan oleh plugin IDE. Hal ini telah dirancang untuk mendapatkan performa terbaik dengan tetap menjaga nuansa bahasa yang dinamis.
* **Dibentuk agar kuat dan fleksibel:** Dengan tetap mempertahankan type annotations bersifat opsional dan menambahkan fitur OOP, Dart dapat menyeimbangkan dua fitur utama yaitu fleksibilitas dan ketangguhan.

Dart adalah bahasa modern yang luar biasa, mendukung lintas platform, dan memiliki tujuan umum dengan terus meningkatkan fitur-fiturnya, membuatnya lebih kekinian dan fleksibel. Itulah sebabnya tim Flutter framework memilih bahasa Dart untuk digunakan.

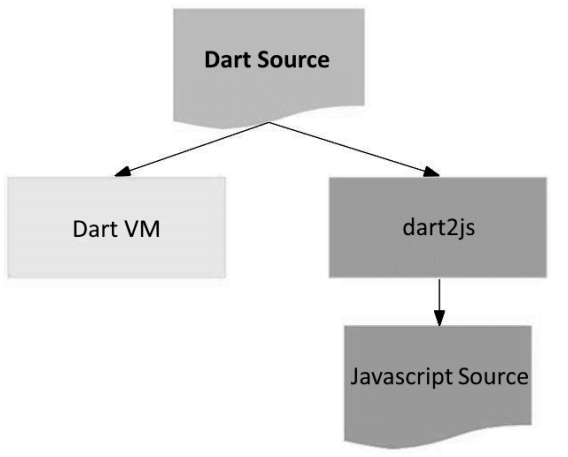
# How Dart works

Duration: 15:00

Untuk memahami dari mana fleksibilitas bahasa dart itu berasal, kita perlu tahu bagaimana cara mengeksekusi kode Dart. Ini dapat dilakukan dengan dua cara, seperti yang dijelaskan berikut ini:

* Dart **virtual machines (VMs**)
* JavaScript compilations

Perhatikan gambar diagram berikut:



Seperti yang Anda lihat, pada bagian atas diagram adalah kode Dart Anda. Perlu dicatat bahwa kode dan dependensi Anda bisa jadi tidak sesuai dengan cara Anda mengeksekusikan aplikasi; namun tidak perlu melakukan banyak perubahan pada kode Anda untuk dapat mendukung cross-platform.

## Dart VM and JavaScript compilation

Kode Dart dapat dieksekusi pada lingkungan yang mendukung bahasa Dart. Lingkungan yang mendukung bahasa Dart perlu memperhatikan fitur-fitur penting seperti berikut:

* *Runtime systems*
* *Dart core libraries*
* *Garbage collectors*

Eksekusi kode Dart dapat beroperasi dalam dua mode — kompilasi **Just-In-Time (JIT)** atau Kompilasi **Ahead-Of-Time (AOT).** Dijelaskan secara lebih rinci sebagai berikut:

* Kompilasi JIT adalah tempat kode sumber dikompilasi sesuai kebutuhan—*Just in time*. Dart VM memuat dan mengkompilasi kode sumber ke kode mesin asli (*native*). Pendekatan ini digunakan untuk menjalankan kode pada *command line* atau selama proses pengembangan aplikasi mobile yang dapat memanfaatkan fitur seperti debugging dan *hot reload*.
* Kompilasi AOT adalah dimana Dart VM dan kode Anda dikompilasi sebelumnya, VM bekerja lebih seperti sistem runtime Dart, yang menyediakan *garbage collector* dan metode-metode *native* dari Dart **software development kit (SDK)** pada aplikasi. Pendekatan ini memiliki keuntungan performa yang sangat besar dibandingkan kompilasi JIT, tetapi fitur lain seperti debugging dan hot reload tidak tersedia.

| **Hot reload**  Dart banyak berkontribusi pada fitur Flutter yang paling terkenal, yaitu hot reload, yang didasarkan pada kompiler Dart JIT. Hal Ini memungkinkan pengembang untuk mendapatkan umpan balik yang sangat cepat terhadap perubahan kode yang dilakukan, sehingga memungkinkan untuk beralih lebih cepat. Dapat mendukung pengembangan perangkat lunak yang lebih cepat dan berkualitas tinggi. |
| --- |

Sebelum kita menggunakan bahasa pemrograman Dart, sebaiknya kita pahami terlebih dahulu hal-hal mendasar terkait bahasa Dart.

