

Modul 1 Pemrograman Mobile

Bahasa Pemrograman Dart



Dart

Apa itu Bahasa Pemrograman Dart

Dart merupakan programming language lintas platform atau platform independen yang artinya dapat dijalankan pada sistem operasi yang berbeda seperti Windows, Linux, Unix dan MacOS, dll yang awalnya dikembangkan oleh Google dan kemudian disetujui sebagai standar oleh Ecma, yang saat ini digunakan untuk membangun aplikasi web, server, desktop, dan seluler.

Dart awalnya dirancang sebagai programming language yang dioptimalkan klien untuk pengembangan cepat aplikasi web dan seluler. Sebagai salah satu dari banyaknya programming language yang mendukung multi paradigma, Dart, bersifat imperatif, fungsional, reflektif dan berorientasi objek. Selain itu, Dart juga mengikuti semua konsep pendekatan pemrograman berorientasi objek seperti kelas, pewarisan, abstraksi, enkapsulasi, dan polimorfisme, dll. Dart juga merupakan tipe bahasa pemrograman yang sangat kuat yang menyediakan fitur pengumpul sampah otomatis. Bahasa Dart sendiri bersifat opensource yang dilisensikan di bawah BDS. Sintaks nya merupakan sintaksis gaya-C sederhana.

Pengembangan Bahasa Pemrograman Dart

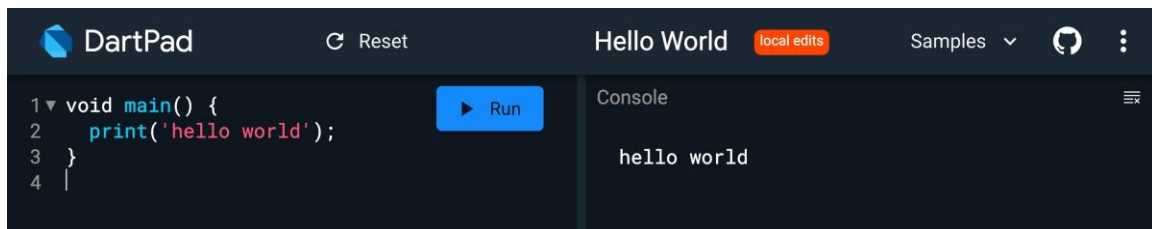
Bahasa pemrograman ini dikembangkan untuk mudah digunakan dalam pengembangan, sesuai dengan pengembangan aplikasi modern. dan memiliki implementasi berkinerja tinggi. Bahkan, bahasa ini juga dapat digunakan sebelum dikompilasi.

- Dart merupakan bahasa yang wajib Anda kuasai untuk mengembangkan aplikasi Flutter.
- Dengan Dart, Anda hanya perlu satu codebase untuk dapat mengembangkan aplikasi pada berbagai platform seperti web, Android, dan iOS.
- Bahasa Dart dirancang supaya familier dengan bahasa pemrograman lain sehingga mudah bagi yang sudah mengerti bahasa pemrograman lain maupun bagi yang baru memulai perjalanannya sebagai developer.
- Penggunaan bahasa Dart itu gratis (open source) dan dikembangkan oleh komunitas developer ahli yang aktif dan terbuka.
- Dart adalah bahasa yang dioptimalkan untuk pengembangan UI secara cepat dan produktif pada banyak platform.

Praktik Menggunakan Bahasa Pemrograman Dart

Bahasa Pemrograman Dart bisa di coba dengan cara online menggunakan browser, situs untuk mencoba Bahasa pemrograman dart adalah <https://dartpad.dev> , anda bisa mencoba langsung pada situs tersebut.

- **Program Hello World**



The screenshot shows the DartPad web interface. At the top, there's a 'DartPad' logo, a 'Reset' button, and the title 'Hello World' with a 'local edits' indicator. Below the title are 'Samples' and a menu icon. The main area is split into two panels. The left panel contains a Dart code snippet:

```
1 void main() {  
2   print('hello world');  
3 }  
4
```

 A blue 'Run' button is to the right of the code. The right panel, titled 'Console', shows the output 'hello world'.

