Documento: dart-estudos

Criação: 03/10/2020

Autor: Ricardo Cardoso Pompêo

Objetivo: Documentar o estudo da linguagem de programação Dart.

Horas de estudo: 07h11min.

03/10/2020 - 17h09min | Start

Canal de estudos: Flutter e Dart - Udemy

VS Code - Plugin Code Runner

Este plugin facilita o desenvolvimento do Dart, onde torna-se necessário configurar o PATH com o interpretador da linguagem.

Teclas de atalho:

* ctrl+alt+N -> para rodar o programa.
* ctrl+alt+M -> para parar o programa.

O código não é executado caso parte dele esteja selecionado, ele também não é executado caso o cursor esteja no console, com exceção do comando ctrl+alt+M para parar o código.

Fundamentos

Primeiro programa

Quando vamos aprender uma nova linguagem devemos nos perguntar como um programa é organizado nessa linguagem, isso refere-se a como é organizada sua simbologia, blocos e sentenças.

Se fossemos explicar um livro para alguém que nunca viu um, diríamos que a junção de símbolos que são as letras formam palavras, e a junção de palavras formam frases, várias frases formam um texto que juntos formam capítulos e por fim temos um livro. Na programação não é diferente, basta pensar analogamente.

A finalização obrigatória de uma sentença em Dart é feita com ponto e vírgula **“;”** e também há a possibilidade de quebrarmos as linhas dessa sentença, por fim, a definição de bloco é feita pelas chaves.

Assim entendemos que a programação é baseada em blocos e sentenças de código.

* Sentença de código: é uma linha de código que termina com ponto e vírgula;
* Bloco de código: é um conjunto de sentenças de código.

*Nota: Podemos criar sentenças e blocos vazios, o que não faz sentido, porém é possível.*

A porta de entrada dos programas em Dart é o método ***main***. Ele é obrigatório para que o nosso programa funcione.

main() {

print('Olá Dart!');

}

Variáveis

Variáveis em Dart são fortemente tipadas, assim definimos o seu tipo, o nome e a sua atribuição de valor. Também podemos usar valores literais, mas o comum é atribuí-los a variáveis.

main() {

// Definição da variável "a" com atribuição do valor 2.

int a = 2;

print(a);

// Impressão do resultado de valores literais que somam 9.

print(1 + 2 \* 4);

}

Também podemos declarar variáveis em Dart utilizando a palavra reservada ***var***, isso faz com que o compilador atribua internamente o tipo da variável por ***inferência***. Perceba que utilizando a palavra reservada ***var*** na declaração o tipo da variável não pode ser alterado no decorrer do código.

main() {

// Declaração de variáveis utilizando var.

var n1 = 2;

var n2 = 4.56;

// Tentativa de atribuição de texto em uma variável inteira.

n1 = "texto";

// Impressão da soma dos valores

print(n1 + n2);

}

O método ***runtimeType*** possibilita saber qual tipo da variável atribuída por ***inferência***.

main() {

// Declaração de variáveis utilizando var.

var n1 = 2;

var n2 = 4.56;

var t1 = "Texto";

// Impressão da soma dos valores.

print(n1 + n2);

// runtimeType: retorna o tipo da variável declarada.

print(n1.runtimeType);

print(n2.runtimeType);

print(t1.runtimeType);

}

6.56

int

double

String

Constantes

Para o conceito de constantes vamos calcular a área da circunferência. Mais antes vamos compreender que o Dart utiliza de bibliotecas importadas, uma delas permite capturar dados digitados pelo usuário.

import 'dart:io';

main() {

// Área da circunferência = PI \* raio \* raio

// Variável texto com atribuição do método readLineSync

// proveniente da biblioteca dart:io.

var texto = stdin.readLineSync();

print("O valor digitado é: " + texto);

}

teste

O valor digitado é: teste

Modificando algumas informações no código acima o usuário pode nos fornecer o valor do raio. Aqui utilizamos o método ***toString*** para que no final de toda ação o valor do raio seja convertido para uma ***String***.

import 'dart:io';

main() {

// Área da circunferência = PI \* raio \* raio

var entradaDoUsuario = stdin.readLineSync();

var raio = double.parse(entradaDoUsuario);

print("O valor raio é: " + raio.toString());

}

4.5

O valor raio é: 4.5

Sabendo que o valor do raio não terá alteração no decorrer do código entendemos que o valor dele é constante, ou seja, não se altera, e para defini-lo como uma constante utilizamos a palavra reservada ***final***.

Também podemos declarar a variável constante definindo seu tipo.

Outro ponto interessante é que dentro da biblioteca ***dart:io*** temos os métodos:

* ***stdin.readLineSync();***
  + Usado para entrada de dados.
* ***stdout.write();***
  + Usado para saída de dados com a informação na mesma linha.

Onde ***std*** significa standard (Padrão), ***in*** vem de input (entrada) e ***out*** vem output (saída). Entrada e saída padronizada de dados.

É interessante usarmos valores constantes no código, pois isso evita bugs por mudança nos valores das variáveis, sabemos que haverá variáveis que necessitam de mudanças, mas se puder evitá-las será melhor.

import 'dart:io';

main() {

// Área da circunferência = PI \* raio \* raio

stdout.write("Informe o raio: ");

var entradaDoUsuario = stdin.readLineSync();

final double raio = double.parse(entradaDoUsuario);

print("O valor do raio é: " + raio.toString());

}

Informe o raio: 4.5

O valor do raio é: 4.5

O Dart possui duas palavras reservadas para definição de constantes, o ***final*** e o ***const***.

