

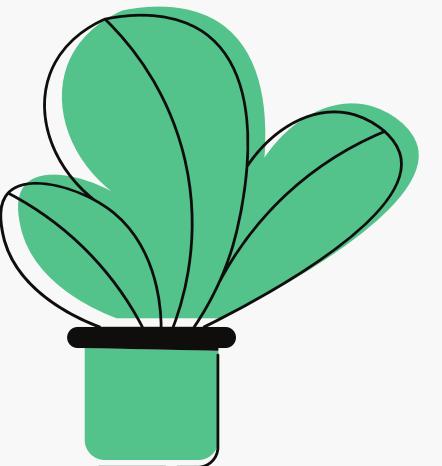
# Sintaxe da Linguagem

Curso Flutter de Verão



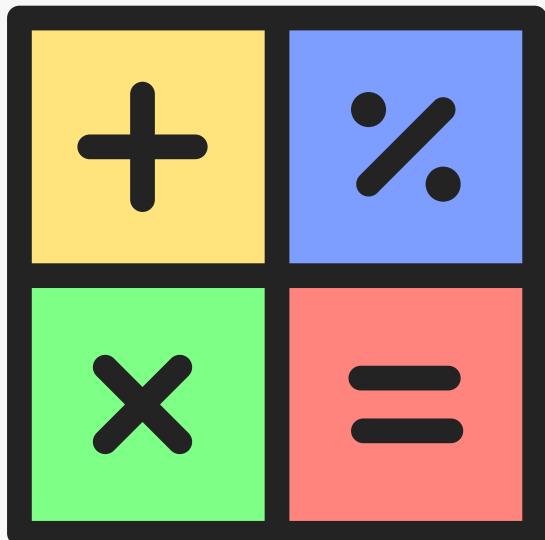
- Explorar os operadores aritméticos em Dart (soma, subtração, multiplicação e divisão).
- Discutir os operadores de comparação em Dart (igual, diferente, maior e menor que).
- Introduzir os operadores lógicos em Dart (e, ou, não e xou).
- Apresentar as estruturas de condição em Dart (if e switch-case).
- Realizar exercícios para aplicar os conceitos aprendidos na sintaxe da linguagem Dart.

# Objetivo da aula de hoje



# Revisão da aula anterior

# Operadores



Os operadores são símbolos especiais que permitem realizar operações em valores e variáveis. Eles podem ser utilizados para realizar cálculos matemáticos, combinar valores, comparar valores e executar outras operações.

Exemplos:

- + : adição
- % : resto da divisão
- \*= : atribuição com multiplicação
- >= : maior ou igual que
- && : E lógico
- ??= : Atribui se nulo

# Operadores aritméticos

Eles permitem que você execute adições, subtrações, multiplicações, divisões e outras operações matemáticas básicas em Dart.

Esses operadores são úteis para executar cálculos matemáticos básicos e manipular valores numéricos em Dart. Eles podem ser combinados e usados em expressões mais complexas para obter os resultados desejados.

# Exemplo

```
void main() {  
    int valor = 0;  
    double valorDouble = 0;  
  
    valor = 7 + 3; // Adição  
    valor = 7 - 3; // Subtração  
    valor = 7 * 3; // Multiplicação  
    divisaoDouble = 7 / 4; // Divisão  
    valor = 7 ~/ 4; // Divisão inteira  
}
```

# Exemplo

```
import 'dart:math';

void main() {
    double montanteInicial = 1000.0;
    double taxaJurosAnual = 0.05;
    int periodoInvestimentoAnos = 5;

    double valorFuturo = montanteInicial * pow(1 + taxaJurosAnual, periodoInvestimentoAnos);
    print("Valor Futuro: \$\$valorFuturo");
}
```

# Funções Matemáticas

```
void main() {  
    double sqrtResult = sqrt(16); // raiz quadrada  
    double expResult = exp(2); // exponencial  
    double logResult = log(10); // logaritmo natural  
    double log10Result = log10(100); // logaritmo na base 10  
    double sinResult = sin(0.5); // seno  
    double cosResult = cos(0.5); // cosseno  
    double tanResult = tan(0.5); // tangent  
    double asinResult = asin(0.5); // arco seno  
    double acosResult = acos(0.5); // arco cosseno  
    double atanResult = atan(0.5); // arco tangente  
}
```

# Operadores de atribuição

Os operadores de atribuição em Dart são usados para atribuir valores a variáveis. Eles permitem que você atualize o valor de uma variável usando diferentes operações matemáticas ou lógicas. Os operadores de atribuição combinam a atribuição de um valor a uma variável com uma operação específica.

