

DDM-II

Aula 2: Imersão em Dart

Prof. Anderson Vanin

2026

Semana 2: Imersão em Dart

1. Introdução: O que é Dart e por que estamos aqui?

O Flutter é apenas um framework (um conjunto de ferramentas visuais). A "mágica" acontece na linguagem de programação chamada **Dart**, criada pelo Google.

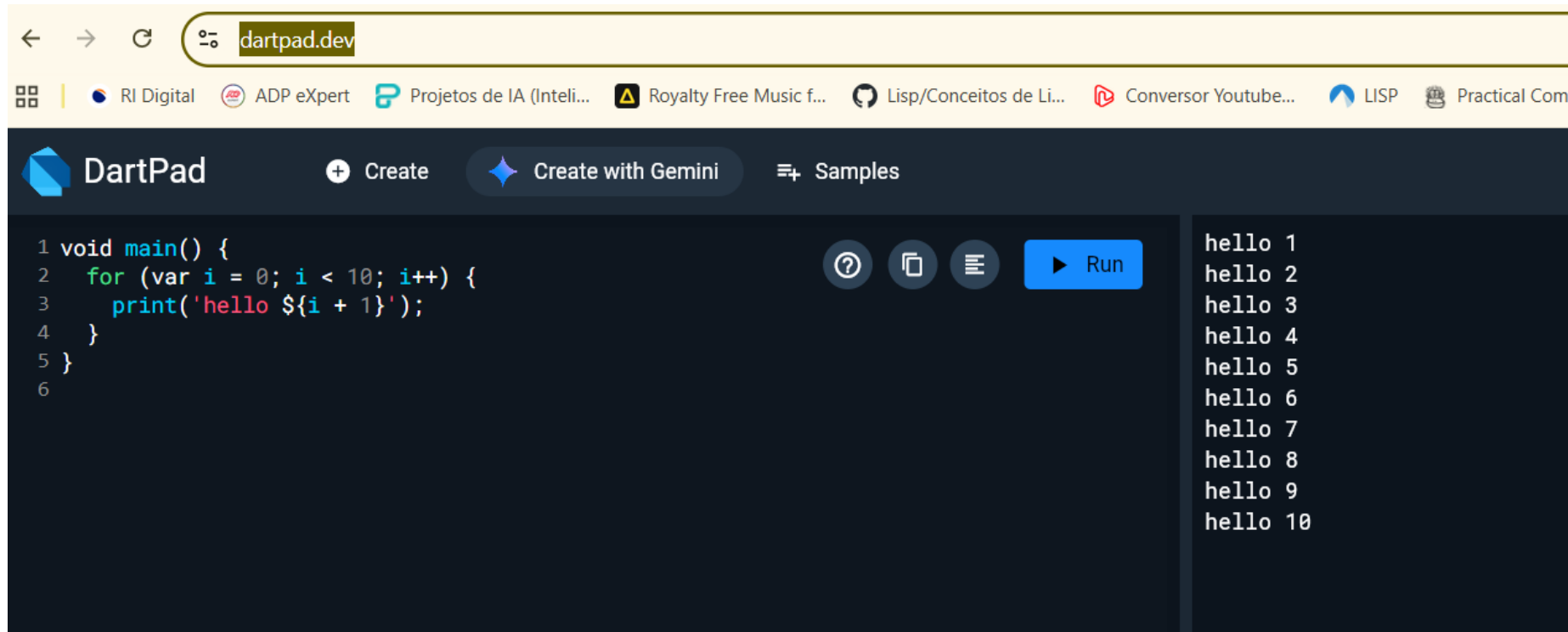
Características Chave do Dart:

1. **Otimizada para UI:** Foi desenhada para criar interfaces de usuário de forma fácil.
2. **Dois Modos de Compilação (O Segredo do Flutter):**
 - **JIT (Just-In-Time):** Durante o desenvolvimento, o código é compilado "na hora". Isso permite o Hot Reload (mudar a cor de um botão e ver o resultado em milissegundos sem reiniciar o app).
 - **AOT (Ahead-Of-Time):** Quando o app vai para a loja, o código é compilado para binário nativo (código de máquina ARM). Isso garante a performance de 60 FPS.

