



Dart programski jezik

Sadržaj

O Dart-u	2
Upotreba.....	3
Kodiranje u Dart-u.....	4
Foreign function interface	8

O Dart-u

Dart je objektno orijentisani programski jezik optimiziran za klijente na raznim platformama. Razvio ga je Google, a koristi se za izradu mobilnih, desktop, backend i web aplikacija. Skalabilna je i kohezivna platforma za izgradnju aplikacija. Upotrebite jezik Dart, biblioteke i alate da biste napravili bilo šta, od jednostavnih skripti do gotovih aplikacija s mnoštvo opcija.

Dart je programski jezik koji se koristi za kodiranje Flutter aplikacija. Predstavlja softver otvorenog koda pod BSD licencom. Malo podseća na programski jezik C. Kada se koristi u web aplikacijama, prevodi se u JavaScript pa se može pokretati na svim web pretraživačima. Dart koristi "hot reload" kako biste odmah videli rezultat u vašoj pokrenutoj aplikaciji.

Dart-ovi ciljevi dizajna su:

- Stvoriti strukturirani, ali fleksibilni jezik za veb programiranje.
- Učiniti da se Dart programerima učini poznatim i prirodnim, a time i lakim za učenje.
- Uveriti se da Dart pruža visoke performanse u svim modernim veb pretraživačima i okruženjima, od malih ručnih uređaja do izvršenja na strani servera.

Dart cilja širok spektar razvojnih scenarija: od projekta za jednu osobu bez velike strukture do projekta velikog obima koji zahteva formalne tipove u kodu za navodjenje namera programera. Da bi podržao ovaj širok spektar projekata Dart ima opcione tipove: to znači da možete početi sa kodiranjem bez tipova i dodati ih kasnije po potrebi.

Upotreba

Postoji četiri glavna načina za izvršavanje Dart koda:

1. Preveden kao JavaScript

Da bi se pokrenuo u običnom veb pretraživaču Dart se oslanja na izvor-u-izvor kompajliranje (source-to-source) u JavaScript. Prema sajtu projekta "Dart je dizajniran da bude lak za pisanje razvojnih alata, pogodan savremenom razvoju aplikacija i sposoban za implementacije visokih performansi. Kada je pokrenut u veb pretraživaču Dart se prevodi u JavaScript pomoću dart2js kompajler. Kompajliran kao JavaScript Dart kod je kompatibilan sa svim glavnim pretraživačima bez potrebe da pretraživači implementiraju Dart. Kroz optimizaciju prevedenog JavaScript izlaza, izbegavajući skupe provere i operacije, kod napisan u Dartu može u nekim slučajevima raditi brže nego ekvivalentni kod pisan koristeći JavaScript idiome.

2. U Dart-ijum pretraživaču

Dart Softver Development Kit(SDK) se isporučuje sa verzijom Chromium veb pretraživača, modifikovanim da sadrži Dart virtuelnu masinu (VM). Ovaj pretraživač može pokrenuti Dart kod direktno bez prevodjenja u JavaScript. Namenjen je kao razvojni alat za aplikacije napisane u tom jeziku, a ne kao veb pretraživač opšte namene. Prvo je planirano da se podrška za Dart ugradi direktno u Gugl Kroum, ali je kasnije obustavljeno.

3. Samostalno

Dart SDK se isporučuje sa samostalnom Dart VM, omogućavajući da se Dart kod pokrene u okruženju interfejsa komandne linije. Kao jezik alati koji su uključeni u Dart SDK su napisani uglavnom u Dartu, samostalni Dart VM je kritični deo SDK. Ovi alati uključuju dart2js kompajler i paket upravljanja pod nazivom pub. Dart se isporučuje sa kompletnom standardnom bibliotekom koja omogućava korisnicima da pišu potpuno funkcionalne sistemske aplikacije, kao što su prilagodjeni veb server.

4. Unapred kompajliran

Dart kod može biti AOT-kompajliran u mašinski kod. Aplikacije napravljene koristeći "Flutter", SDK aplikacijom za mobilne uređaje izgradjenom na Dart-u, postavljen je su na prodavnice aplikacija kao AOT-kompajliran Dart kod.

Kodiranje u Dart-u

Dart je kao i Java, C, JavaScript, C# I drugi jezici, potomak ALGOL porodice jezika. Sintaksa kaskade metoda, koja pruža sintaksnu prečicu za povezivanje nekoliko metoda jednog za drugim na isti objekat, usvojena je od strane Smalltalk-a. Dart koristi izolate kao vlasničke I zaštitne jedinice pri strukturisanju aplikacija. Koncept izolata nadograđuje se na model Aktora, koji je najpoznatiji u jeziku Erlang.

