Linguagem de Programação Dart

Trilha de Flutter

Danilo Perez

Full Stack Developer / Criador do canal “Fala Devs” no Youtube

Microsoft Certified Professional Developer e Pós Graduado em Java com Oracle

https://github.com
/perez-danilo

Danilo Perez

Full Stack Developer
@in/perez-danilo

Criador do canal “Fala Devs” no Youtube
Microsoft Certified Professional Developer
Pós Graduado em Java com Oracle

Objetivo Geral

Objetivo deste módulo é apresentar a linguagem de programação Dart que é o que usamos para desenvolver em Flutter. Será abordado o fundamento, orientação a objetos, estrutura condicionais e de repetição, funções, boas práticas, tratamento de erros e testes.

Percurso

Etapa 1	Fundamentos da Linguagem Dart
Etapa 2	Estruturas Condicionais e de Repetição em Dart
Etapa 3	Dominando Funções em Dart
Etapa 4	Orientação a Objetos em Dart
Etapa 5	Boas Práticas e Tratamento de Exceções em Dart
Etapa 6	Introdução à Testes em Dart

Etapa 1

Flutter

// Fundamentos da Linguagem Dart

Introdução

Dart é uma linguagem de programação apresentada pelo Google me 2011 com o objetivo de ser uma opção ao TypeScript para desenvolvimento Web.

Começou a ser mais difundida com o advento do Flutter que trouxe o Dart como linguagem para seu SDK.

Dart VM

https://dart.dev

Introdução

- Dart é uma linguagem muito parecida com o a linguagem C, com isso ela lembra muito Java, C#, Javascript e PHP.
- Linguagem fortemente tipada, mas que possibilita o uso também de tipos dinâmicos
- Orientada a objetos
- DartPad (https://dartpad.dartlang.org/)

Compilado x Interpretado

- Pode ser compilada de duas formas diferentes: ahead-of-time (AOT) e just-in-time (JIT)
- ahead-of-time: quando o Código já é compilado para a linguagem nativa, possibilitando uma alta performance
- just-in-time: código é compilado com a aplicação em execusão, possibilidtando o hot-reload

Aplicabilidade do Dart

- Console
- REST
- WebSocket
- Flutter

Instalação

- Flutter.
- Site https://dart.dev
- Windows
- Mac
- Linux

Criação de projeto Dart

dart create <NOME_DIRETORIO>

dart create meu_app

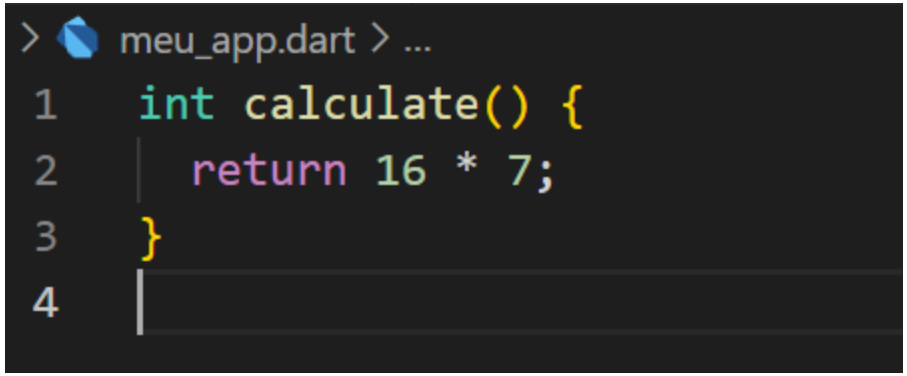
Uso do Dart

- Console
- Pacotes
- REST
- WebSocket

Linguagem Dart

- Lógicos
  - ==
  - !=
  - &&
  - ||
- Matemáticos
  - +
  - -
  - *
  - /
  - %
- Condicionais
  - if
  - else
  - else if
  - Ternário
  - switch case
- Laços
  - for
  - foreach
  - while
  - do while