1. Introdução: O que é Dart e por que estamos aqui?

Para programar em Dart de forma mais fácil e sem a necessidade de um ambiente Flutter configurado é possível usar o site DartPad. O DartPad possui código aberto e pode ser executado em qualquer navegador atual.

<https://dartpad.dev/>



The screenshot shows the DartPad web interface. The browser's address bar displays 'dartpad.dev'. The page header includes the DartPad logo and navigation links: 'Create', 'Create with Gemini', and 'Samples'. The main area features a code editor with the following Dart code:

```
1 void main() {  
2   for (var i = 0; i < 10; i++) {  
3     print('hello ${i + 1}');  
4   }  
5 }  
6
```

To the right of the code editor are icons for help, copy, and a 'Run' button. The output panel on the right side of the interface displays the result of the program execution:

```
hello 1  
hello 2  
hello 3  
hello 4  
hello 5  
hello 6  
hello 7  
hello 8  
hello 9  
hello 10
```

2. Variáveis e Tipagem

O Dart é uma linguagem **Fortemente Tipada** (Strongly Typed). Isso significa que, se uma variável nasce número, ela morre número. Porém, o Dart é inteligente e possui **Inferência de Tipo**.

A. Declaração de Variáveis

1. **Tipagem Explícita (Modo Verboso):** Você diz exatamente o que a variável é. Recomendado para atributos de classes ou APIs públicas.

```
String nome = "João";
```

```
int idade = 25;
```

```
double altura = 1.75;
```

```
bool estuda = true;
```

2. Variáveis e Tipagem

- 2. **Inferência de Tipo (Modo Moderno - var):** Você deixa o Dart adivinhar pelo valor inicial. É o padrão da comunidade Flutter para variáveis locais.

```
var cidade = "São Paulo";
```

```
// O Dart sabe que isso é String
```

```
// cidade = 10;
```

```
// ERRO! Você não pode colocar int numa variável que nasceu String.
```

2. Variáveis e Tipagem

B. Tipagem Dinâmica (`dynamic`)

Existe um tipo especial chamado `dynamic`. Ele desliga a verificação de tipos.

```
dynamic coringa = "Texto";
```

```
coringa = 123; // Funciona!
```

```
coringa = true; // Funciona!
```

- **Perigo:** Use com extrema moderação (geralmente ao consumir APIs JSON mal formatadas). O uso excessivo de *dynamic* mata a segurança e a performance do app.

2. Variáveis e Tipagem

c. Constantes (final vs const)

- **final**: O valor é definido uma única vez, mas pode ser definido em tempo de execução (ex: a hora que o app abriu).
- **const**: O valor deve ser conhecido antes de compilar (ex: o valor de $\text{Pi} = 3.14$).

3. Estruturas de Controle (Decisões)

A lógica condicional no Dart é muito similar ao C, Java e JavaScript (família C-Style).

A. If e Else

```
var nota = 7.5;
```

```
if (nota >= 9.0) {  
    print("Excelente");  
} else if (nota >= 7.0) {  
    print("Aprovado");  
} else {  
    print("Reprovado");  
}
```

```
1 void main() {  
2   var nota = 7.5;  
3  
4   if (nota >= 9.0) {  
5     print("Excelente");  
6   } else if (nota >= 7.0) {  
7     print("Aprovado");  
8   } else {  
9     print("Reprovado");  
10  }  
11 }
```

