DART

1. Origem e História

O Dart foi revelado ao mundo em outubro de 2011, durante a conferência GOTO na Dinamarca.

A ideia original era ambiciosa: o Google queria criar um sucessor para o JavaScript. Na época, o JS era visto como uma linguagem com muitos "problemas de design" que dificultavam a criação de aplicações web complexas e de alto desempenho.

Os Criadores

A linguagem não nasceu por acaso; ela foi projetada por "pesos pesados" da computação:

Lars Bak: Um engenheiro de software dinamarquês, famoso por liderar a criação da V8 (o motor de JavaScript super-rápido do Google Chrome).

Kasper Lund: Também vindo do projeto V8 e especialista em máquinas virtuais.

---

2. A Evolução: Do "Quase Esquecimento" ao Topo

No início, o Dart sofreu resistência. Os desenvolvedores web não queriam abandonar o JavaScript, e os navegadores (exceto o Chrome) se recusaram a dar suporte nativo ao Dart.

A Virada de Chave:

Em vez de morrer, o Dart se reinventou. O Google percebeu que a estrutura da linguagem era perfeita para interface de usuário (UI).

a) Dart 1.0 (2013): Focado na web.

b) A Ascensão do Flutter (2017/2018): O Google lançou o Flutter, usando Dart como linguagem principal. Isso mudou tudo.

c) Dart 2.0 (2018): A linguagem foi redefinida com um sistema de tipos fortes e otimizada para o desenvolvimento client-side (mobile e web).

d) Dart 3.0 (2023): Introduziu o Null Safety obrigatório e padrões de registro, tornando-a uma linguagem moderna e extremamente segura.

---

3. Principais Características

O Dart é uma linguagem orientada a objetos, com sintaxe baseada em C (o que a torna muito familiar para quem já mexeu com Java, C# ou JavaScript).

O que torna o Dart especial?

Compilação Híbrida (JIT e AOT): * JIT (Just-in-Time): Usado durante o desenvolvimento. Permite o famoso Hot Reload (você salva o código e a mudança aparece no app em menos de 1 segundo).

AOT (Ahead-of-Time): Usado para a versão final. Transforma o código em linguagem de máquina pura, garantindo performance máxima.

Single-Threaded (Isolates): O Dart roda em uma única thread, o que evita problemas complexos de concorrência (como deadlocks), mas usa "Isolates" para tarefas pesadas em paralelo.

Forte e Estática: Desde o Dart 3, você tem total segurança contra erros de "valor nulo" (Null Safety).

Mãos à Obra

O Dart é uma linguagem **fortemente tipada**, mas também é inteligente o suficiente para entender o que você está escrevendo.

Tipagem Explícita.

Você diz exatamente o que a variável vai guardar:

String: Textos (sempre entre aspas)

int: Números inteiros (1, 10, -5).

double: Números decimais (1.5, 3.14).

bool: Valores lógicos (true ou false).

Trabalhando com variáveis

primeiro.dart

```dart
Run | Debug
void main() {
  String nome = "João da Silva";
  int idade = 10;
  double peso = 80.568;
  bool maiorIdade = idade >= 18;
  print(
    "Seu nome, $nome,sua idade $idade, seu peso ${peso.toStringAsFixed(2)}",
  );
  print(" É maior de idade $maiorIdade");
}
```

```
Seu nome, João da Silva,sua idade 10, seu peso 80.57
 É maior de idade false
```

- **Inferência de Tipo** (var)

O Dart olha para o valor que você colocou e "deduz" o tipo. Se você escrever var idade = 25;, o Dart já sabe que idade é um int.segundo.dart

- **Constantes: final e const**

final: O valor é definido apenas uma vez quando o programa roda (em tempo de execução).

const: O valor deve ser conhecido antes mesmo do programa rodar (em tempo de compilação).

segundo.dart

```dart
Run | Debug
void main() {
  var idade = 20;
  final nome = "João";
  final dataAtual = DateTime.now(); // runtime
  const pi = 3.14;
  print(idade);
  print(nome);
  print(dataAtual);
  print(pi);
}
```

```
20
João
2026-04-03 19:45:20.359350
3.14
```

Entrada de dados.

```dart
import 'dart:io';

Run | Debug
void main() {
  print("--- SISTEMA DE NOTAS ---");


  stdout.write("Digite o nome do aluno: ");
  String nome = stdin.readLineSync()!;

  stdout.write("Digite a nota 1: ");
  double nota1 = double.parse(stdin.readLineSync()!);

  stdout.write("Digite a nota 2: ");
  double nota2 = double.parse(stdin.readLineSync()!);

  stdout.write("Digite a nota 3: ");
  double nota3 = double.parse(stdin.readLineSync()!);

  double media = (nota1 + nota2 + nota3) / 3;

  print("\nO aluno $nome ficou com média ${media.toStringAsFixed(1)}");

  if (media < 5) {
    print("Resultado: REPROVADO!");
  } else if (media < 6 ) {
    print("Resultado: RECUPERAÇÃO");
  }else{
    print("Resultado: APROVADO");
  }
}
```

