



Dart - Introdução

- Características gerais:
 - Linguagem de programação criação pela empresa Google;
 - Fortemente utilizada para criar apps para dispositivos móveis;
 - •É cross-platform, ou seja, permite criar apps para Android e iOS;
 - Também permite criar para desktop e web;
 - Semelhante ao Java em alguns aspectos;
 - Utiliza uma VM chamada Dart VM permite hot reload ao desenvolver;

Prof. David

Dart - Introdução

- Plataformas:
 - Nativa
 - Para apps que serão executados em dispositivos móveis ou desktop;
 - Dart VM com compilação just-in-time (JIT) e compilador ahead-of-time (AOT);
 - Web:
 - Para apps que tem como alvo os navegadores web;
 - Transforma Dart em JavaScript;

ISW-911 (DM-1)

Dart - Introdução

Bibliotecas:

- Possui um extenso conjunto de bibliotecas, tais como:
 - · dart:core tipos próprios, collections e demais funcionalidades padrões;
 - dart:collection tipos de collections mais estruturadas, como filas, listas ligadas, hashmaps e árvores binárias;
 - dart:convert codificadores e decodificadores entre diferentes representações de dados, incluindo JSON e UTF-8;
 - dart:math funções, constantes matemáticas e geradores de números aleatórios;
 - dart:io manipulação de arquivos, criação de socket, conexão HTTP e outros tipos de I/O;
 - dart:async suporte para programação assíncrona, com classes como Future e Stream;
 - dart:typed_data listas que manipulam eficientemente dados de tamanho fixo (p. ex.: inteiros de 8 bits) e tipos SIMD;
 - dart:async suporte para programação assíncrona, com classes como Future e Stream;
 - dart:ffi interfaces com funções estrangeiras para interop com outros códigos lembram uma interface em C;
 - dart:isolate programação concorrente, usando workers independentes e semelhantes às threads, mas que não compartilham memória, trocando apenas mensagens;
 - dart:html elementos HTML e outros recursos para apps que precisam interagir com o navegador e o DOM (document object model) da página.

Prof. Davi

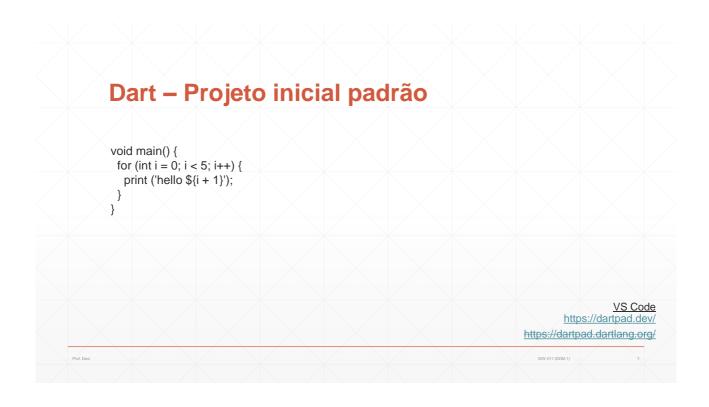
ISW-011 (DDM-

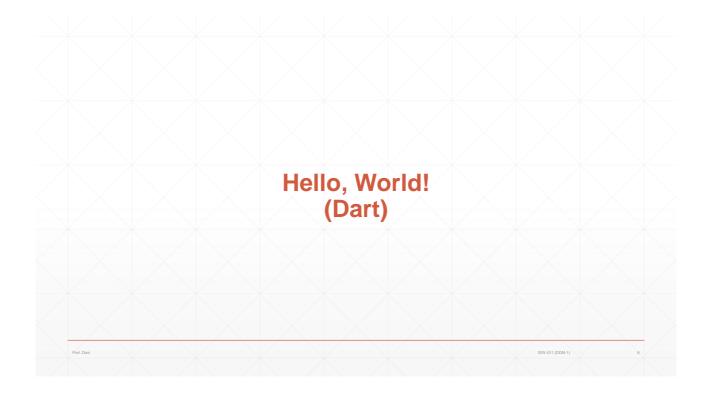
Dart - Introdução

- Pontos importantes:
 - Pode-se utilizar aspas ou apóstrofo ao manipular Strings;
 - É possível concatenar String com variáveis sem usar caracteres especiais;
 - Não é obrigatório especificar o tipo de variáveis, métodos ou parâmetros (ou seja, não é fortemente "tipada");
 - Os operadores lógicos no Dart são: && (AND) e || (OR);

Prof. Davi

ISW-011 (DDM-1)