O ***final*** define uma constante em tempo de execução, ou seja, como não sabemos qual será o valor digitado pelo usuário para o raio, essa constante só será definida quando o código for executado.

Já o ***const*** define um constante em tempo de compilação, veja que no nosso exemplo o valor de PI já está definido e portanto o Dart irá declarar essa constante com antecedência pois ele já sabe qual o é o valor atribuído.

Essa regra não permite a criação de constantes ***final*** oriundas de uma ***const*** e vice-versa, pois o conceito de ambas são distintos. É importante notar também que não podemos criar constantes a partir de uma variável.

import 'dart:io';

main() {

// Área da circunferência = PI \* raio \* raio

const PI = 3.1415;

stdout.write("Informe o raio: ");

final entradaDoUsuario = stdin.readLineSync();

final double raio = double.parse(entradaDoUsuario);

final area = PI \* raio \* raio;

print("O valor da área é: " + area.toString());

}

Informe o raio: 100

O valor da área é: 31415.000000000004

03/10/2020 - 00h20min | Finish

Horas estudo: 07h11min.

04/10/2020 - 16h26min | Start

Tipos Básicos - 1

Iniciaremos com os tipos básicos:

* ***int*** e ***double***: para números;
* ***String***: para textos;
* ***bool***: para verdadeiro e falso;
* **dynamic**: para valores dinâmicos.

Todos os tipo básicos em Dart permitem a utilização da notação ponto **“.”** onde é possível obter funcionalidades dos valores ou das variáveis.

main() {

int n1 = 3;

double n2 = -5.67.abs();

}

No exemplo acima a notação ponto no valor -5.67 permite pegar o valor absoluto do valor literal.

Se somarmos um tipo ***int*** com um tipo ***double*** o Dart irá nos devolver um valor ***double***, porque esse tipo de variável pode armazenar mais informações que um tipo ***int***. Nós também podemos converter um valor ***String*** para ***double*** utilizando o método ***parse()***.

main() {

int n1 = 3;

double n2 = (-5.67).abs();

double n3 = double.parse("12.765");

print(n1 + n2 + n3);

}

21.435000000000002

Há um outro tipo de variável que se chama ***num***, ele é o “pai” do ***int*** e do ***double***, pois ambos herdam de ***num***, ele permite a atribuição de números inteiros e de ponto flutuante, mas é pouco usado.

main() {

int n1 = 3;

double n2 = (-5.67).abs();

double n3 = double.parse("12.765");

num n4 = 6;

print(n1 + n2 + n3 + n4);

n4 = 6.7;

print(n1 + n2 + n3 + n4);

}

27.435000000000002

28.135

*Nota: Nós podemos entrar nas classes dos tipos das variáveis segurando o* ***ctrl e clicando no tipo****.*

Para ***Strings*** não há muito segredo, o diferencial está em declará-la como ***String*** e o valor deve estar entre aspas simples ou duplas. Podemos ***concatená-las*** e utilizar a notação ponto **“.”** com o método ***toUpperCase()*** para, por exemplo, deixar todas as letras maiúsculas.

main() {

int n1 = 3;

double n2 = (-5.67).abs();

double n3 = double.parse("12.765");

num n4 = 6;

print(n1 + n2 + n3 + n4);

n4 = 6.7;

print(n1 + n2 + n3 + n4);

String s1 = "Bom";

String s2 = " dia";

print(s1 + s2.toUpperCase() + "!!!");

}

Bom DIA!!!

O tipo ***bool*** trabalha com valores verdadeiros ou falsos, a partir deles podemos fazer comparações com retorno de valores. Essas comparações utilizam os símbolos ***&&*** e ***||***.

main() {

int n1 = 3;

double n2 = (-5.67).abs();

double n3 = double.parse("12.765");

num n4 = 6;

print(n1 + n2 + n3 + n4);

n4 = 6.7;

print(n1 + n2 + n3 + n4);

String s1 = "Bom";

String s2 = " dia";

print(s1 + s2.toUpperCase() + "!!!");

bool estaChovendo = true;

bool muitoFrio = false;

print(estaChovendo || muitoFrio);

print(estaChovendo && muitoFrio);

}

true

false

O tipo dynamic é exatamente o que o nome diz, ele permite que a variável definida com este tipo seja dinâmica, ou seja, ela pode assumir qualquer valor seja ***int***, ***double*** e ***String***.

main() {

int n1 = 3;

double n2 = (-5.67).abs();

double n3 = double.parse("12.765");

num n4 = 6;

print(n1 + n2 + n3 + n4);

n4 = 6.7;

print(n1 + n2 + n3 + n4);

String s1 = "Bom";

String s2 = " dia";

print(s1 + s2.toUpperCase() + "!!!");

bool estaChovendo = true;

bool muitoFrio = false;

print(estaChovendo || muitoFrio);

print(estaChovendo && muitoFrio);

dynamic x = "Um texto bem legal";

print(x);

x = 123;

print(x);

x = false;

print(x);

}

Um texto bem legal

123

false

Tipos básicos - 2

A partir deste ponto as anotações ficaram como comentários dentro do código dos exercícios. Consulte por lá!