



Curso de Flutter



Flutter

Instrutor: 1º Ten PM Otniel CAMPOS da Silva



Tópicos

- 1) Contexto do Desenvolvimento Mobile
- 2) O que é o Flutter
- 3) Linguagem de programação DART
- 3) Porque o Flutter?
- 4) Ambiente de Desenvolvimento Dart (prática)
- 5) Hello World
- 6) Sintaxe
- 7) Operadores
- 8) Funções



Contexto Desenvolvimento Mobile

ANDROID vs IOS



Contexto Desenvolvimento Mobile

IOS 2007, desenvolvimento nativo Objective C/Swift

Android 2008, desenvolvimento nativo em JAVA/Kotlin

Windows Phone 2010, desenvolvimento nativo C#

O que é desenvolvimento nativo?

Desenvolvimento nativo é o desenvolvimento de um software que se comunica diretamente com os recursos do dispositivo.

PROBLEMA?

Cada SO precisava ter um app com um código diferente, e com linguagens diferentes.

Altos custos para desenvolvimento e manutenção multiplataforma.



Contexto Desenvolvimento Mobile

Solução:

Desenvolvimento Multiplataforma.

Desenvolvimento multiplataforma, é o desenvolvimento por meio de frameworks que possibilita a criação de aplicativos para diversas plataformas: android, ios, web, desktop

As empresas passaram a adotar o desenvolvimento híbrido economizando tempo, dinheiro e recursos humanos.



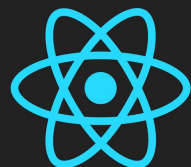
Contexto Desenvolvimento Mobile

A maior vantagem do desenvolvimento multiplataforma, é que com o domínio de apenas uma linguagem de programação , conseguimos desenvolver para qualquer plataforma.

Dévido as peculiaridades de cada SO, nem sempre com apenas um código iremos compilar um app para as diversas plataformas.

Contexto Desenvolvimento Mobile

FRAMEWORKS DE DESENVOLVIMENTO MOBILE MULTIPLATAFORMA



React Native



Contexto Desenvolvimento Mobile

Microsoft foi Pioneira no desenvolvimento multiplataforma com o framework XAMARIN, que a gigante adquiriu no ano de 2016,

O xamarin prometia revolucionar o desenvolvimento mobile, mas com o fim do windows phone, a ferramenta perdeu muita força.

Com o tempo outras gigantes da Tecnologia investiram no desenvolvimento multiplataforma, criando ferramentas que são atualizadas constantemente, possuem uma comunidade engajada, e são mantidas por empresas ligadas ao mundo mobile.



O que é o Flutter

Segundo a própria GOOGLE , empresa que mantém a tecnologia,

“Flutter é o kit de ferramentas de IU do Google para construir aplicativos bonitos e nativamente compilados para dispositivos móveis , web , desktop e incorporados a partir de uma única base de código”.

Linguagem de programação DART



Why Flutter
uses Dart



DART

Linguagem de programação multiparadigma, criada em 2011 e mantida pela Google.

- Orientada a objetos

- Funcional

Sintaxe baseada no C

Utilização em front e back-end



Porque FLutter

Motivos para se estudar o flutter:

Framework mantido pela google , empresa renomada e a mesma empresa que mantém o android.

Framework com suporte a multiplataforma: Android, ios, Desktop, Web.

FUCHSIA*



Ambiente de Desenvolvimento Dart

No curso iremos usar o windows 10 arquitetura x64

O dart possui suporte para windows , linux e macOS

Acesse e confira compatibilidade com o seu SO:

<https://dart.dev/get-dart>

Caso não esteja possibilitado em fazer a instalação do dart,acesse e treine com o dart na web:

https://dartpad.dev/?null_safety=true



Instalando o dart

Para instalar o DART em sua máquina, vamos instalar o flutter diretamente, a página oficial do dart recomenda a instalação do flutter diretamente , não sendo necessário fazer uma instalação separada.

<https://dart.dev/tools/sdk>

Além disso precisaremos instalar e configurar uma IDE própria para executarmos testarmos nossos códigos em Flutter.


A IDE utilizada em nosso curso será o Android Studio , mas também é possível fazer o desenvolvimento em Flutter com o Visual Studio que é mais leve que o Android Studio.

