

Sintaxe da Linguagem

Curso Flutter de Verão



- Explorar as estruturas de repetição em Dart (for, while, do-while e métodos de coleção).
- Apresentar a programação funcional em Dart (funções e lambdas).
- Realizar exercícios para aplicar os conceitos aprendidos na sintaxe da linguagem Dart.

Objetivo da aula de hoje



Revisão da aula anterior

Estruturas de repetição



As estruturas de repetição, também conhecidas como laços ou loops, são recursos utilizados na programação para executar repetidamente um bloco de código enquanto uma determinada condição for verdadeira ou por um número específico de vezes.

Exemplos:

- for
- while
- do-while

for

O laço "for" é usado para iterar sobre uma sequência de valores, como uma lista ou um intervalo numérico. Ele possui três partes: a inicialização (definir o valor inicial), a condição (verificar se o laço deve continuar) e a atualização (alterar o valor de controle após cada iteração).

Essa estrutura de repetição é amplamente utilizada para percorrer listas, executar tarefas iterativas e controlar o fluxo de execução de um bloco de código. O "for" oferece flexibilidade e controle para lidar com tarefas repetitivas de maneira eficiente.

Exemplo

```
void main() {  
    List listOfNumbers = [10, 20, 30, 40, 50, 60, 70, 80];  
    for (var i = 0; i < listOfNumbers.length; i++) {  
        print('Número: ${listOfNumbers[i]}');  
    }  
  
    // Maneira alternativa  
    for (int number in listOfNumbers) {  
        print('Número: $number');  
    }  
}
```

Exemplo

```
void main() {  
    List<double> vendasMensais = [1500.0, 2000.0, 1800.0, 2500.0, 3000.0];  
    double receitaTotal = 0.0;  
  
    for (int i = 0; i < vendasMensais.length; i++) {  
        receitaTotal += vendasMensais[i];  
    }  
  
    double mediaVendas = receitaTotal / vendasMensais.length;  
  
    print("Receita Total: \${receitaTotal.toStringAsFixed(2)}");  
    print("Média de Vendas por Mês: \${mediaVendas.toStringAsFixed(2)}");  
}
```

Exemplo


```
ListView(  
  children: [  
    for (int i = 1; i <= 5; i++)  
      ListTile(  
        title: Text('Item $i'),  
      ),  
  ],  
)
```


while

O laço "while" executa um bloco de código repetidamente enquanto uma condição específica for verdadeira. Antes de cada iteração, a condição é verificada. Se a condição for verdadeira, o bloco de código é executado; caso contrário, o laço é encerrado.

Os "while" loops oferecem um alto nível de flexibilidade e controle para executar repetidamente um bloco de código com base em uma condição específica. Eles são úteis em várias situações e podem ser combinados com outras estruturas de controle para criar lógica mais complexa e personalizada em seus programas.

Exemplo



```
while (isOpen) {  
    if (!isOutOfStock) checkForNewOrder();  
}
```