Operadores Aritméticos

Operador	Operação	Exemplo	Resultado
`+`	Adição	`5 + 2`	`7`
`-`	Subtração	`5 - 2`	`3`
`*`	Multiplicação	`5 * 2`	`10`
`/`	Divisão (Retorna `double`)	`5 / 2`	`2.5`
`~/`	Divisão Inteira	`5 ~/ 2`	`2`
`%`	Módulo (Resto da divisão)	`5 % 2`	`1`

EXERCICIOS.

Atenção salve dentro de uma pasta zip e mande para mim.

1 – Crie um programa em DART, que receba o nome de uma pessoa, seu ano de nascimento, o programa deverá mostrar o nome da pessoa, sua idade e se ela é maior de idade ou menor de idade.

2 – Crie um programa que receba 3 números, mostre o maior número digitado e se a soma dos números e par ou ímpar.

3 – Crie um programa que receba 3 números e mostre se a soma destes números e divisível por 3 e 5 ao mesmo tempo.

Listas em Dart

O que é uma Lista?

Em Dart, Listas são um tipo de coleção ordenada, onde os elementos são armazenados em uma sequência e podem ser acessados por seus índices. Uma lista pode conter qualquer tipo de dado, como números, strings, objetos, entre outros. As listas são mutáveis, ou seja, seus elementos podem ser alterados ao longo do tempo.

Como Declarar uma Lista

Existem várias formas de declarar uma lista em Dart. Aqui estão as mais comuns:

1. Usando a palavra-chave List

```
List<int> numeros = [1, 2, 3, 4, 5];
```

Neste exemplo, números é uma lista de inteiros (int).

2. Usando a Sintaxe Literais

```
var frutas = ['maçã', 'banana', 'laranja'];
```

Neste caso, o Dart pode inferir automaticamente o tipo dos dados.

3. Lista Vazia

```
List<String> nomes = [];
```

Você pode criar uma lista vazia e adicionar elementos a ela mais tarde.

Operações Comuns em Listas

1. Acessar Elementos da Lista

Para acessar um elemento em uma lista, você usa o índice (começando do 0):

```
var primeiraFruta = frutas[0]; // 'maçã'
```

2. Adicionar Elementos

Você pode adicionar novos elementos usando o método add():

```
frutas.add('uva');
```

3. Remover Elementos

Para remover um elemento da lista, utilize o método remove():

```
frutas.remove('banana'); // Remove 'banana'
```

Ou, para remover o elemento de um índice específico:

```
frutas.removeAt(1); // Remove o elemento no índice 1
```

4. Verificar Tamanho da Lista

Você pode verificar o número de elementos em uma lista usando o atributo length:

```
print(frutas.length); // 3 (se a lista tiver 3 elementos)
```

5. Modificar um Elemento

Para modificar um elemento em uma lista, basta atribuir um novo valor ao índice correspondente:

```
frutas[0] = 'morango';
```

6. Iterar sobre uma Lista

Você pode iterar sobre uma lista usando um loop for:

```
for (var fruta in frutas) {
  print(fruta);
}
```

Ou usar o método forEach():

```
frutas.forEach((fruta) {
  print(fruta);
});
```

Métodos Úteis para Listas

1. contains()

Verifica se a lista contém um elemento específico:

```
print(frutas.contains('maçã')); // true ou false
```

2. indexOf()

Retorna o índice de um elemento:

```
print(frutas.indexOf('laranja')); // 1
```

3. sublist()

Retorna uma sublista a partir de um índice inicial até o final:

```
var subLista = frutas.sublist(1, 3); // ['banana', 'laranja']
```

4. sort()

Ordena a lista:

```
frutas.sort(); // Ordena em ordem alfabética
```

5. reversed

Retorna uma nova lista com os elementos invertidos:

```
var listaInvertida = frutas.reversed.toList();
```

Listas com Tipos Diferentes

Dart também permite que você tenha listas com tipos diferentes de dados (listas heterogêneas). Contudo, é uma prática recomendada utilizar listas homogêneas sempre que possível para manter a consistência do código.

Exemplo de uma lista com tipos diferentes:

```
List<dynamic> dados = ['Alice', 25, true];
```

6. Listas Imutáveis

Se você deseja criar uma lista imutável, use a função const:

```
var imutavel = const [1, 2, 3];
```

Uma lista constante não pode ser alterada após sua criação.