Guia inicial de Dart

Criando projeto

Para criar um projeto em Dart, basta executar o comando abaixo:

```
dart create <nome_projeto>
```

Executando projeto

Para executar um projeto em Dart, basta executar o comando abaixo dentro da pasta do projeto:

```
dart run
```

Tipos de variáveis

String

```
String nome = "João";
```

Int

```
int idade = 20;
```

Double

```
double altura = 1.80;
```

Boolean

```
bool isTrue = true;
```

Dynamic

```
dynamic variavel = "João";
variavel = 20;
variavel = 1.80;
variavel = true;
```

Const

```
const pi = 3.14;
```

Final (variável imutável)

```
final pi = 3.14;
```

Var (variável dinâmica)

```
var nome = "João"; //var é dinâmico (variável mutável)
var idade = 20;
var altura = 1.80;
var isTrue = true;
```

Operadores

Aritméticos

```
int a = 10;
int b = 20;
int soma = a + b;
int subtracao = a - b;
int multiplicacao = a * b;
double divisao = a / b;
int resto = a % b;
```

Relacionais

```
int a = 10;
int b = 20;
bool maior = a > b;
bool menor = a < b;
bool maiorOuIgual = a >= b;
bool menorOuIgual = a <= b;
bool igual = a == b;
bool diferente = a != b;
```

Lógicos

```
bool verdadeiro = true;
bool falso = false;
bool resultado = verdadeiro && falso;
bool resultado = verdadeiro || falso;
bool resultado = !verdadeiro;
```

Atribuição

```
int a = 10;
int b = 20;
int c = 30;
int d = 40;
int e = 50;
a += 10; // a = a + 10;
b -= 10; // b = b - 10;
c *= 10; // c = c * 10;
```

```
d /= 10; // d = d / 10;
e %= 10; // e = e % 10;
```

Incremento e Decremento

```
int a = 10;
int b = 20;
a++; // a = a + 1;
b--; // b = b - 1;
```

Estruturas condicionais

If

```
int idade = 20;
if (idade >= 18) {
  print("Maior de idade");
}
```

If/Else

```
int idade = 20;
if (idade >= 18) {
  print("Maior de idade");
} else {
  print("Menor de idade");
}
```

If/Else If

```
int idade = 20;
if (idade >= 18) {
  print("Maior de idade");
} else if (idade >= 12) {
  print("Adolescente");
} else {
  print("Criança");
}
```

Switch

```
int dia = 1;
switch (dia) {
  case 1:
    print("Domingo");
    break;
  case 2:
```

```
    print("Segunda");
    break;
  case 3:
    print("Terça");
    break;
  case 4:
    print("Quarta");
    break;
  case 5:
    print("Quinta");
    break;
  case 6:
    print("Sexta");
    break;
  case 7:
    print("Sábado");
    break;
  default:
    print("Dia inválido");
}
```

Estruturas de repetição

For

```
for (int i = 0; i < 10; i++) {
  print(i);
}
```

While

```
int i = 0;
while (i < 10) {
  print(i);
  i++;
}
```

Do/While

```
int i = 0;
do {
  print(i);
  i++;
} while (i < 10);
```

Funções

Função sem retorno

```
void soma(int a, int b) {
  print(a + b);
}
```

Função com retorno

```
int soma(int a, int b) {
  return a + b;
}
```

Função com parâmetro opcional

```
void soma(int a, [int b]) {
  print(a + b);
}
```

Função com parâmetro opcional nomeado

```
void soma(int a, {int b}) {
  print(a + b);
}
```

Função anônima

```
() {
  print("Função anônima");
}
```

Arrow function

```
() => print("Arrow function");
```

Classes

Classe

```
class Pessoa {
  String nome;
  int idade;
  double altura;
  bool isMaiorDeIdade;
}
```

Construtor

```
class Pessoa {
  String nome;
  int idade;
  double altura;
  bool isMaiorDeIdade;

  Pessoa(String nome, int idade, double altura, bool isMaiorDeIdade) {
    this.nome = nome;
    this.idade = idade;
    this.altura = altura;
    this.isMaiorDeIdade = isMaiorDeIdade;
  }
}
```

Construtor com parâmetros nomeados

```
class Pessoa {
  String nome;
  int idade;
  double altura;
  bool isMaiorDeIdade;

  Pessoa({this.nome, this.idade, this.altura, this.isMaiorDeIdade});
}
```