- **Tipe data pada Dart**

Tipe data adalah konsep fundamental dalam pemrograman yang menentukan jenis nilai yang dapat disimpan dalam suatu variabel. Setiap tipe data memiliki sifat-sifat tertentu yang menggambarkan bagaimana nilai-nilai tersebut dapat digunakan, dioperasikan, dan dimanipulasi dalam program. Tipe data **int** mewakili angka bulat (bilangan bulat) tanpa komponen decimal, Tipe data **double** mewakili angka pecahan (bilangan desimal), Tipe data **String** mewakili rangkaian karakter atau teks. Tipe data **bool** mewakili nilai kebenaran (benar atau salah / true atau false) dan Tipe data **var** memungkinkan Dart secara otomatis menentukan tipe data berdasarkan nilai yang diberikan pada saat inisialisasi. Namun, setelah tipe data ditentukan, variabel tidak dapat mengubah tipe data tersebut.



The screenshot shows the DartPad web interface with a more complex Dart program. The code in the left panel is:

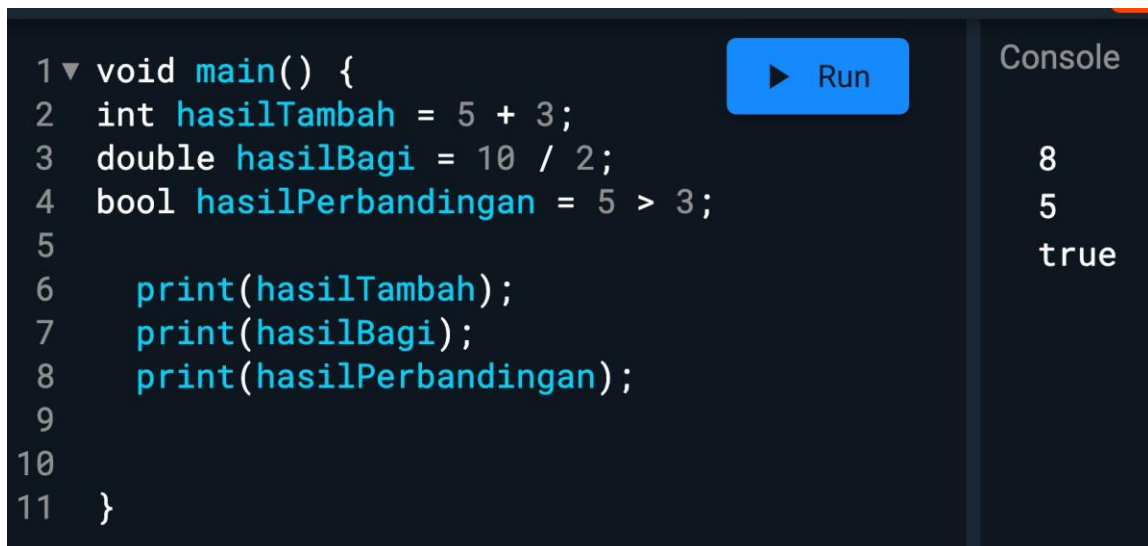
```
1 void main() {  
2   int angka = 10;  
3   double desimal = 3.14;  
4   String teks = "Halo, dunia!";  
5   bool benar = true;  
6   print(angka);  
7   print(desimal);  
8   print(teks);  
9   print(benar);  
10  
11 }
```

 A blue 'Run' button is to the right. The right panel, titled 'Console', shows the output of the program:

```
10  
3.14  
Halo, dunia!  
true
```

- **Operasi Matematika dan Logika**

Di dalam bahasa pemrograman Dart, Anda dapat melakukan berbagai operasi matematika dan logika untuk melakukan perhitungan dan pengambilan keputusan. Berikut ini adalah beberapa contoh operasi matematika dan logika yang dapat Anda gunakan dalam program Dart:



The screenshot shows a code editor with the following Dart code:

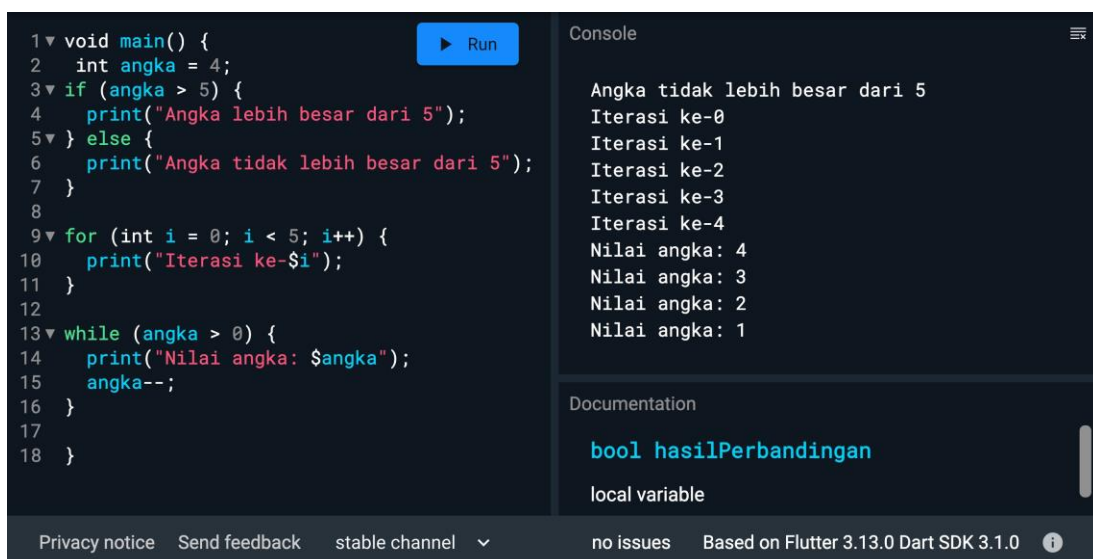
```
1 void main() {  
2   int hasilTambah = 5 + 3;  
3   double hasilBagi = 10 / 2;  
4   bool hasilPerbandingan = 5 > 3;  
5  
6   print(hasilTambah);  
7   print(hasilBagi);  
8   print(hasilPerbandingan);  
9  
10  
11 }
```

A blue "Run" button is visible next to the code. To the right, a "Console" panel displays the output of the program:

```
8  
5  
true
```

- **Struktur Kontrol**

Struktur kontrol dan percabangan adalah elemen penting dalam pemrograman untuk mengatur alur eksekusi program berdasarkan kondisi tertentu. Di bahasa pemrograman Dart, Anda dapat menggunakan beberapa pernyataan kontrol dan percabangan untuk mengendalikan alur program.



The screenshot shows a code editor with the following Dart code:

```
1 void main() {  
2   int angka = 4;  
3   if (angka > 5) {  
4     print("Angka lebih besar dari 5");  
5   } else {  
6     print("Angka tidak lebih besar dari 5");  
7   }  
8  
9   for (int i = 0; i < 5; i++) {  
10    print("Iterasi ke-$i");  
11  }  
12  
13  while (angka > 0) {  
14    print("Nilai angka: $angka");  
15    angka--;  
16  }  
17  
18 }
```

A blue "Run" button is visible next to the code. To the right, a "Console" panel displays the output of the program:

```
Angka tidak lebih besar dari 5  
Iterasi ke-0  
Iterasi ke-1  
Iterasi ke-2  
Iterasi ke-3  
Iterasi ke-4  
Nilai angka: 4  
Nilai angka: 3  
Nilai angka: 2  
Nilai angka: 1
```

Below the console, a "Documentation" panel shows the definition of the `bool hasilPerbandingan` local variable.

At the bottom of the IDE, there is a status bar with links for "Privacy notice", "Send feedback", "stable channel", and "no issues". It also indicates the version: "Based on Flutter 3.13.0 Dart SDK 3.1.0".