# Introducing the structure of the Dart language

Duration: 45:00

Jika Anda sudah mengetahui beberapa bahasa pemrograman seperti bahasa C atau memiliki beberapa pengalaman dengan JavaScript, sebagian besar sintaks Dart akan lebih mudah bagi Anda untuk memahaminya. Dart menyediakan sebagian besar operator standar untuk memanipulasi variabel; *built-in types* adalah yang paling umum ditemukan dalam bahasa pemrograman tingkat tinggi. *Control flow* dan *function* sangat mirip dengan bahasa pemrograman lainnya.

## Object orientation

Seperti kebanyakan bahasa modern, Dart dirancang untuk **object-oriented (OO)**. Secara singkat, Bahasa OOP didasarkan pada konsep **objek** yang menyimpan kedua data (disebut **fields**) dan kode (disebut **methods**). Objek-objek ini dibuat dari cetak biru yang disebut **class** yang mendefinisikan *field* dan *method* yang akan dimiliki oleh sebuah objek.

Sesuai prinsip OO memastikan bahwa Dart memiliki fitur encapsulation, inheritance, composition, abstraction, dan polymorphism. Kita akan mempelajari kelas Dart lebih banyak lagi dipertemuan dengan topik *Class Dart* dan *Construct,* namun sudah cukup jika Anda telah belajar konsep OO dalam bahasa lain seperti Java, maka sebagian besar desain OO pada Dart akan sangat mirip.

## Dart operators

Di Dart, operator tidak lebih dari *method* yang didefinisikan dalam *class* dengan sintaks khusus.

Jadi, ketika Anda menggunakan operator seperti x == y, seolah-olah Anda sedang memanggil

x.==(y) metode untuk melakukan perbandingan kesetaraan.

Seperti yang mungkin telah Anda catat, kita menggunakan *method* pada x. Untuk semua tipe data, tidak seperti bahasa Java yang memiliki data primitif, x selalu berupa turunan dari kelas yang memiliki *method*. Ini berarti bahwa operator dapat diganti sesuai logika yang Anda inginkan.

## Arithmetic operators

Dart hadir dengan banyak operator *typical* yang bekerja seperti banyak bahasa pemrograman lainnya; yaitu sebagai berikut:

* + untuk tambahan.
* - untuk pengurangan.
* \* untuk perkalian.
* / untuk pembagian.
* ~/ untuk pembagian bilangan bulat. Di Dart, setiap pembagian sederhana dengan / menghasilkan nilai *double*. Untuk mendapatkan nilai bilangan bulat, Anda perlu membuat semacam transformasi (yaitu, *typecast*) dalam bahasa pemrograman lain; namun Dart sudah mendukung untuk operasi ini.
* % untuk operasi modulus (sisa bagi dari bilangan bulat).
* -expression untuk negasi (yang membalikkan suatu nilai).

Beberapa operator memiliki perilaku yang berbeda tergantung pada jenis operan di sisi kiri; Misalnya, operator + dapat digunakan untuk menjumlahkan variabel dari tipe num, tetapi juga dapat digunakan untuk menggabungkan string. Karena *method* yang dirujuk diimplementasikan secara berbeda pada kelas yang berbeda.

Dart juga menyediakan *shortcut* operator untuk menggabungkan variabel setelah operasi lainnya. Operator aritmatika atau *shortcut* operator adalah +=, -=, \*=, /=, dan ~/=.

## Increment and decrement operators

Operator penambahan dan pengurangan juga merupakan operator umum dan diimplementasikan pada angka, sebagai berikut:

* ++var atau var++ untuk menambah nilai variabel var sebesar 1
* --var atau var-- untuk mengurangi nilai variabel var sebesar 1

Operator Dart increment dan decrement berperilaku mirip dengan bahasa lain. Penerapan operator increment dan decrement sangat baik untuk operasi perhitungan pada perulangan.