JDK

Pré requisito para o Android Studio será instalar o JDK e JRE:

<https://www.oracle.com/java/technologies/javase-downloads.html>



Windows x64 Installer	150.58 MB	 jdk-16.0.2_windows-x64_bin.exe
-----------------------	-----------	--

Inicie a instalação



JRE

Acesse :<https://www.oracle.com/java/technologies/javase-downloads.html>

Em JAVA SE 8 clique em JRE e faça o download da versão windows 64x.

REalize a instalação do JRE.

Após a instalação do JDK e do JRE abra o CMD e escreva o seguinte comando para verificar a instalação

```
java -version
```

```
java version "1.8.0_261"  
Java(TM) SE Runtime Environment (build 1.8.0_261-b12)  
Java HotSpot(TM) 64-Bit Server VM (build 25.261-b12, mixed mode)
```

Caso não funcione, verifique as variáveis de ambiente do JAVA



Android Studio

Acesse: <https://developer.android.com/studio/index.html> e faça o Download do AS

Instale o AS sem configurações personalizadas, apenas next=>

Execute o AS para verificar se a instalação ocorreu normalmente



Flutter SDK

Baixe o Flutter SDK em: <https://flutter.dev/docs/get-started/install/windows>

Crie uma pasta C:\src

Inicia o AS novamente

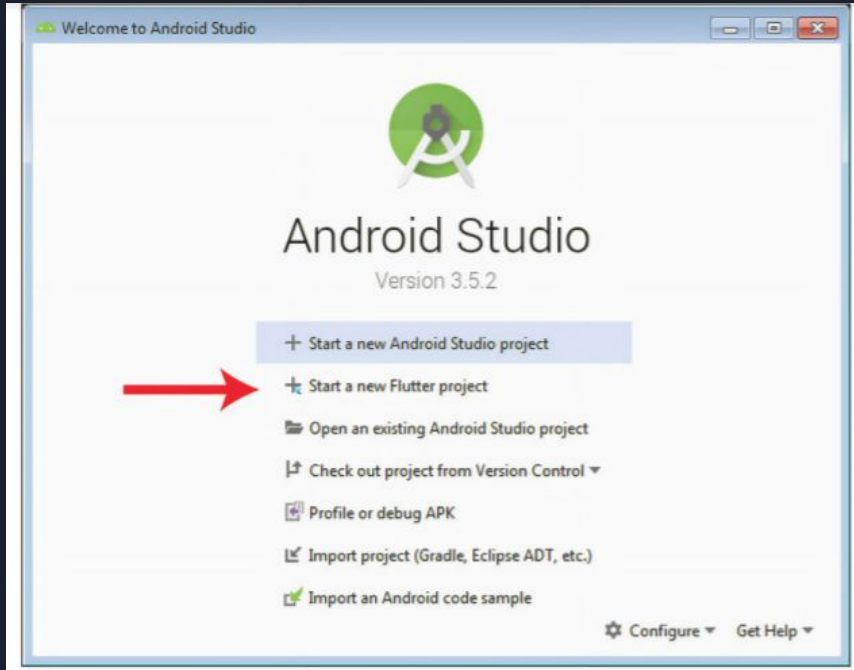
Navegue até Configurações -> Plugins -> no marketplace digite Flutter , selecione a opção Flutter e instale esse plugin.

Caso apareça mensagens , clique em aceitar.

Reinicie a IDE

Flutter SDK

Quando reiniciar o AS verifique se existe a opção de Iniciar um novo projeto em FLutter como na imagem seguinte



Verifique a instalação com o seguinte comando no CMD:
flutter doctor

Caso ocorra uma mensagem de erro, configure o FLutter nas variáveis de ambiente.