Primeiro Projeto – "Hello, world"

```
void main() {
  final String msg = "Hello, world!";

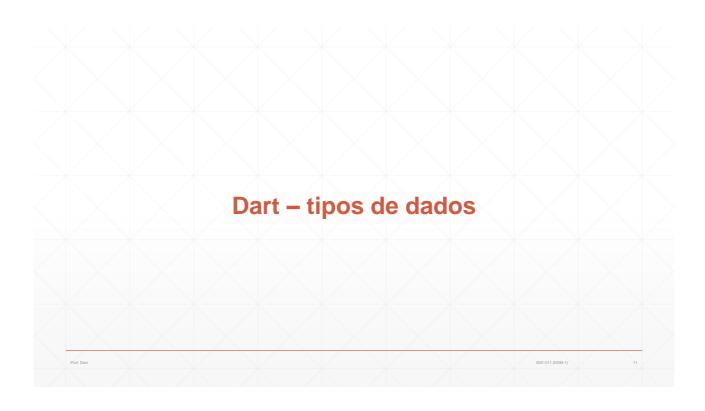
// Comentário em linha
  print("Exibirá a primeira mensagem!!");
  print("Mensagem: $msg");
}
```

https://dartpad.dev/ https://dartpad.dartlang.org/

ISW-011 (DDM-1)

Segundo Projeto – sub-rotina e variáveis

```
void main() {
                                                         if (c > 0){
                                                          print("A soma foi positiva");
final String nome = "Rubens Lara";
                                                         else {
int a = 1;
                                                          print("A soma foi negativa");
 a = 3;
 var b = 2;
                                                         print("Soma: R\$ $c");
int c = somar(a, b);
                                                         print("Nome: $nome");
// Comentário em linha
                                                        somar(a, b) {
                                                         return a + b;
Comentário em bloco
                                                                                             https://dartpad.dev/
                                                                                    https://dartpad.dartlang.org/
```



Dart - Tipos de dados

- Saída:
 - Como já vimos, para exibir uma saída podemos utilizar o comando "print";
 - Ex.:

print("Minha idade é " + idade.toString()); // conversão manual para concatenar print("Minha idade é \$idade"); // exibindo o valor sem concatenar (interpolação) print("Minha idade ano que vem será: \${idade + 1}"); // uso de expressão

VS Code https://dartpad.dev/ https://dartpad.dartlang.org/

ISW-011 (DDM-1)





```
Dart - Classes

void main() {

Pessoa p = Pessoa("Rubens Lara");

int c = Calculadora.somar (7, 8);

print("Soma: $c");
print("Nome: ${p.nome}");
}

class Pessoa {
String nome;

Pessoa (this.nome);
}

class Calculadora {
static somar (a, b) {
return a + b;
}
}

VS Code
https://dartpad.dev/
https://dartpad.dartlang.org/
```

Dart - Classes e Lista void main(){ Carro carro1 = Carro("HB20"); Carro carro2 = Carro("Gol"); Carro carro3 = Carro("Argo"); class Carro { String modelo; final carros = [carro1, carro2, carro3]; // carros.add(carro1); carros.add(carro2); carros.add(carro3); Carro (this.modelo); String toString(){ print("Carros disponíveis: \$carros, tamanho: \${carros.length}"); return modelo; // looping com For Each for (Carro carroTemp in carros){ print("Carro disponível: \${carroTemp.modelo}"); VS Code // looping com convencional for (int i = 0; i < carros.length; i++) { https://dartpad.dev/ https://dartpad.dartlang.org/ print(" Carro: \${carros[i].modelo}");

Dart - Herança e Polimorfismo

```
void main() {
   Carro carro1 = Carro("HB20", 2019);
                                                                                    String dadosVeiculo() {
                                                                                     return modelo;
 Carro carro2 = Carro("Gol", 2008);
 Carro carro3 = Carro("Argo", 2021);
 final carros = [carro1, carro2, carro3];
                                                                                   class Carro extends Veiculo {
 // carros.add(carro1);
                                                                                    Carro(nomeModelo, this.ano): super(nomeModelo):
 // carros.add(carro2)
 // carros.add(carro3);
                                                                                    // usado para imprimir todos os objetos da lista
 print("Lista: $carros, quantidade de carros: ${carros.length}");
                                                                                    @override
 for (Carro carroTemp in carros) {
                                                                                    String toString() {
  //print("Carro: ${carroTemp.modelo} - Ano: ${carroTemp.ano}");
                                                                                     return "$modelo: $ano";
  print(carroTemp.dadosVeiculo());
                                                                                    String dadosVeiculo() {
                                                                                    // return "Modelo: ${super.dadosVeiculo()}; ano fabricação: $ano"; return "Carro: $modelo e Ano: $ano";
class Veiculo {
 String modelo;
 Veiculo(this.modelo);
                                                                                                                                                         VS Code
                                                                                                                                             https://dartpad.dev/
```