Run

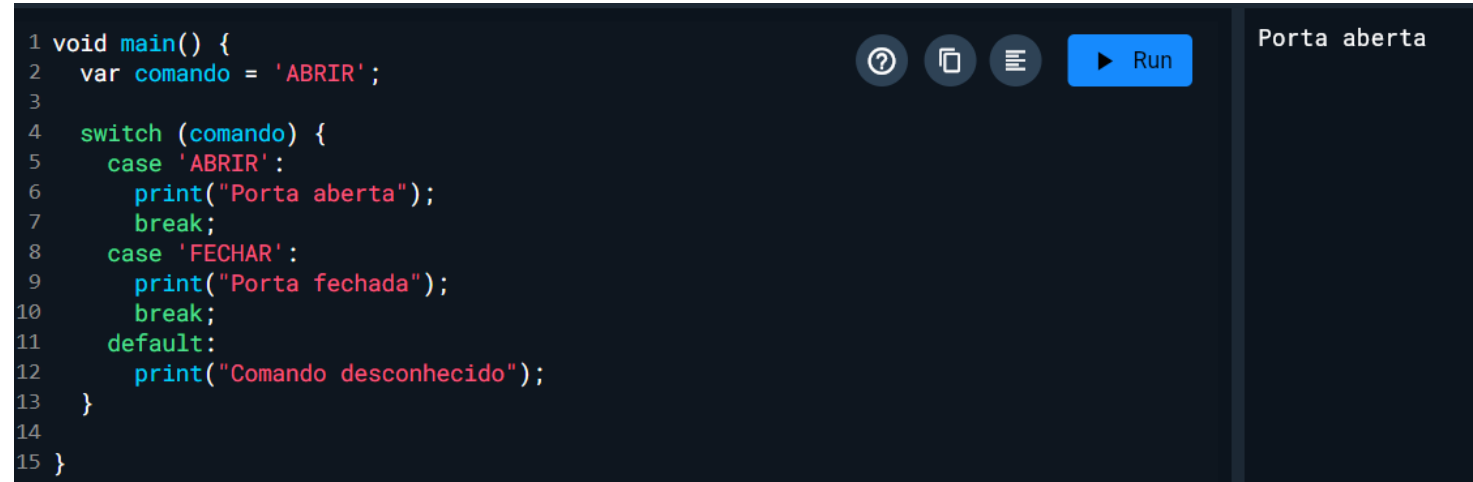
Aprovado

3. Estruturas de Controle (Decisões)

B. Switch Case

```
var comando = 'ABRIR';
```

```
switch (comando) {  
    case 'ABRIR':  
        print("Porta aberta");  
        break;  
    case 'FECHAR':  
        print("Porta fechada");  
        break;  
    default:  
        print("Comando desconhecido");  
}
```



```
1 void main() {  
2     var comando = 'ABRIR';  
3  
4     switch (comando) {  
5         case 'ABRIR':  
6             print("Porta aberta");  
7             break;  
8         case 'FECHAR':  
9             print("Porta fechada");  
10            break;  
11            default:  
12                print("Comando desconhecido");  
13        }  
14    }  
15 }
```

Porta aberta

4. Estruturas de Repetição (Loops)

A. For (Padrão)

```
for (int i = 0; i < 5; i++) {  
    print("Contagem: $i");  
}
```

```
1 void main() {  
2     for (int i = 0; i < 5; i++) {  
3         print("Contagem: $i");  
4     }  
5 }
```



```
Contagem: 0  
Contagem: 1  
Contagem: 2  
Contagem: 3  
Contagem: 4
```

B. While / Do-While

```
int bateria = 0;  
  
while (bateria < 100) {  
    print("Carregando... $bateria%");  
    bateria += 20;  
}
```

```
1 void main() {  
2     int bateria = 0;  
3     while (bateria < 100) {  
4         print("Carregando... $bateria%");  
5         bateria += 20;  
6     }  
7 }
```



```
Carregando... 0%  
Carregando... 20%  
Carregando... 40%  
Carregando... 60%  
Carregando... 80%
```

5. Entrada e Saída no Console (Para a Prática)

Como ainda não temos interface gráfica (botões e telas), usaremos o terminal. Para isso, precisamos importar a biblioteca de Input/Output (`dart:io`).

- **Saída:** `print("Mensagem");`
- **Entrada:** `stdin.readLineSync();` (Lê o que o usuário digitou e deu Enter).

LIMITAÇÕES DO DARTPAD

Não é permitido acesso a bibliotecas de sistema como por exemplo entrada de dados via teclado!