Esses operadores de atribuição são úteis quando você deseja atualizar o valor de uma variável com base em uma operação específica, ao mesmo tempo em que atribui o resultado à mesma variável. Eles ajudam a simplificar a escrita de código e tornar suas expressões mais concisas.

# Exemplo

```
void main() {  
    // Atribuição simples  
    int valor = 3;  
  
    // Atribuição com Operadores aritméticos  
    valor += 3;  
  
    valor -= 3;  
  
    valor *= 3;  
  
    double valorDivisivel /= 3;  
  
    // Valores Nulo  
    int? valorNullable ??= 3;  
}
```

# Exemplo

— □ ×

```
Text text = Text("dattebayo");  
  
Container(  
    child: text,  
);
```

# Operadores de comparação

Os operadores de comparação são usados para comparar valores e determinar a relação entre eles. Eles retornam um valor booleano (verdadeiro ou falso) com base no resultado da comparação.

Em Dart, os operadores de comparação são usados principalmente para tomar decisões condicionais e controlar o fluxo de um programa.

# Exemplo

```
void main() {  
    bool igual = 7 == 3; // false  
    bool diferente = 7 != 3; // true  
    bool maiorQue = 7 > 3; // true  
    bool menorQue = 7 < 3; // false  
    bool maiorIgualQue = 7 >= 3; // true  
    bool menorIgualQue = 4 <= 4; // true  
}
```

# Exemplo

```
TextButton(  
    child: text,  
    onPressed: () {  
        if (DateTime.now().hour == 11){  
            print("dattebayo");  
        }  
    }  
);
```

# Operadores lógicos

Os operadores lógicos em Dart são usados para combinar e avaliar expressões lógicas. Eles trabalham com valores booleanos (verdadeiro ou falso) e retornam um resultado booleano com base na avaliação das expressões.

Além disso, é importante compreender a precedência dos operadores lógicos. O operador de negação (!) tem a maior precedência, seguido pelo E lógico (&&) e, por último, o OU lógico (||). No entanto, você pode usar parênteses para definir explicitamente a ordem de avaliação quando for necessário.

# Exemplo

```
void main() {  
    // ! é um 'não' lógico.  
    bool isFalse = !(7 > 3);  
  
    // && é um 'e' lógico.  
    bool isTrue = (7 > 3) && (3 < 7);  
  
    // !! é um 'ou' lógico.  
    bool atLeastOneTrue = (7 > 3) || (3 > 7);  
}
```

# Exemplo

– □ ×

```
if (assentosDisponiveis && maiorIdade && !vooLotado) {  
    return Text("Reserva confirmada. Assentos disponíveis e passageiro é  
    maior de idade.");  
} else {  
    return Text("Desculpe, não foi possível concluir a reserva.");  
}
```

# Exercício



## 01 Operadores Aritméticos

Imagine que você está desenvolvendo um programa de calculadora em Dart. O programa deve receber dois números do usuário e realizar as seguintes operações:

- Soma: some os dois números e exiba o resultado na tela.
- Subtração: subtraia o segundo número do primeiro número e exiba o resultado na tela.
- Multiplicação: multiplique os dois números e exiba o resultado na tela.
- Divisão: divida o primeiro número pelo segundo número e exiba o resultado na tela.
- Resto da divisão: calcule o resto da divisão do primeiro número pelo segundo número e exiba o resultado na tela.

# Exercício

## 02 Operadores de Comparação

Suponha que você está desenvolvendo um programa em Dart para verificar se um número digitado pelo usuário é par ou ímpar. O programa deve exibir uma mensagem indicando se o número é par ou ímpar.



# Estruturas de controle

As estruturas de controle permitem tomar decisões condicionais, repetir a execução de um bloco de código várias vezes e lidar com diferentes casos em uma lógica de programação.