- **Fungsi**

Di bahasa pemrograman Dart, fungsi adalah blok kode yang dapat dipanggil dan dieksekusi untuk melakukan tugas tertentu. Fungsi membantu dalam mengorganisir kode menjadi bagian-bagian yang lebih kecil, lebih mudah dikelola, dan dapat digunakan kembali. Fungsi juga membantu dalam memecah logika program menjadi unit-unit yang terpisah, meningkatkan modularitas, dan mengurangi pengulangan kode.



```
1 ▾ int tambah(int a, int b) {  
2   return a + b;  
3 }  
4  
5 ▾ void main() {  
6   int hasil = tambah(3, 7);  
7   print("Hasil penjumlahan: $hasil");  
8 }
```

Run

Console

Hasil penjumlahan: 10

- **Kelas dan Objek**

Di dalam pemrograman berorientasi objek, seperti bahasa Dart, konsep kelas dan objek sangat penting. Mereka memungkinkan Anda untuk mengorganisir dan mengelola kode Anda dengan lebih baik melalui abstraksi. Mari kita bahas lebih detail:

Kelas:

Kelas adalah blueprint atau template untuk menciptakan objek. Itu berisi definisi atribut (variabel) dan perilaku (metode) yang akan dimiliki oleh objek yang dibuat dari kelas tersebut. Dalam kelas, Anda mendefinisikan karakteristik umum yang akan dimiliki oleh objek-objek yang terkait.

Objek:

Objek adalah instansi konkret yang diciptakan dari kelas. Mereka mewakili entitas nyata atau konsep dalam dunia nyata dan dapat memiliki keadaan (nilai atribut) serta perilaku (metode) yang didefinisikan oleh kelas. Objek adalah hasil dari pembuatan dan inisialisasi kelas.

```
1 class Mahasiswa {
2   String nama;
3   int usia;
4
5   Mahasiswa(this.nama, this.usia);
6
7   void perkenalan() {
8     print("Halo, saya $nama dan saya berusia $usia tahun.");
9   }
10 }
11
12 void main() {
13   Mahasiswa mhs = Mahasiswa("Alice", 22);
14   mhs.perkenalan();
15 }
16
```

Run

Console

Halo, saya Alice dan saya berusia 22 tahun.

- **Import**

Dalam bahasa pemrograman Dart, pernyataan import digunakan untuk mengimpor kode dari file atau pustaka lain ke dalam file saat ini. Hal ini memungkinkan Anda menggunakan kode yang didefinisikan di tempat lain tanpa perlu menulis ulang kode tersebut. Pernyataan import adalah bagian penting dari modularitas dalam pengembangan perangkat lunak.

Pada dasarnya, import memungkinkan Anda membagi kode Anda menjadi beberapa file yang lebih kecil dan lebih terfokus. Ini membuat kode lebih mudah dikelola dan memungkinkan reusabilitas yang lebih tinggi.

```
1 import 'dart:math';
2
3 void main() {
4   int acak = Random().nextInt(10);
5   print("Angka acak: $acak");
6 }
7
```