Getters e Setters

```
class Pessoa {
  String _nome;
  int _idade;
  double _altura;
  bool _isMaiorDeIdade;

  Pessoa({String nome, int idade, double altura, bool isMaiorDeIdade}) {
    this._nome = nome;
    this._idade = idade;
    this._altura = altura;
    this._isMaiorDeIdade = isMaiorDeIdade;
  }

  String get nome => _nome;
  set nome(String nome) => _nome = nome;

  int get idade => _idade;
  set idade(int idade) => _idade = idade;
```

```
  double get altura => _altura;
  set altura(double altura) => _altura = altura;

  bool get isMaiorDeIdade => _isMaiorDeIdade;
  set isMaiorDeIdade(bool isMaiorDeIdade) => _isMaiorDeIdade = isMaiorDeIdade;
}
```

Herança

```
class Pessoa {
  String _nome;
  int _idade;
  double _altura;
  bool _isMaiorDeIdade;

  Pessoa({String nome, int idade, double altura, bool isMaiorDeIdade}) {
    this._nome = nome;
    this._idade = idade;
    this._altura = altura;
    this._isMaiorDeIdade = isMaiorDeIdade;
  }

  String get nome => _nome;
  set nome(String nome) => _nome = nome;

  int get idade => _idade;
  set idade(int idade) => _idade = idade;

  double get altura => _altura;
  set altura(double altura) => _altura = altura;

  bool get isMaiorDeIdade => _isMaiorDeIdade;
  set isMaiorDeIdade(bool isMaiorDeIdade) => _isMaiorDeIdade = isMaiorDeIdade;
}

class PessoaFisica extends Pessoa {
  String _cpf;

  PessoaFisica(
      {String nome, int idade, double altura, bool isMaiorDeIdade, String cpf})
      : super(
            nome: nome,
            idade: idade,
            altura: altura,
            isMaiorDeIdade: isMaiorDeIdade) {
```

```
    this._cpf = cpf;
  }

  String get cpf => _cpf;
  set cpf(String cpf) => _cpf = cpf;
}

class PessoaJuridica extends Pessoa {
  String _cnpj;

  PessoaJuridica(
      {String nome, int idade, double altura, bool isMaiorDeIdade, String cnpj})
      : super(
            nome: nome,
            idade: idade,
            altura: altura,
            isMaiorDeIdade: isMaiorDeIdade) {
    this._cnpj = cnpj;
  }

  String get cnpj => _cnpj;
  set cnpj(String cnpj) => _cnpj = cnpj;
}
```

Métodos

```
class Pessoa {
  String _nome;
  int _idade;
  double _altura;
  bool _isMaiorDeIdade;

  Pessoa({String nome, int idade, double altura, bool isMaiorDeIdade}) {
    this._nome = nome;
    this._idade = idade;
    this._altura = altura;
    this._isMaiorDeIdade = isMaiorDeIdade;
  }

  String get nome => _nome;
  set nome(String nome) => _nome = nome;

  int get idade => _idade;
  set idade(int idade) => _idade = idade;

  double get altura => _altura;
```

```
  set altura(double altura) => _altura = altura;

  bool get isMaiorDeIdade => _isMaiorDeIdade;
  set isMaiorDeIdade(bool isMaiorDeIdade) => _isMaiorDeIdade = isMaiorDeIdade;

  void dormir() {
    print("Dormir");
  }

  void comer() {
    print("Comer");
  }

  void andar() {
    print("Andar");
  }
}
```

Classes abstratas

```
abstract class Pessoa {
  String _nome;
  int _idade;
  double _altura;
  bool _isMaiorDeIdade;

  Pessoa({String nome, int idade, double altura, bool isMaiorDeIdade}) {
    this._nome = nome;
    this._idade = idade;
    this._altura = altura;
    this._isMaiorDeIdade = isMaiorDeIdade;
  }

  String get nome => _nome;
  set nome(String nome) => _nome = nome;

  int get idade => _idade;
  set idade(int idade) => _idade = idade;

  double get altura => _altura;
  set altura(double altura) => _altura = altura;

  bool get isMaiorDeIdade => _isMaiorDeIdade;
  set isMaiorDeIdade(bool isMaiorDeIdade) => _isMaiorDeIdade = isMaiorDeIdade;

  void dormir() {
```

```
    print("Dormir");
  }

  void comer() {
    print("Comer");
  }

  void andar() {
    print("Andar");
  }
}

class PessoaFisica extends Pessoa {
  String _cpf;

  PessoaFisica(
      {String nome, int idade, double altura, bool isMaiorDeIdade, String cpf})
      : super(
            nome: nome,
            idade: idade,
            altura: altura,
            isMaiorDeIdade: isMaiorDeIdade) {
    this._cpf = cpf;
  }

  String get cpf => _cpf;
  set cpf(String cpf) => _cpf = cpf;
}

class PessoaJuridica extends Pessoa {
  String _cnpj;

  PessoaJuridica(
      {String nome, int idade, double altura, bool isMaiorDeIdade, String cnpj})
      : super(
            nome: nome,
            idade: idade,
            altura: altura,
            isMaiorDeIdade: isMaiorDeIdade) {
    this._cnpj = cnpj;
  }

  String get cnpj => _cnpj;
  set cnpj(String cnpj) => _cnpj = cnpj;
}
```