The screenshot shows the DartPad web interface. On the left, a code editor contains the following Dart code:

```
1 import 'dart:io'; // Biblioteca para ler do teclado
2
3 void main() {
4   print("--- Calculadora de IMC ---");
5
6   // 1. Entrada de Dados
7   print("Digite seu peso (kg):");
8   // O readLineSync retorna uma String? (pode ser nulo), por isso o '!'
9   String? inputPeso = stdin.readLineSync();
```

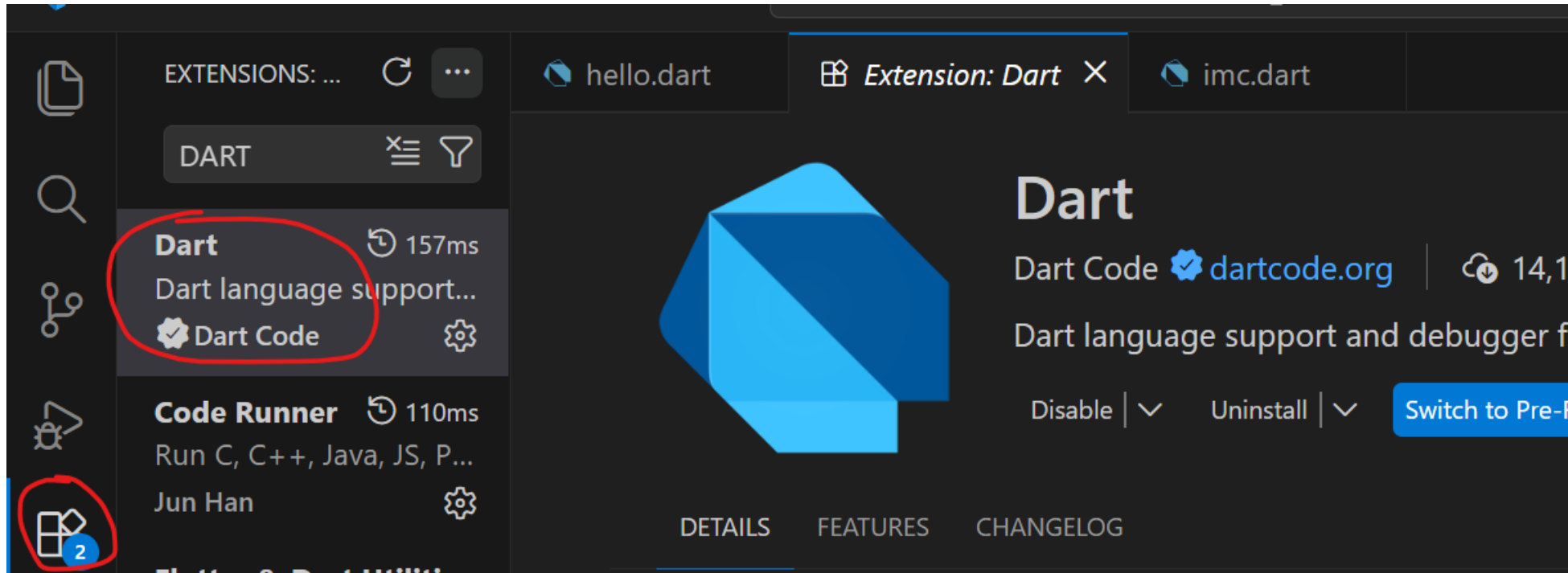
The line `import 'dart:io';` is circled in red. To the right of the code editor is a 'Run' button. On the right side of the interface, an error message is displayed:

```
Uncaught Error, error: Error: Unsupported operation: StdIOUtils._getStdioInputStream
```

A red arrow points from the error message back to the `dart:io` import in the code.