Run

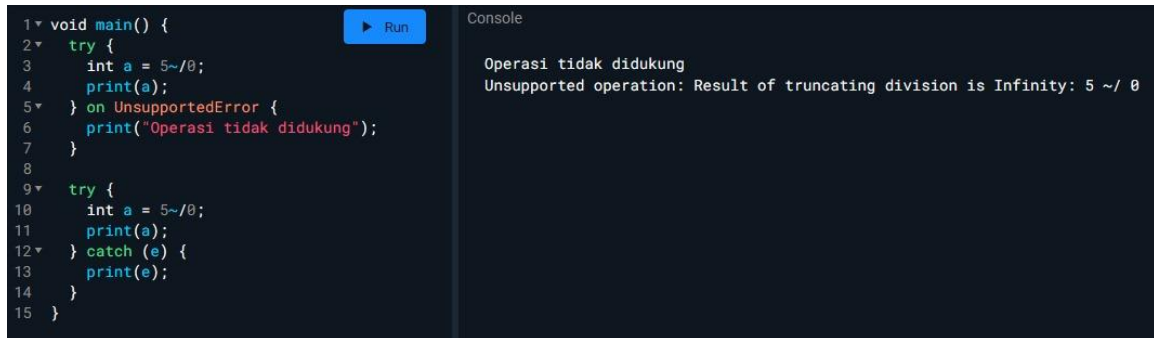
Console

Angka acak: 9

- **Eksepsi**

Fungsi eksepsi (exception) dalam bahasa pemrograman Dart digunakan untuk mengatasi situasi yang tidak terduga atau kesalahan yang dapat terjadi selama eksekusi program. Ketika suatu kesalahan terjadi, Dart akan menciptakan objek eksepsi yang mencerminkan jenis kesalahan tersebut. Anda kemudian dapat menangani eksepsi ini dengan menggunakan blok try dan catch untuk mengambil tindakan yang sesuai.

Contoh umum penggunaan fungsi eksepsi adalah ketika Anda berinteraksi dengan sumber daya eksternal (seperti berkas atau jaringan) yang mungkin tidak berfungsi seperti yang diharapkan. Berikut adalah struktur dasar penggunaan fungsi eksepsi:



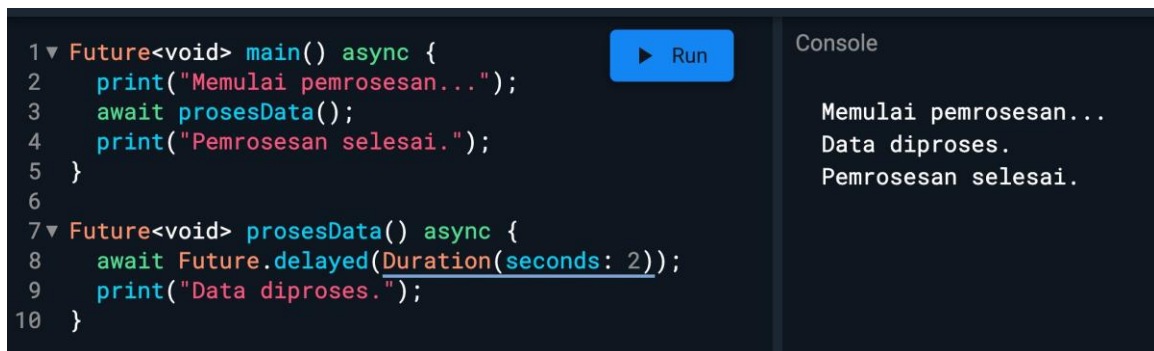
```
1 void main() {  
2   try {  
3     int a = 5~/0;  
4     print(a);  
5   } on UnsupportedError {  
6     print("Operasi tidak didukung");  
7   }  
8  
9   try {  
10    int a = 5~/0;  
11    print(a);  
12  } catch (e) {  
13    print(e);  
14  }  
15 }
```

Console

Operasi tidak didukung
Unsupported operation: Result of truncating division is Infinity: 5 ~/ 0

- **Async dan Await**

Di bahasa pemrograman Dart, fitur `async` dan `await` digunakan untuk mengelola kode yang bersifat asinkronus. Asinkronus berarti kode yang dapat berjalan tanpa harus menunggu tugas sebelumnya selesai. Ini sangat penting dalam situasi di mana tugas-tugas seperti operasi jaringan, operasi berkas, atau pemanggilan API dapat memakan waktu yang lama. Kata kunci `async` digunakan untuk mendeklarasikan bahwa suatu fungsi adalah fungsi asinkronus. Ini memungkinkan Anda menggunakan `await` di dalamnya. Kata kunci `await` digunakan di dalam fungsi yang dinyatakan sebagai `async`. Ini digunakan untuk menunggu hasil dari operasi asinkronus sebelum melanjutkan eksekusi kode.



```
1 Future<void> main() async {  
2   print("Memulai pemrosesan...");  
3   await prosesData();  
4   print("Pemrosesan selesai.");  
5 }  
6  
7 Future<void> prosesData() async {  
8   await Future.delayed(Duration(seconds: 2));  
9   print("Data diproses.");  
10 }
```

Console

Memulai pemrosesan...
Data diproses.
Pemrosesan selesai.