Classes estáticas

```
class Pessoa {
  String _nome;
  int _idade;
  double _altura;
  bool _isMaiorDeIdade;

  Pessoa({String nome, int idade, double altura, bool isMaiorDeIdade}) {
    this._nome = nome;
    this._idade = idade;
    this._altura = altura;
    this._isMaiorDeIdade = isMaiorDeIdade;
  }

  String get nome => _nome;
  set nome(String nome) => _nome = nome;

  int get idade => _idade;
  set idade(int idade) => _idade = idade;

  double get altura => _altura;
  set altura(double altura) => _altura = altura;

  bool get isMaiorDeIdade => _isMaiorDeIdade;
  set isMaiorDeIdade(bool isMaiorDeIdade) => _isMaiorDeIdade = isMaiorDeIdade;

  void dormir() {
    print("Dormir");
  }

  void comer() {
    print("Comer");
  }

  void andar() {
    print("Andar");
  }

  static void falar() {
    print("Falar");
  }
}
```

Classes genéricas

```
class Pessoa<T, U> {
  T _nome;
  U _idade;

  Pessoa(this._nome, this._idade);

  T get nome => _nome;
  set nome(T nome) => _nome = nome;

  U get idade => _idade;
  set idade(U idade) => _idade = idade;
}
```

Tratamento de erros

Try/Catch

```
try {
  int.parse("João");
} catch (e) {
  print("Erro: $e");
}
```

Try/Catch/Finally

```
try {
  int.parse("João");
} catch (e) {
  print("Erro: $e");
} finally {
  print("Finally");
}
```

Try/On/Catch

```
try {
  int.parse("João");
} on FormatException catch (e) {
  print("Erro: $e");
}
```

Try/On/Catch/Stacktrace

```
try {
  int.parse("João");
} on FormatException catch (e, s) {
```

```
  print("Erro: $e");
  print("Stacktrace: $s");
}
```

Try/On/Catch/Finally

```
try {
  int.parse("João");
} on FormatException catch (e) {
  print("Erro: $e");
} finally {
  print("Finally");
}
```

Try/On/Catch/Stacktrace/Finally

```
try {
  int.parse("João");
} on FormatException catch (e, s) {
  print("Erro: $e");
  print("Stacktrace: $s");
} finally {
  print("Finally");
}
```

Try/On/Catch/Stacktrace/Finally/Rethrow

```
try {
  int.parse("João");
} on FormatException catch (e, s) {
  print("Erro: $e");
  print("Stacktrace: $s");
  rethrow;
} finally {
  print("Finally");
}
```

Try/On/Catch/Stacktrace/Finally/Rethrow/Throw

```
try {
  int.parse("João");
} on FormatException catch (e, s) {
  print("Erro: $e");
  print("Stacktrace: $s");
  rethrow;
} catch (e) {
  print("Erro: $e");
```

```
  throw Exception("Erro ao converter idade");
} finally {
  print("Finally");
}
```

Exceções

Exception

```
throw Exception("Erro ao converter idade");
```

FormatException

```
throw FormatException("Erro ao converter idade");
```

IntegerDivisionByZeroException

```
throw IntegerDivisionByZeroException();
```

IOException

```
throw IOException();
```

TimeoutException

```
throw TimeoutException("Tempo de conexão esgotado");
```

Error

```
throw Error();
```

AssertionError

```
throw AssertionError();
```

CastError

```
throw CastError();
```

NullThrownError

```
throw NullThrownError();
```

RangeError

```
throw RangeError();
```

StateError

```
throw StateError("Erro de estado");
```

UnimplementedError

```
throw UnimplementedError();
```

UnsupportedError

```
throw UnsupportedError("Operação não suportada");
```

Referências

Mais informações sobre Dart em: dart.dev

Material criado por: Higor Pereira