Laços de Repetição em Dart: Todas as Opções

Os laços de repetição em Dart são ferramentas fundamentais para a automação de tarefas repetitivas e para o processamento eficiente de coleções de dados. Eles aumentam a legibilidade e a eficiência do seu código.

Produzido por MARCIO SCHOENFELDER

Laço `for`: Repetições Controladas

Definição

Ideal quando o número de repetições é conhecido antecipadamente. Possui inicialização, condição e incremento explícitos.

Sintaxe

```
for (inicialização; condição; incremento) {
// código a ser repetido}
```

Exemplo

```
for (int i = 0; i < 3; i++) {  print(i);}
```

Saída

0, 1, 2

Laço `for-in`: Iterando Coleções

Propósito

Perfeito para percorrer todos os elementos de uma coleção, como Listas, Sets ou Maps, de forma simples e direta.

Exemplo Prático

```
List<String> cores = ['azul', 'verde', 'amarelo'];for (var cor in cores) {  print(cor);}
```

Saída

- azul
- verde
- amarelo

Laço `while`: Condições Flexíveis

Avaliação Constante

O laço continua executando enquanto uma condição específica permanecer verdadeira. A condição é verificada antes de cada iteração.

Exemplo

```
int contador = 0;while (contador < 2) {
print('Passo $contador');  contador++;}
```

Sintaxe

```
while (condição) {  // código a ser repetido}
```

Saída

Passo 0, Passo 1

Laço `do-while`: Execução Mínima Garantida

Garantia de Primeira Execução

Ao contrário do `while`, o `do-while` executa seu bloco de código pelo menos uma vez, mesmo que a condição seja falsa desde o início. A condição é verificada apenas após a primeira execução.

Sintaxe

```
do {  // código a ser executado} while (condição);
```

Exemplo

```
int valor = 10;do {  print('Valor: $valor');
valor++;} while (valor < 10);
```

Saída

Valor: 10

`break`: Interrompendo Laços Imediatamente

> O comando `break` oferece controle preciso sobre a execução de laços, permitindo que você saia de um laço instantaneamente quando uma condição específica é atendida.

```
for (int i = 0; i < 5; i++) {  if (i == 3) {
print('Parando em 3!');    break; // Sai do laço  }
print(i);}
```

Saída: 0, 1, 2, Parando em 3!

`continue`: Pulando a Iteração Atual

O comando `continue` permite que você pule a iteração atual de um laço e passe diretamente para a próxima, ignorando o restante do código dentro do bloco da iteração atual.

Sintaxe

```
for (...) {  if (condição) {    continue; // Pula para a próxima iteração  }  // Código que será ignorado se a condição for verdadeira}
```

Exemplo

```
for (int j = 0; j < 4; j++) {  if (j == 1) {
continue; // Pula o valor 1
}  print(j);}
```

Saída

0, 2, 3

Conclusão e Melhores Práticas

Escolha Inteligente

Use `for` para contagens, `for-in` para coleções e `while`/`do-while` para condições dinâmicas.

Controle Fino

Empregue `break` para saídas imediatas e `continue` para pular iterações específicas.

Cuidado Essencial

Sempre garanta que seus laços tenham uma condição de término para evitar laços infinitos e travamentos.

Dominar os laços de repetição é crucial para escrever código eficiente, limpo e performático em Dart.