Estruturas de decisão

Em dart podemos controlar o fluxo do nosso código usando as seguintes estruturas de decisões:

- if: declarações e elementos
- if-case: declarações e elementos
- switch: declarações e expressões

if Dart suporta instruções if com cláusulas else opcionais. A condição entre parênteses após if deve ser uma expressão que avalia para um booleano:

```
if (estaChovendo()) {
  you.tragaCapaDeChuva();
} else if (estaNevando()) {
  you.vistaUmaJaqueta();
} else {
  car.colocarDeCimaParaBaixo();
}
```

If-case

As instruções Dart `if` suportam cláusulas `case` seguidas por um pattern

```
//Se o padrão corresponder ao valor, a ramificação será executada com
quaisquer variáveis que o padrão definir no escopo.
if (pair case [int x, int y]) return Point(x, y);

//ou também podemos escrever desta outra maneira
if (pair case [int x, int y]) {
  print('Was coordinate array $x,$y');
} else {
  throw FormatException('Invalid coordinates.');
}
```

Switch statements

Uma instrução `switch` avalia uma expressão de valor em relação a uma série de casos. Cada cláusula `case` é um padrão para o valor a ser correspondido. Você pode usar qualquer tipo de padrão para um caso.

Quando o valor corresponde ao padrão de um `case`, o corpo do `case` é executado. Cláusulas de `case` não vazias saltam para o final do switch após a conclusão.

Elas não exigem uma instrução `break`. Outras maneiras válidas de terminar uma cláusula `case` não vazia são uma instrução `continue`, `throw` ou `return`.

```
var command = 'OPEN';
switch (command) {
```

```
  case 'CLOSED':
    executeClosed();
  case 'PENDING':
    executePending();
  case 'APPROVED':
    executeApproved();
  case 'DENIED':
    executeDenied();
  case 'OPEN':
    executeOpen();
  default:
    executeUnknown();
}
```

Switch expressions Uma expressão `switch` produz um valor com base no corpo da expressão de qualquer case que corresponda. Você pode usar uma expressão switch sempre que o Dart permitir expressões, `exceto` no início de uma declaração de expressão.

```
var x = switch (y) { ... };

print(switch (x) { ... });

return switch (x) { ... };


// As expressões de troca permitem que você reescreva uma instrução switch
como esta
switch (charCode) {
  case slash || star || plus || minus: // Logical-or pattern
    token = operator(charCode);
  case comma || semicolon: // Logical-or pattern
    token = punctuation(charCode);
  case >= digit0 && <= digit9: // Relational and logical-and patterns
    token = number();
  default:
    throw FormatException('Invalid');
}

//Em uma expressão, como esta
token = switch (charCode) {
  slash || star || plus || minus => operator(charCode),
  comma || semicolon => punctuation(charCode),
  >= digit0 && <= digit9 => number(),
  _ => throw FormatException('Invalid')
};
```