Essas estruturas de controle são fundamentais na programação, pois permitem que o desenvolvedor controle o fluxo e o comportamento do programa com base em diferentes condições e requisitos específicos.

# If e Else

As estruturas de controle "if-else" são usadas para tomar decisões condicionais em um programa. Elas permitem que você execute diferentes blocos de código com base em uma condição específica.

A instrução if compara uma expressão e, se verdadeira, executa a lógica do código. A instrução if também oferece suporte a várias instruções else opcionais, que são usadas para avaliar vários cenários. Existem dois tipos de instruções else: else if e else.

# Exemplo

```
if (isClosed) {  
    print('Store is closed');  
}  
else if (isOpen) {  
    print('Store is open');  
}  
else if (isOutOfStock) {  
    print('Item is out of stock');  
}  
else {  
    print('Nothing matched');;  
}
```

# Exemplo

```
Column(  
    children: [  
        Text("Prioridade: ",  
            style: TextStyle(fontWeight: FontWeight.bold),  
        ),  
        if (prioridade == 1)  
            const Text("Baixa")  
        else if (prioridade == 2)  
            const Text("Média")  
        else if (prioridade == 3)  
            const Text("Alta")  
        else  
            const Text("Sem prioridade"),  
        Text("Outras informações: "),  
        ...  
    ]  
);
```

# Operador ternário

O operador ternário usa três argumentos e geralmente é usado quando apenas duas ações são necessárias. O operador ternário verifica o primeiro argumento para comparação, o segundo é a ação se o argumento for verdadeiro e o terceiro é a ação se o argumento for falso.

Os operadores ternários são úteis quando você precisa fazer uma decisão simples com base em uma condição. Eles permitem que você escreva o código de forma mais concisa e legível, especialmente em casos em que a lógica é direta e envolve apenas duas opções.

# Exemplo

- □ ×

```
// Maneira mais curta de declaração if e else  
isClosed ? askToOpen( ) : askToClose( );
```

# Exemplo

```
Container(  
    color: isContainer == true ? Colors.green : Colors.red,  
    child: child,  
);
```

# switch - case

As estruturas de condição "switch-case" são usadas para tomar decisões com base em diferentes valores ou casos específicos. Essa estrutura oferece uma alternativa ao uso de várias declarações "if-else" quando há uma necessidade de selecionar um bloco de código para executar com base em um valor específico.

A instrução switch avalia uma expressão e usa a cláusula case para corresponder a uma condição e executa o código dentro do caso correspondente. Cada cláusula case termina colocando uma instrução break como a última linha.

# Exemplo

```
String coffee = 'espresso';
switch (coffee) {
    case 'flavored':
        orderFlavored();
        break;
    case 'dark-roast':
        orderDarkRoast();
        break;
    case 'espresso':
        orderEspresso();
        break;
    default:
        orderNotAvailable();
}
```

# Exercício



## 03 Operadores Lógicos e Estruturas de Condição

Imagine que você esteja escrevendo um programa em Dart que solicita ao usuário um número e verifica as seguintes condições:

- O número é maior que 10 e menor que 20?
- O número é igual a 0 ou igual a 50?
- O número é diferente de 100 ou igual a 200?

Com base nessas condições, o programa deve exibir mensagens apropriadas indicando se as condições são verdadeiras ou falsas.

# Exercício



## 04 Estruturas de Condição – if e else

Assuma que você esteja desenvolvendo um programa em Dart para verificar se uma pessoa é maior de idade ou menor de idade com base em sua idade fornecida. Se a idade for maior ou igual a 18, o programa deve exibir a mensagem "Você é maior de idade". Caso contrário, o programa deve exibir a mensagem "Você é menor de idade".

# Exercício



## 05 Estruturas de Condição – switch-case

Vamos supor que você esteja escrevendo um programa em Dart que solicita ao usuário um dia da semana usando números (1 para segunda-feira, 2 para terça-feira, etc.) e exibe uma mensagem correspondente ao dia da semana usando a estrutura switch-case. Por exemplo, se o usuário digitar 1, o programa deve exibir a mensagem "Hoje é segunda-feira".

# Referências

## Documentação

Documentação do Dart: <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.