## Equality and relational operators

Persamaan operator Dart dijelaskan sebagai berikut:

* == untuk memeriksa apakah operan sama
* != untuk memeriksa apakah operan berbeda

Untuk melakukan pengujian relasional, maka gunakan operator sebagai berikut:

* > memeriksa apakah operan kiri lebih besar dari operan kanan
* < memeriksa apakah operan kiri lebih kecil dari operan kanan
* >= memeriksa apakah operan kiri lebih besar dari atau sama dengan operan kanan
* <= memeriksa apakah operan kiri kurang dari atau sama dengan operan kanan

Di Dart, tidak seperti Java dan bahasa lainnya, operator == tidak membandingkan referensi/alamat memori melainkan isi dari variabel tersebut.

Juga, tidak seperti JavaScript, tidak ada operator === yang diperlukan pada Dart karena telah memiliki fitur *type safety* yang memastikan bahwa operator persamaan == hanya digunakan pada objek dengan tipe yang sama.

## Logical operators

Operator logika di Dart adalah operator yang diterapkan pada operan bool; bisa berupa variabel, ekspresi, atau kondisi. Selain itu, dapat dikombinasikan dengan ekspresi kompleks dengan menggabungkan nilai ekspresi yang dievaluasi. Operator logika yang disediakan adalah sebagai berikut:

* !expression negasi atau kebalikan hasil ekspresi—yaitu, true menjadi false dan false menjadi true.
* || menerapkan operasi logika OR antara dua ekspresi.
* && menerapkan operasi logika AND antara dua ekspresi.

Sekarang kita telah mengetahui dasar-dasar bahasa pemrograman Dart, mari kita lihat beberapa contoh kode Dart!

# Hands-on with Dart

Duration: 30:00

Desain Flutter sangat dipengaruhi oleh bahasa Dart, jadi kita mempelajari bahasa Dart itu sangat penting untuk keberhasilan dalam menggunakan framework Flutter. Mari kita mulai dengan menulis beberapa kode untuk memahami dasar-dasar sintaks dan kakas yang tersedia untuk pengembangan Dart.

## DartPad

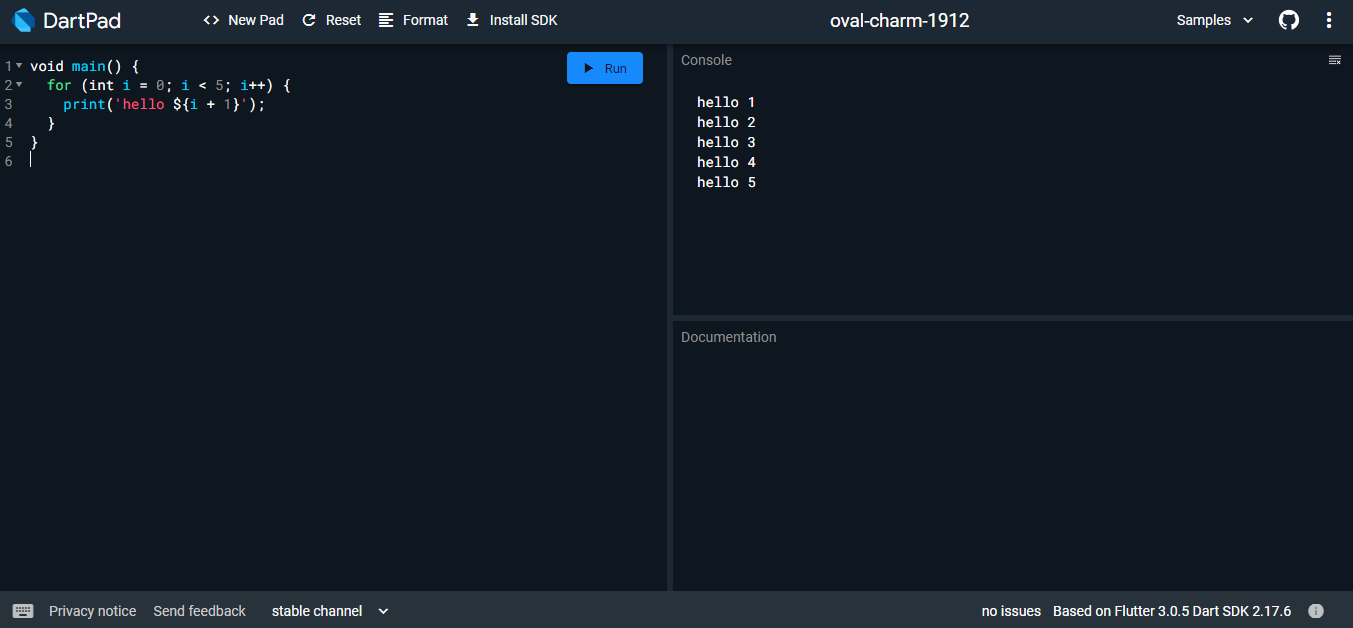
Cara termudah untuk memulai pemrograman Dart adalah dengan menggunakan kakas DartPad, yang dapat Anda akses di tautan berikut:

| Untuk memahami cara penggunaan DartPad lebih lanjut dapat mengakses tautan berikut  <https://dart.dev/resources/dartpad-best-practices#what-is-dartpad> |
| --- |

Ini adalah kakas online yang bagus untuk belajar dan bereksperimen dengan fitur bahasa Dart.

Kakas ini mendukung *core library* Dart, kecuali untuk *library VM* seperti dart:io.

Seperti inilah tampilan kakas DartPad, meskipun kode yang disajikan dalam kakas mungkin berbeda:



Saat Anda membuka DartPad, Anda akan melihat potongan kode. Jika belum ada kode pada DartPad, Anda dapat mengganti dengan kode berikut, sehingga Anda dapat menjalankan kode Dart pertama Anda:

| void main() {  for (int i = 0; i < 5; i++) {  print('hello ${i + 1}');  }  } |
| --- |

Kita akan menjelajahi potongan kode tersebut pada pembahasan subtopik berikutnya, jadi jangan khawatir jika itu terlihat sedikit rumit. Namun, coba jalankan pada kakas DartPad dengan menekan tombol **Run**, dan Anda akan melihat keluaran pada konsol yang mirip seperti berikut:

| hello 1  hello 2  hello 3  hello 4  hello 5 |
| --- |

| **Running locally**  Jika Anda memilih untuk menjalankan kode tersebut secara lokal di PC/Laptop Anda, simpan kode ke file Dart dan eksekusi dengan Dart di terminal—misalnya, simpan ke file bernama hello\_world.dart dan kemudian run perintah dart hello\_world.dart. Ini akan mengeksekusi fungsi main dari kode Dart tersebut. |
| --- |

Mari kita lihat kode di DartPad lebih detail.

## Hello world Dart style

Kode yang ada di DartPad terlihat mirip seperti berikut:

| void main() {  for (int i = 0; i < 5; i++) {  print('hello ${i + 1}');  }  } |
| --- |

Kode ini berisi beberapa fitur dasar dalam bahasa Dart. Dalam pembahasan ini, kita akan menjelajahi dasar-dasar bahasa Dart yang mungkin Anda telah memiliki pengalaman dalam pemrograman dasar untuk menerapkan pengetahuan itu ke bahasa Dart.

Jika Anda merasa membutuhkan pengenalan yang lebih dalam tentang dasar-dasar pemrograman, maka buku yang cocok adalah *Learning Dart*, oleh *Dzenan Ridjanovic*.

## Main function

Seperti kebanyakan bahasa modern, Dart menggunakan *function* dan *method* sebagai cara untuk memisahkan kode. Sebuah *function* atau *method* adalah potongan kode yang menerima beberapa data, eksekusi kode, dan kemudian mengembalikan beberapa data (*return*).

*Function* dari contoh hello world terlihat seperti ini:

| void main() {  …  } |
| --- |

Baris pertama memiliki beberapa informasi penting, yang dijelaskan sebagai berikut:

* Tipe data yang dikembalikan dari *method* perlu didefinisikan terlebih dahulu. Pada kasus ini, void menunjukkan bahwa *method* tidak mengembalikan data apa pun ketika telah selesai eksekusi. void adalah *keyword* dalam bahasa Dart yang hanya dapat digunakan secara spesifik. Kita akan mempelajari tipe data pada pertemuan berikutnya.
* Berikutnya adalah nama *function* —dalam hal ini, main. Nama tersebut digunakan oleh kode lain untuk merujuk pada nama *method* ini. Dalam kasus ini, main adalah nama *function* utama yang dicari oleh Dart VM saat pertama kali mengeksekusi kode. Setiap aplikasi Dart harus memiliki *function* main sehingga Dart VM tahu di mana harus memulai eksekusi kode.
* Tanda kurung kosong ( ) adalah tempat *function* untuk mendefinisikan data yang akan diterima. *Function* main ini tidak menerima data apa pun, oleh karena itu tanda kurung ini kosong. Kita akan lihat cara sebuah *function* dapat menerima data sebagai parameter pada pertemuan berikutnya.
* Terakhir, kurung kurawal { } di akhir baris pertama menentukan di mana kode *function* main dimulai, dan kurung kurawal penutup setelah beberapa baris kemudian untuk menentukan di mana kode *function* main berakhir. Berbeda dengan beberapa bahasa seperti Python, dia tidak menggunakan kurung kurawal.