Dart - Map

```
void main() {
                                                                       // looping pelo "values" do map
 Carro carro1 = Carro("HB20");
                                                                       print("\nImprimindo a lista com foreach a partir do 'value'");
 Carro carro2 = Carro("Gol");
                                                                       for (Carro carroTemp in carrosMap.values) {
 Carro carro3 = Carro("Argo");
                                                                        print(" >> ${carroTemp.modelo}");
 // map é um par de chave e valor, igual o json
 final carrosMap = {"1": carro1, "2": carro2};
 carrosMap["3"] = carro3;
                                                                      class Carro {
                                                                       String modelo:
 print("Imprimindo a lista toda de uma vez...");
 print("Lista: $carrosMap, length: ${carrosMap.length}");
                                                                       Carro(this.modelo);
 // looping pela "key" do map
                                                                        @override
 print("\nImprimindo a lista com foreach a partir da 'key'");
                                                                       String toString() {
 for (String id in carrosMap.keys) {
                                                                        return modelo;
  final carroTemp = carrosMap[id];
  print(" >> ${carroTemp?.modelo}");
// o "?" acima serve porque "modelo" pode ser nulo
                                                                                                                                VS Code
 }
                                                                                                                    https://dartpad.dev/
```



Dart - Mixin void main() { @override Carro carro1 = Carro("HB20"); print("Modelo do carro: \$carro1"); String toString(){ return modelo; carro1.acelerar(100); carro1.abastecer(40, "gasolina"); abstract class Veiculo { void acelerar(int velocidade); class Carro extends Veiculo with Combustivel { String modelo; Carro(this.modelo); mixin Combustivel { abastecer(int qtde, String tipo) { @override void acelerar(int velocidade) { print("Abastecendo \$qtde litros de \$tipo"); print("Acelerando a \$velocidade km/h"); VS Code https://dartpad.dev/

Dart - Mixin

- Observações:
 - Mixin permite adicionar métodos de outras classes na classe em questão sem precisar utilizar herança;
 - para isso, basta adicionar a palavra "with" na declaração da classe;
 - é possível utilizar o "mixin" para incluir mais do que uma classe na classe em questão: para isso, basta separar as demais com vírgula;
 - é possível também declarar uma classe como abstract: com isso, todos os métodos dessa classe só terão assinatura (serão abstratos) e sua implementação deverá ser feita na classe que implementá-la.

Prof. Davi

Dart - Arrow Function

```
void main() {
   Carro carro1 = Carro("HB20");
   print("Modelo do carro: $carro1");
   carro1.acelerar(100);
   carro1.abastecer(40, "gasolina");
}

class Carro extends Veiculo with Combustivel {
   String modelo;
   Carro(this.modelo);
   @ override
   void acelerar(int velocidade) {
      print("Acelerando a $velocidade km/h");
   }
}
```

```
@ override
String toString() => modelo;
}
abstract class Veiculo {
   void acelerar(int velocidade);
}
mixin Combustivel {
   abastecer(int qtde, String tipo) {
      print("Abastecendo $qtde litros de $tipo");
   }
}
```

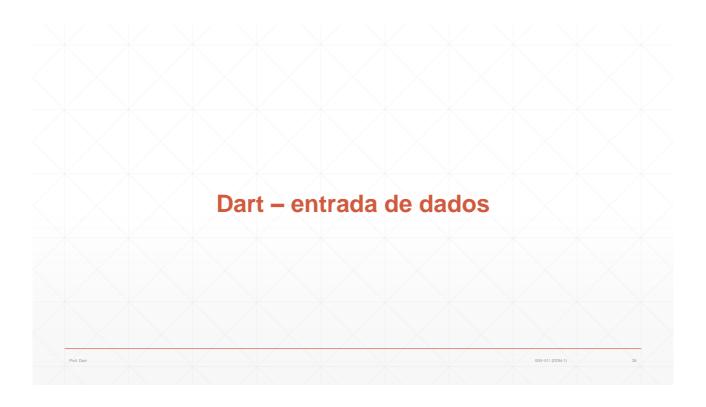
VS Code https://dartpad.dev/

ISW-011 (DDM-1)

Dart - Arrow Function

- Observações:
 - é possível criar funções de forma mais resumida, utilizando apenas uma seta;
 - com isso, não é preciso ter as chaves e nem a palavra "return";
 - é possível apenas para retornos e expressões, não para declarações.

Dart – Exercícios



Dart – Entrada e saída de dados • Entrada: • Deve-se fazer a importação de uma biblioteca específica: "dart.io". • Por isso, somente é possível utilizá-la em uma IDE: não dá para utilizá-la no https://dartpad.dev/!! • Ex.: import 'dart:io'; void main(){ print ("Qual seu nome?"); String nome = stdin.readLineSync(); print ("Olá, \$nome, seja bem-vindo(a)."); VS Code

Dart - Entrada e saída de dados Conversão de tipos (type casting): if (input != null){ import 'dart:io'; int idade = int.parse(input); print ("Então você nasceu em \$(2023 void main(){ idade)."); print ("Qual seu nome?"); } String nome = stdin.readLineSync(); else { print ("Olá, \$nome, seja bem-vindo(a)."); print ("Não foi possível determinar o ano de nascimento."); print ("Qual sua idade?"); String? temp = stdin.readLineSync(); } VS Code

