**Apostila de Funções em Dart/Flutter**

Este material didático aborda os principais conceitos sobre funções na linguagem de programação Dart, comumente utilizada com o framework Flutter. Aqui você encontrará exemplos práticos para cada tipo de função, desde a mais simples até conceitos mais avançados como *higher-order functions*.

**1. Função Simples (Sem Parâmetros, Sem Retorno)**

**Objetivo:** Criar uma função que executa uma ação pré-definida sem a necessidade de receber dados externos e sem retornar nenhum valor.

**Exercício:** Crie uma função chamada exibirBoasVindas que não recebe nenhum parâmetro e, quando chamada, simplesmente imprime a mensagem "Bem-vindo ao mundo do Flutter!" no console.

**Exemplo de Código:**

// Declaração da função 'exibirBoasVindas'

void exibirBoasVindas() {

  print("Bem-vindo ao mundo do Flutter!");

}

void main() {

  // Chamada da função

  exibirBoasVindas();

}

**Explicação:** A palavra-chave *void* indica que a função não retorna nenhum valor. Como não há nada entre os parênteses *()*, a função não aceita parâmetros. A ação de imprimir no console é executada toda vez que   *exibirBoasVindas()* é chamada.

**2. Função com Parâmetros (Sem Retorno)**

**Objetivo:** Escrever uma função que recebe dados (parâmetros) para processar, mas não retorna um valor.

**Exercício:** Escreva uma função chamada *saudacaoPersonalizada* que aceita um parâmetro do tipo *String* chamado *nome*. A função deve imprimir no console a mensagem "Olá, [nome]! Tenha um ótimo dia.".

**Exemplo de Código:**

// Função que aceita um parâmetro 'nome' do tipo String

void saudacaoPersonalizada(String nome) {

  print("Olá, $nome! Tenha um ótimo dia.");

}

void main() {

  // Chamando a função e passando o argumento "Ana"

  saudacaoPersonalizada("Ana");

}

**Explicação:** A função *saudacaoPersonalizada* tem um parâmetro *String nome* definido em sua assinatura. Ao chamá-la, você deve fornecer um argumento (um valor do tipo

*String*, como "Ana"), que será utilizado dentro da função. A interpolação de string *"$nome"* insere o valor da variável diretamente no texto.

**3. Função com Retorno (Sem Parâmetros)**

**Objetivo:** Criar uma função que, ao ser executada, retorna um valor que pode ser armazenado em uma variável ou usado em outras operações.

**Exercício:** Crie uma função chamada *obterAnoAtual* que não precisa de parâmetros, mas retorna o ano atual como um valor *int*. Chame a função e armazene o resultado em uma variável, depois imprima essa variável.

**Exemplo de Código:**

// Função que retorna um valor do tipo int

int obterAnoAtual() {

  // Em uma aplicação real, você poderia usar uma biblioteca de data/hora.

  // Para este exemplo, vamos retornar um valor fixo.

  return 2025;

}

void main() {

  // A variável 'ano' recebe o valor retornado pela função

  int ano = obterAnoAtual();

  print("O ano atual é: $ano");

}

**Explicação:** O tipo de retorno (*int*) é especificado antes do nome da função.

A palavra-chave   *return* é usada para enviar o valor de volta para onde a função foi chamada.

**4. Função com Parâmetros e Retorno**

**Objetivo:** Desenvolver uma função que recebe dados, realiza um processamento com eles e retorna um resultado.

**Exercício:** Desenvolva uma função chamada *somarDoisNumeros* que recebe dois parâmetros do tipo *double*, *a* e *b*.

A função deve retornar a soma desses dois números.

**Exemplo de Código:**

// Função que recebe dois doubles e retorna a soma

double somarDoisNumeros(double a, double b) {

  return a + b;

}

void main() {

  double resultado = somarDoisNumeros(10.5, 22.3);

  print("O resultado da soma é: $resultado");

}

**Explicação:** Esta é a forma de função mais completa e comum. Ela combina a recepção de parâmetros ( *double a*, *double b*) com o retorno de um valor (*double*). O resultado da operação *a + b* é retornado.

**5. Função de Seta (Arrow Function)**

**Objetivo:** Utilizar uma sintaxe mais concisa para funções que contêm apenas uma única expressão.

**Exercício:** Crie uma função *multiplicar* que aceita dois inteiros e retorna sua multiplicação, tudo em uma única linha usando a sintaxe *=>*.

**Exemplo de Código:**

// Arrow function para multiplicar dois números

[cite\_start]int multiplicar(int a, int b) => a \* b; [cite: 49]

void main() {

  int resultado = multiplicar(5, 10);

  print("O resultado da multiplicação é: $resultado");

}

**Explicação:** A sintaxe *=> expressao* é um atalho para *{ return expressao;}*

É ideal para funções simples e de uma única linha, tornando o código mais limpo e legível.