Exemplo

```
import 'dart:io';

void main() {
  int totalVotos = 0;
  int limiteVotos = 10;

  while (totalVotos < limiteVotos) {
    print('Digite o número da opção que você deseja votar: ');
    int opcao = int.parse(stdin.readLineSync());

    // Lógica para registrar o voto
    // ...

    totalVotos++;
  }

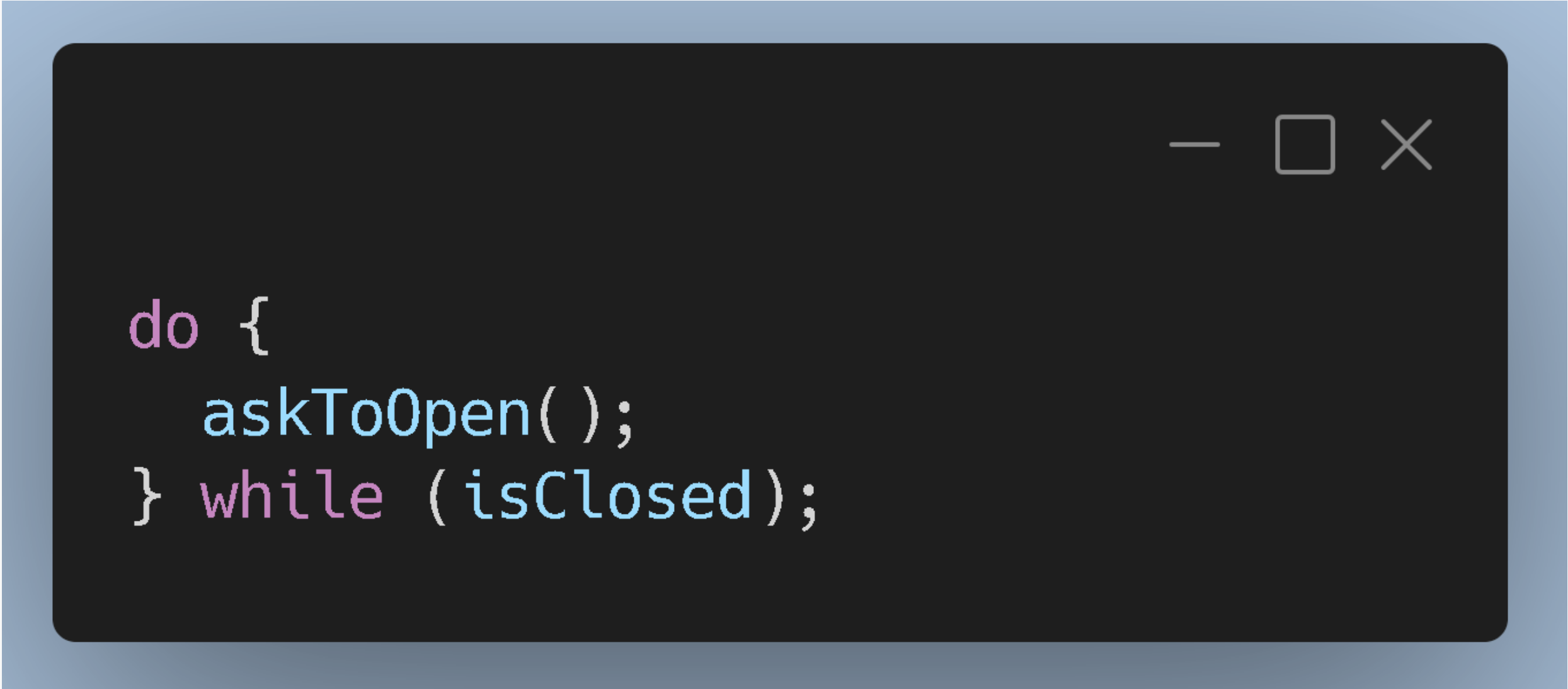
  print('Limite de votos atingido. A votação foi encerrada.');
```

do-while

O laço "do-while" é semelhante ao "while", mas a condição é verificada após a execução do bloco de código. Isso garante que o bloco seja executado pelo menos uma vez, mesmo que a condição seja falsa desde o início.

O "do-while" é útil quando você deseja garantir que um bloco de código seja executado pelo menos uma vez, independentemente da condição. Ele é especialmente útil em situações em que você precisa executar uma ação inicialmente e, em seguida, repetir essa ação com base em uma condição específica.

Exemplo



```
do {  
    askToOpen();  
} while (isClosed);
```

Exemplo

```
import 'dart:io';

void main() {
  bool jogarNovamente = false;

  do {
    // Lógica do jogo ...

    print('Deseja jogar novamente? (S/N)');
    String resposta = stdin.readLineSync();

    if (resposta.toLowerCase() == 's') {
      jogarNovamente = true;
    } else {
      jogarNovamente = false;
    }

  } while (jogarNovamente);

  print('Obrigado por jogar! Até logo.');
}
```

Repetição com métodos de coleções

Os métodos de coleção em Dart são um conjunto de funções disponíveis nas coleções (listas, conjuntos e mapas) que permitem manipular e transformar seus elementos de maneira mais concisa e eficiente.

Esses métodos de coleção podem ser usados para substituir loops tradicionais, como o "for" e o "while", para realizar operações repetitivas em coleções. Isso ajuda a tornar o código mais conciso e expressivo.

Exemplo

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

// forEach()
numeros.forEach((numero) {
    print(numero);
});

// map()
List<int> quadrados = numeros.map((numero) => numero *
numero).toList();

// where()
List<int> numerosPares = numeros.where((numero) =>
numero % 2 == 0).toList();
```


Exemplo

```
List<String> nomes = ["Mario", "Luigi", "Bowser", "Princesa  
Peach"];

// map()
Column(
    children: nomes.map((nome) => Text(nome)).toList(),
)