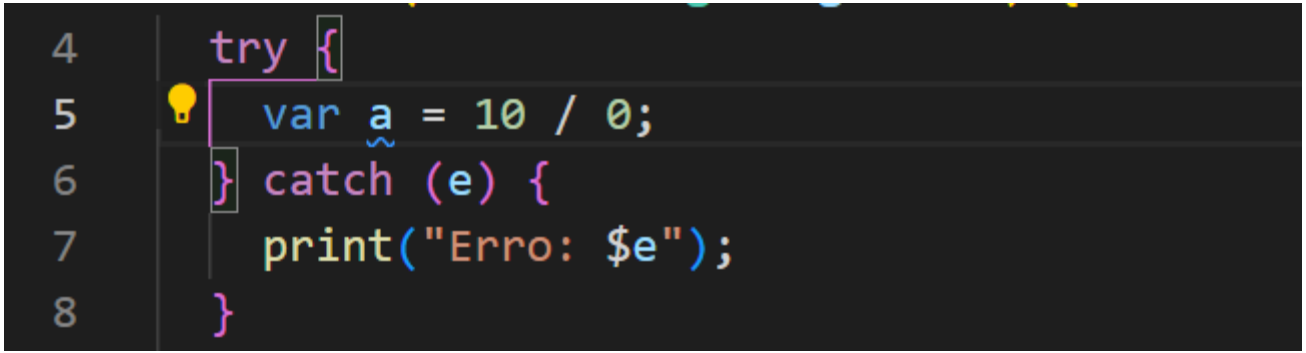
Funções



Orientação a Objetos

```
1  class Carro {
2    int _rodas = 0;
3    String _cor = "";
4
5    Carro(String cor, int rodas) {
6      this._cor = cor;
7      this._rodas = rodas;
8    }
9  }
```

Exceções



Testes

```
Run | Debug
4  void main() {
     Run | Debug
5    test('calculate', () {    Expected: <12> Actual: <112> package:test_api
6      expect(calculate(), 12);
7    });
8  }
9
```

```
Run | Debug
4  void main() {
     Run | Debug
5    test('calculate', () {
6      expect(calculate(), 112);
7    });
8  }
9
```

Tipo de dados

- int
- double
- String
- bool
- List e List<>
- Map – chave/valor
- constant
- Dynamic
- Date

Hands On!

“Falar é fácil. Mostre-me o código!”

Linus Torvalds

Etapa 2

Flutter

// Estruturas Condicionais e de Repetição em Dart

Linguagem Dart

- Lógicos
  - ==
  - !=
  - > e <
  - &&
  - ||
- Condicionais
  - if
  - else
  - else if
  - Ternário
  - switch case
- Laços
  - for
  - foreach
  - while
  - do while

Etapa 3

Flutter

// Dominando Funções em Dart

Benefícios das funções

- Reduzindo a duplicação de código
- Melhorar a clareza do código
- Reutilização de código
- Decompondo problemas complexos em partes mais simples
- Ocultação de informações

Estrutura de uma função

```
// Função
int add(int a, int b)
{
    int result = a + b;
    return result;
}

int add(int a, int b) => a + b;

// Função sem parametros e sem retorno
void hello() => print("Hello World");

void hello() {
    print("Hello World");
}
```

Etapa 4

Flutter

// Orientação a Objetos em Dart

O que é OO

A Orientação a Objetos é um paradigma de computação que nos auxilia a efetuar abstrações de objetos e outras coisas imateriais do mundo real. Essas abstrações serão escritas em forma de estruturas de fácil compreensão, estruturas essas que servirão de modelo para criação de nossos dados dentro dos sistemas.

Princípios da OO

- Programação Estruturada vs Programação Orientada a Objetos
- Abstração
- Encapsulamento
- Herança
- Polimorfismo

Detalhamento da OO

- Classes
- Herança
- Objetos
- Encapsulamento
- Polimorfismo
- Classes Abstratas
- Interfaces
- Inversão de Controle e Injeção de dependência
- SOLID
- DDD — Domain Driven Design

Etapa 5

Flutter

// Boas Práticas e Tratamento de Exceções em Dart

Tratamento de erros

- Try catch
- Try catch finally

```
try {
  // ...
} on ExcessaoEspecifica catch(e) {
  //Trata Exceção específica
} catch(e) {
  //Trata qualquer outra exceção
}
```

Boas práticas

- lints
- https://pub.dev/packages/lints
- https://pub.dev/packages/flutter_lints

Etapa 6

Flutter

// Introdução à Testes em Dart

Introdução a testes

- Evitar erros antes de ir para produção
- Redução de custos com testes manuais
- Confiabilidade do código

Testes no DART

- dart run test
- Utilização de Matchers
- Confiabilidade do código

```
Run | Debug
test('Calcula o valor do produto com desconto sem porcentagem', () {
  expect(app.calcularDesconto(1000, 150, false), equals(850));
});
```

Links Úteis

- digitalinnovationone/dio-flutter (github.com)
- Dart programming language | Dart
- DIO: Cursos de Orientação a objetos
- DIO: Testes

Para saber mais

Artigos e cursos da DIO

“Fala Devs” youtube

Dúvidas?

> Fórum/Artigos

> Comunidade Online (Discord)

Desafio - IMC

- Criar classe Pessoa (Nome / Peso / Altura)
- Ler dados do terminal
- Tratar exceções
- Calcular IMC
- Printar na tela o resultado do cálculo
- Testes

IMC	Classificação
< 16	Magreza grave
16 a < 17	Magreza moderada
17 a < 18,5	Magreza leve
18,5 a < 25	Saudável
25 a < 30	Sobrepeso
30 a < 35	Obesidade Grau I
35 a < 40	Obesidade Grau II (severa)
≥ 40	Obesidade Grau III (mórbida)

$$IMC = \frac{Peso \text{ (em quilos)}}{Altura^2 \text{ (em metros)}}$$