Solução: rodar localmente (VS CODE)

No VS CODE



VERIFICAR SE O SDK DART ESTÁ INSTALADO

```
C:\Users\Anderson>flutter doctor
```

```
A new version of Flutter is available!
```

```
To update to the latest version, run "flutter upgrade".
```

```
Doctor summary (to see all details, run flutter doctor -v):
```

```
[✓] Flutter (Channel stable, 3.38.4, on Microsoft Windows [versão 10.0.26200.7840], locale pt-BR)
[✓] Windows Version (11 Education 64-bit, 25H2, 2009)
[✓] Android toolchain - develop for Android devices (Android SDK version 36.1.0)
[✓] Chrome - develop for the web
[✓] Visual Studio - develop Windows apps (Ferramentas de Build do Visual Studio 2019 16.11.26)
[✓] Connected device (3 available)
[✓] Network resources
```

```
• No issues found!
```

```
C:\Users\Anderson>dart --version
```

```
Dart SDK version: 3.10.3 (stable) (Tue Dec 2 01:04:53 2025 -0800) on "windows_x64"
```

```
C:\Users\Anderson>
```

Roteiro Prático: Calculadora de IMC (Terminal)

Esta atividade consolida variáveis, conversão de tipos e if/else.

Passo 1: Configurar o arquivo Crie um arquivo chamado `imc.dart`.

Passo 2: O Código Comentado

```
import 'dart:io'; // Biblioteca para ler do teclado

void main() {
  print("--- Calculadora de IMC ---");

  // 1. Entrada de Dados
  print("Digite seu peso (kg):");
  // O readLineSync retorna uma String? (pode ser
  nulo), por isso o '!'
  String? inputPeso = stdin.readLineSync();

  print("Digite sua altura (m):");
  String? inputAltura = stdin.readLineSync();

  // 2. Conversão e Validação (Tratamento básico)
  if (inputPeso != null && inputAltura != null) {
    // Convertendo String para double
    double peso = double.parse(inputPeso);
    double altura = double.parse(inputAltura);

    // 3. Processamento
    double imc = peso / (altura * altura);

    // 4. Saída formatada
    print("Seu IMC é: ${imc.toStringAsFixed(2)}"); //
    Limita a 2 casas decimais

    // 5. Lógica de Decisão
    if (imc < 18.5) {
      print("Classificação: Abaixo do peso");
    } else if (imc < 24.9) {
      print("Classificação: Peso normal");
    } else if (imc < 29.9) {
      print("Classificação: Sobrepeso");
    } else {
      print("Classificação: Obesidade");
    }
  } else {
    print("Erro: Dados inválidos.");
  }
}
```


FileEditSelectionViewGoRunTerminalHelp

EXPLORER

...

▼ AULA_...

hello.dart

imc.dart

imc.dart > main

1 import 'dart:io'; // Biblioteca para ler do teclado

2

Run | Debug

3 void main() {

4 print("--- Calculadora de IMC ---");

5

6 // 1. Entrada de Dados

7 print("Digite seu peso (kg):");

8 // O readLineSync retorna uma String? (pode ser nulo), por isso o '!'

9 String? inputPeso = stdin.readLineSync();

10

11 print("Digite sua altura (m):");

12 String? inputAltura = stdin.readLineSync();

13

14 // 2. Conversão e Validação (Tratamento básico)

15 if (inputPeso != null && inputAltura != null) {

16 // Convertendo String para double

17 double peso = double.parse(inputPeso);

18 double altura = double.parse(inputAltura);

19

20 // 3. Processamento

21 double imc = peso / (altura * altura);

22

23 // 4. Saída formatada

24 print("Seu IMC é: \${imc.toStringAsFixed(2)}"); // Limita a 2 casas decimais

25

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

PS C:\Users\Anderson\Desktop\ETEC 2026\DDM II\aulas\aula_02> dart imc.dart

--- Calculadora de IMC ---

Digite seu peso (kg):

80

Digite sua altura (m):

1.85

Seu IMC é: 23.37

Classificação: Peso normal

PS C:\Users\Anderson\Desktop\ETEC 2026\DDM II\aulas\aula_02>

> OUTLINE

> TIMELINE

> DEPENDENCIES

Desafio para os Alunos

1. O código acima quebra se o usuário digitar letras ou usar vírgula em vez de ponto (ex: "80,5").
2. Usem `try/catch` ou substituição de string (`replaceAll`) para corrigir o problema da vírgula.