| ***Function* versus *method***  *Function* dan *method* memiliki sintaks yang identik (aturan tentang struktur kodenya), dan sering kali penggunaan ***function*** dan ***method*** digunakan bergantian, lalu apa bedanya? Secara khusus, sebuah *function* berada di luar *class* (kita akan mempelajari tentang *class* pada pertemuan berikutnya). *Function* main adalah contoh di sini. Sebaliknya, sebuah *method* terikat pada turunan *class* dan memiliki referensi secara implisit ke *instance class* melalui *keyword* this. |
| --- |

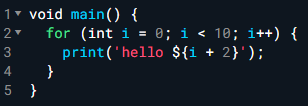
Jadi, Anda sekarang sudah tahu bahwa kode dapat dieksekusi karena Dart VM mencari *function* main, yang kemudian memanggil *function* tersebut.

# Tugas Praktikum

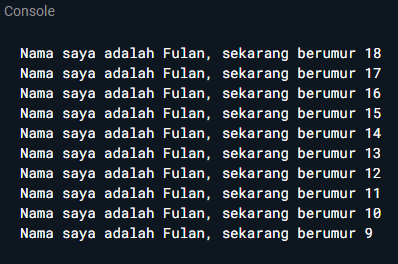
Duration: 120:00

## Soal 1

Modifikasilah kode pada baris 3 di VS Code atau Editor Code favorit Anda berikut ini agar mendapatkan keluaran (*output*) sesuai yang diminta!



Output yang diminta:



## Soal 2

Mengapa sangat penting untuk memahami bahasa pemrograman Dart sebelum kita menggunakan framework Flutter ? Jelaskan!

## Soal 3

Rangkumlah materi dari codelab ini menjadi poin-poin penting yang dapat Anda gunakan untuk membantu proses pengembangan aplikasi mobile menggunakan framework Flutter.

## Soal 4

Buatlah slide yang berisi penjelasan dan contoh eksekusi kode tentang perbedaan Null Safety dan Late variabel ! **(Khusus soal ini kelompok berupa link google slide)**

Kumpulkan jawaban Anda di spreadsheet pada tautan yang telah disediakan di grup telegram. Untuk soal nomor 1 sampai 3 push repo GitHub Anda.

# Selamat!

Duration: 0:00

Selamat Anda telah menyelesaikan Codelab ini sebagai langkah awal untuk memahami bahasa pemrograman Dart yang digunakan dalam framework Flutter.

## Apa selanjutnya?

Silakan cek beberapa sumber belajar lainnya…

* [FlutLab is a modern Flutter online IDE](https://flutlab.io/)
* [Flutter System Architecture](https://docs.google.com/presentation/d/1cw7A4HbvM_Abv320rVgPVGiUP2msVs7tfGbkgdrTy0I/edit#slide=id.p)
* [The Dart type system](https://dart.dev/guides/language/type-system)
* [DartPad in tutorials: Best practices](https://dart.dev/resources/dartpad-best-practices)
* [Pengantar Dart untuk Developer Java (Codelab)](https://developers.google.com/codelabs/from-java-to-dart)
* <https://dart.dev/codelabs>
* <https://dart.dev/resources/videos>

## Referensi

* [Flutter for Beginners Second Edition](https://drive.google.com/file/d/1hJbctx-_O3B6kUeG70bJD5xfbnsWwBXY/view?usp=sharing)
* <https://www.youtube.com/flutterdev>
* <https://dart.dev/guides/language/language-tour>
* <https://github.com/PacktPublishing/Flutter-for-Beginners-Second-Edition>
* Buku Ajar Pemrograman Mobile Dengan Flutter, Jurusan Teknologi Informasi, Politeknik Negeri Malang. 2021.