Primeri:

1. Hello World

Svaka aplikacija ima main() funkciju. Da biste prikazali tekst na konzoli možete da koristite funkciju najvišeg nivoa print().

```
void main() {  
  print('Hello, World!');  
}
```

2. Promenljive

Čak I u tipičnom Dart kodu, većini promenljivama nisu potrebni eksplicitni tipovi, zahvaljujući zaključivanju tipa:

```
var name = 'Voyager I';  
var year = 1977;  
var antennaDiameter = 3.7;  
var flybyObjects = ['Jupiter', 'Saturn', 'Uranus', 'Neptune'];  
var image = {  
  'tags': ['saturn'],  
  'url': '//path/to/saturn.jpg'  
};
```

3. Funkcije

Preporučuje se da navedete tipove argumenata svake funkcije i povratnu vrednost:

```
int fibonacci(int n) {  
    if (n == 0 || n == 1) return n;  
    return fibonacci(n - 1) + fibonacci(n - 2);  
}  
  
var result = fibonacci(20);
```

4. Komentari

Dart komentari obično počinju sa //.

```
// This is a normal, one-line comment.  
  
/// This is a documentation comment, used to document libraries,  
/// classes, and their members. Tools like IDEs and dartdoc treat  
/// doc comments specially.  
  
/* Comments like these are also supported. */
```

5. Uvoz

Da biste pristupili API-ima definisanim u drugim bibliotekama, koristite import .

```
// Importing core libraries  
import 'dart:math';  
  
// Importing libraries from external packages  
import 'package:test/test.dart';  
  
// Importing files  
import 'path/to/my_other_file.dart';
```

6. Klase

Evo primera klase sa tri svojstva, dva konstruktora i metodom. Jedno od svojstava ne može se direktno podesiti, pa se definise pomoću metode getera (umesto promenljive).

```
class Spacecraft {
  String name;
  DateTime launchDate;

  // Constructor, with syntactic sugar for assignment to members.
  Spacecraft(this.name, this.launchDate) {
    // Initialization code goes here.
  }

  // Named constructor that forwards to the default one.
  Spacecraft.unlaunched(String name) : this(name, null);

  int get launchYear =>
    launchDate?.year; // read-only non-final property

  // Method.
  void describe() {
    print('Spacecraft: $name');
    if (launchDate != null) {
      int years =
        DateTime.now().difference(launchDate).inDays ~/
          365;
      print('Launched: $launchYear ($years years ago)');
    } else {
      print('Unlaunched');
    }
  }
}
```

7. Nasleđivanje

Dart ima jedno nasledstvo.

```
class Orbiter extends Spacecraft {
  double altitude;
  Orbiter(String name, DateTime launchDate, this.altitude)
    : super(name, launchDate);
}
```

8. Interfejsi i apstraktne klase

Dart nema ključnu reč za interfejs. Umesto toga, sve klase implicitno definišu interfejs. Stoga možete primeniti bilo koju klasu.

```
class MockSpaceship implements Spacecraft {
  // ...
}
```

Možete napraviti apstraktnu klasu koju će proširiti (ili primeniti) konkretna klasa. Apstraktne klase mogu sadržati apstraktne metode (sa praznim telima).

```

abstract class Describable {
    void describe();

    void describeWithEmphasis() {
        print('=====');
        describe();
        print('=====');
    }
}

```

9. Mixins

Mixins su način ponovne upotrebe koda u više hijerarhija klasa. Sledi mešovita deklaracija:

```

mixin Piloted {
    int astronauts = 1;
    void describeCrew() {
        print('Number of astronauts: $astronauts');
    }
}

```

Da biste klasi dodali mogućnost kombinacije, samo proširite klasu kombinacijom.

```

class PilotedCraft extends Spacecraft with Piloted {
    // ...
}

```


Foreign function interface

Interfejs strane funkcije (FFI) je mehanizam pomoću kojeg program napisan na jednom programskom jeziku može pozivati rutine ili koristiti usluge napisane na drugom. Dart mobilne, komandne linije i serverske aplikacije pokrenute na Dart Native platformi mogu koristiti biblioteku `dart:ffi` za pozivanje C API-ja.

Sledeći primeri pokazuju kako se koristi biblioteka `dart:ffi`:

- [hello_world](#) – Kako pozvati funkciju C bez argumenata i bez povratne vrednosti.
- [primitives](#) – Kako pozvati C funkciju koje imaju argumente i vraćaju vrednosti koje su celobrojne ili pokazivači.
- [structs](#) – kako se koriste strukture za prosleđivanje stringova u i iz C i za rukovanje složenim i jednostavnim C strukturama.