// where() + map()
Column(
    children: nomes
        .where((nome) => nome == 'Mario')
        .map((nome) => Text(nome))
        .toList(),
)
```

Exercício

01 Estrutura de Repetição – for

Desenvolva um programa em Dart para simular uma contagem regressiva em um foguete espacial prestes a ser lançado. Utilize a estrutura de repetição for para exibir os números de 10 a 1, representando a sequência de contagem regressiva. Cada número deve ser exibido em uma linha separada, criando uma atmosfera emocionante e ansiosa antes do lançamento.



Exercício

02 Estrutura de Repetição – while

Vamos chamar esse jogo de "Adivinhe o número".

O jogador deve adivinhar um número secreto gerado aleatoriamente pelo computador. O jogo informará se o número fornecido pelo jogador é maior, menor ou igual ao número secreto. O jogo continuará até que o jogador adivinhe corretamente o número.

Regras:

1. O número secreto será gerado aleatoriamente entre 1 e 10.
2. O jogador terá um número limitado de tentativas para adivinhar o número secreto.
3. O jogo fornecerá pistas ao jogador indicando se o número é maior, menor ou igual ao número secreto.



Exercício

03 Estrutura de Repetição – coleções

Desenvolva um programa em Dart para gerar uma lista de preços de produtos. Solicite ao usuário o preço máximo e crie uma lista utilizando métodos de coleção com todos os preços pares de 0 até o preço máximo digitado. Essa lista representa os preços disponíveis dos produtos para venda.



Programação funcional

A programação funcional trata a computação como uma avaliação de funções matemáticas puras, evitando a mudança de estado e dados mutáveis. Em Dart, você pode usar os conceitos da programação funcional para escrever código mais conciso, legível e modular.

Ao adotar os princípios da programação funcional em Dart, você pode escrever código mais declarativo, modular e fácil de testar. A linguagem oferece suporte a muitos conceitos e recursos da programação funcional, permitindo que você aproveite seus benefícios em seu código.

Usando funções

As funções em Dart são usadas para agrupar lógica reutilizável, podendo receber parâmetros e retornar valores. Elas podem ser atribuídas a variáveis e passadas como argumentos para outras funções.

Em Dart, todas as funções retornam um valor por padrão e o tipo de retorno deve ser especificado. O uso do tipo void é recomendado quando não há retorno esperado. No entanto, quando se espera um valor de retorno, o tipo de dado apropriado deve ser especificado, e a instrução return é usada para passar o valor de retorno.

Exemplo

```
void orderEspresso(int howManyCups) {  
    print('Cups #: $howManyCups');  
}  
  
orderEspresso(3);  
  
bool moreThan3Spressos(int howManyCups) {  
    return howManyCups > 3 ? true : false;  
}  
  
print(moreThan3Spressos(4)); // true
```

Exemplo

```
void saveData(Data data) {  
    SomeService.save(data);  
}
```

```
MaterialButton(  
    onPressed: () {  
        if (dados != null) {  
            enviarDados(dados);  
        }  
    }  
    child: Text("Enviar")  
)
```


Funções lambdas

Funções lambdas, também conhecidas como funções anônimas ou expressões lambda, são funções que não possuem um nome definido e são definidas de forma concisa e inline. Em Dart, as funções lambdas podem ser criadas usando a sintaxe de seta (`=>`) ou a palavra-chave `void` para funções sem retorno.

As funções lambdas em Dart são especialmente úteis quando você precisa passar uma função como argumento para outra função, como é comum em métodos de coleção, como `forEach`, `map` e `where`.

Exemplo

```
void main() {  
    // Recebe dois parâmetros e retorna sua soma  
    var soma = (int a, int b) => a + b;  
  
    // Chamada da função lambda  
    print(soma(2, 3)); // Saída: 5  
}
```

Exemplo

```
MaterialButton(  
  onPressed: () => exibirMensagem('Bom trabalho!'),  
  child: child,  
)
```

Exercício

04 Funções em Dart

Crie uma função em Dart que recebe uma lista de notas como argumento e retorna a média das notas. Em seguida, escreva um programa principal que solicita ao usuário uma lista de notas dos estudantes e chama a função criada para calcular a média. Exiba o resultado, representando o desempenho médio da turma.



Exercício

05 Lambdas em Dart

Implemente uma expressão lambda em Dart que recebe o valor original de um produto como argumento e retorna o valor com desconto. Essa expressão lambda deve aplicar uma porcentagem específica de desconto. Em seguida, solicite ao usuário o valor original de um produto, chama a expressão lambda para calcular o valor com desconto e exibe o resultado. Essa função simula o cálculo do preço final após a aplicação do desconto em uma loja online.



Referências

Documentação

Estruturas de controle de fluxo em Dart:
[Documentação do Dart: https://dart.dev/guides](https://dart.dev/guides)

Funções em Dart: [Documentação do Dart: https://dart.dev/guides](https://dart.dev/guides)

Expressões lambda em Dart: [Documentação do Dart: https://dart.dev/guides](https://dart.dev/guides)

Livros

"Beginning Flutter: A Hands On Guide To App Development" por Marco L. Napoli

"Dart for Absolute Beginners" por David Kopec.

Outros

[Programação em Dart - Youtube \(playlist\)](#)