Estruturas de repetição

Abaixo contem os itens que utilizamos para controlar o fluxo do seu código Dart usando loops e instruções de suporte:

- `for` para loops
- `while` e do `while` loops
- `break` e `continue`

For loops Em dart podemos iterar com o loop for padrão da seguinte maneira:

```
var message = StringBuffer('Dart is fun');
for (var i = 0; i < 5; i++) {
  message.write('!');
}

//Closures dentro dos loops for do Dart capturam o valor do índice. Isso
evita uma armadilha comum encontrada em JavaScript
var callbacks = [];
for (var i = 0; i < 2; i++) {
  callbacks.add(() => print(i));
}

for (final c in callbacks) {
  c();
}
```

Às vezes, não precisamos saber o contador de iteração atual ao iterar sobre um tipo Iterable, como List ou Set. Nesse caso, podemos utilizar o loop for-in para um código mais limpo

```
for (final candidate in candidates) {
  candidate.interview();
}

//Para processar os valores obtidos do iterável, você também pode usar um
padrão em um loop for-in
for (final Candidate(:name, :yearsExperience) in candidates) {
  print('$name has $yearsExperience of experience.');
}

//As classes iteráveis também têm um método forEach() como outra opção
var collection = [1, 2, 3];
collection.forEach(print); //Saída: 1 2 3
```

While and do-while

Um `while` avalia a condição antes do loop

```
while (!isDone()) {
  doSomething();
}
```

Um `do-while` avalia a condição após o loop:

```
do {
  printLine();
} while (!atEndOfPage());
```

Break and continue Use `break` para interromper o loop:

```
while (true) {
  if (shutDownRequested()) break;
  processIncomingRequests();
}
```

Use `continue` para pular para a próxima iteração do loop:

```
for (int i = 0; i < candidates.length; i++) {
  var candidate = candidates[i];
  if (candidate.yearsExperience < 5) {
    continue;
  }
  candidate.interview();
}
```