Tutorial 2 – Fundamentos do Dart

# Sumário:

• Variáveis e Tipos de Dados  
• Controle de Fluxo (if, loops)  
• Funções e Métodos  
• Classes e Objetos

# 1. Variáveis e Tipos de Dados

Int: O tipo `int` representa números inteiros. Ele abrange valores positivos e negativos sem parte decimal.  
int numero = 42;  
  
Double: O tipo `double` é usado para números de ponto flutuante, ou seja, números que possuem parte decimal. Ele é útil para lidar com cálculos que envolvem precisão decimal.  
double altura = 1.75;  
  
Bool: O tipo `bool` representa valores booleanos, ou seja, `true` (verdadeiro) ou `false` (falso). Ele é essencial para a criação de estruturas de controle condicional.  
bool estaChovendo = false;  
  
String: O tipo `String` é usado para armazenar sequências de caracteres. Isso é fundamental para lidar com texto e dados baseados em caracteres.  
String saudacao = 'Olá, mundo!';  
  
Além disso, as variáveis podem ser declaradas usando var, final, ou const, dependendo do uso pretendido:  
var: Permite que o Dart infira o tipo da variável automaticamente. Uma vez atribuída, o tipo não pode ser alterado.  
final: Declara uma variável que pode ser atribuída apenas uma vez, mas cujo valor é determinado em tempo de execução.  
const: Declara uma variável que deve ser uma constante em tempo de compilação, ou seja, o valor é conhecido e atribuído no momento da compilação.  
  
Exemplo prático:  
void main() {  
 var idade = 25; // O Dart infere o tipo como int  
 final nome = 'João'; // Valor atribuído uma única vez  
 const pi = 3.1415; // Valor constante  
  
 print('Idade: $idade, Nome: $nome, Pi: $pi');  
}

# 2. Controle de Fluxo (if, loops)

O controle de fluxo em Dart permite que você execute blocos de código de forma condicional ou repetitiva, dependendo de certas condições.   
Os principais mecanismos de controle de fluxo incluem estruturas condicionais (if, else if, else) e estruturas de repetição (for, while, do-while).  
  
Condicionais  
if, else if, else: São usados para tomar decisões baseadas em uma condição booleana. Se a condição for verdadeira, o código dentro do bloco será executado.  
  
Exemplo prático:  
void main() {  
 int idade = 18;  
  
 if (idade >= 18) {  
 print('Você é maior de idade.');  
 } else if (idade >= 13) {  
 print('Você é adolescente.');  
 } else {  
 print('Você é criança.');  
 }  
}  
  
switch: O switch é usado para testar múltiplas condições baseadas em um valor.  
  
Exemplo prático:  
void main() {  
 String cor = 'vermelho';  
  
 switch (cor) {  
 case 'vermelho':  
 print('Cor quente.');  
 break;  
 case 'azul':  
 print('Cor fria.');  
 break;  
 default:  
 print('Cor não reconhecida.');  
 }  
}  
  
Loops  
for: Executa um bloco de código um número específico de vezes.  
Exemplo prático:  
void main() {  
 for (int i = 0; i < 5; i++) {  
 print('Contagem: $i');  
 }  
}  
  
for-in: Usado para percorrer todos os elementos de uma coleção.  
Exemplo prático:  
void main() {  
 List<String> frutas = ['Maçã', 'Banana', 'Laranja'];  
  
 for (var fruta in frutas) {  
 print(fruta);  
 }  
}  
  
while: Executa o bloco de código enquanto a condição for verdadeira.  
Exemplo prático:  
void main() {  
 int contador = 0;  
  
 while (contador < 5) {  
 print('Contador: $contador');  
 contador++;  
 }  
}  
  
do-while: Executa o bloco de código pelo menos uma vez.  
Exemplo prático:  
void main() {  
 int contador = 0;  
  
 do {  
 print('Contador: $contador');  
 contador++;  
 } while (contador < 5);  
}

# 3. Funções e Métodos

No Dart, funções e métodos são blocos de código reutilizáveis que realizam uma tarefa específica. As funções podem ser definidas de maneira independente, enquanto os métodos são funções associadas a uma classe ou objeto.  
  
Exemplo prático:  
void saudacao() {  
 print('Olá, mundo!');  
}  
  
int soma(int a, int b) {  
 return a + b;  
}  
  
void main() {  
 saudacao(); // Chamada da função  
 int resultado = soma(5, 3);   
 print('Resultado da soma: $resultado');  
}  
  
Parâmetros Opcionais  
Dart oferece suporte a parâmetros opcionais em funções, que podem ser posicionais ou nomeados.  
  
Parâmetros Opcionais Posicionais:  
Exemplo prático:  
void saudacaoOpcional(String nome, [String? saudacao]) {  
 print('${saudacao ?? 'Olá'}, $nome!');  
}  
  
Parâmetros Nomeados:  
Exemplo prático:  
void mostrarDados({required String nome, int idade = 18}) {  
 print('Nome: $nome, Idade: $idade');  
}  
  
void main() {  
 mostrarDados(nome: 'Vinícius');  
 mostrarDados(nome: 'Vinícius', idade: 20);  
}

# 4. Classes e Objetos

Em Dart, classes são usadas como moldes para criar objetos. As classes permitem organizar e encapsular dados e comportamentos relacionados em um único lugar.  
  
Exemplo prático:  
class Carro {  
 String modelo;  
 int ano;  
  
 Carro(this.modelo, this.ano);  
  
 void mostrarDetalhes() {  
 print('Modelo: $modelo, Ano: $ano');  
 }  
}  
  
void main() {  
 Carro meuCarro = Carro('Fusca', 1972);  
 meuCarro.mostrarDetalhes();  
}  
  
Encapsulamento  
Em Dart, os atributos e métodos de uma classe podem ser controlados com encapsulamento usando o prefixo underscore (\_).  
  
Exemplo prático:  
class ContaBancaria {  
 String \_titular;  
 double \_saldo;  
  
 ContaBancaria(this.\_titular, this.\_saldo);  
  
 double get saldo => \_saldo;  
  
 void depositar(double valor) {  
 \_saldo += valor;  
 }  
}  
  
void main() {  
 ContaBancaria conta = ContaBancaria('Vinícius', 1000.0);  
 conta.depositar(500.0);  
}