<https://medium.com/fnplus/setting-up-flutter-for-windows-ca2c5e643fdf>

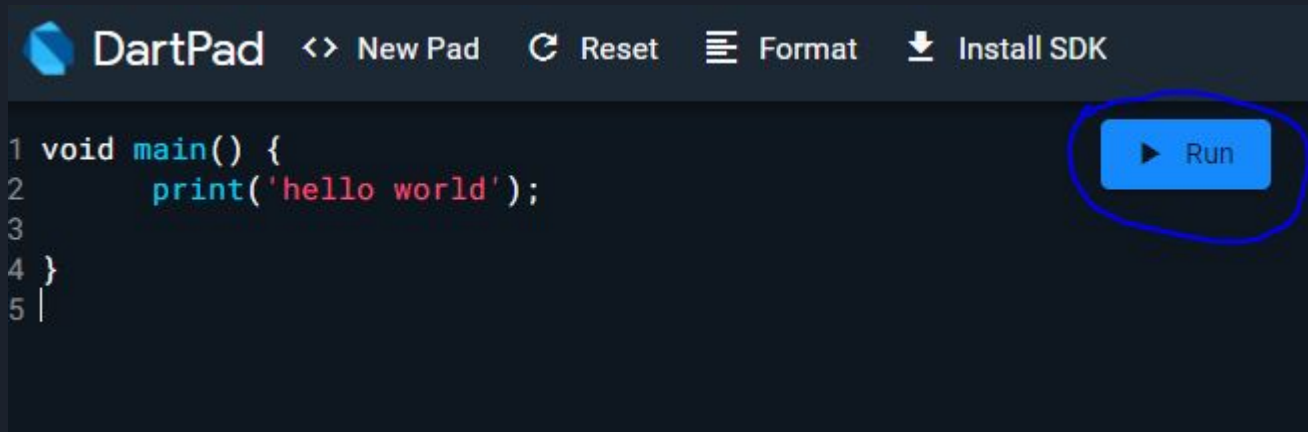


Linguagem Dart: Hello World

- 1) Para ganharmos tempo vou explicar a sintaxe da linguagem DART, levando em consideração o que ela tem de diferente com o que estamos mais acostumados na programação. Para se aprofundar em toda a linguagem acesse sua documentação oficial em: <https://dart.dev/guides>
- 2) Irei utilizar o dart na web ,https://dartpad.dev/?null_safety=true , para ganharmos tempo com a explicação e poderemos colocar a mão na massa o quanto antes e criarmos nosso primeiro APP.

Linguagem Dart: Hello World

Para dar nosso primeiro Hello World com o DART
escreva o código com a seguinte estrutura:



The image shows the DartPad web interface. At the top, there is a dark blue header bar with the DartPad logo and several menu items: '<> New Pad', a circular arrow icon for 'Reset', a list icon for 'Format', and a download icon for 'Install SDK'. Below the header is a code editor with a dark background. The code is as follows:

```
1 void main() {  
2     print('hello world');  
3  
4 }  
5 |
```

On the right side of the code editor, there is a blue button with a play icon and the text 'Run'. This button is circled with a blue hand-drawn line.

O main() é o ponto de partida do nosso código em dart.



DART Tipos de Dados

- booleans
- lists
- maps
- numbers
- runes
- sets
- strings

*Um valor que receba Var no lugar do tipo de dado , o tipo de dado será interpretado de acordo com o valor recebido.



Dart Tipos de Dados

num=>inteiros ou Floats => num total =10 ou num altura = 1.55

int => números inteiros => int total = 10

double => números de ponto flutuante => double altura=1.55

String => cadeia de caracteres => " ou ""

Booleans=>>false ou true => bool mentira =true;

Listas=>No Dart arrays são do tipo List => var estados=[rj,sp,mg]

Sets=> coleção não ordenada de itens únicos => var cores ={'azul','vermelho','vermelho','amarelo'}

Map => Objeto chave e valor var Usuarios ={'1':João,

'2':Maria'

}

Runes =>Caracteres Unicode UTF-16 (emojis) => `Runes emojis = new Runes("\u{1F603}");`

`print(new String.fromCharCode(emojis));`



DART Operadores


Além dos operadores que estamos acostumados o DART possui alguns diferentes que nos auxiliam durante o desenvolvimento.

1) Operadores de teste de tipo

`as`, `is`, `is!` => `as` Converte de um subtipo válido para um tipo. Ex.: converter de `int` para `num`.

`is` Verifica se o valor é de um determinado tipo e `is!` Verifica se o valor não é de um determinado tipo

Dart Operadores

 DartPad <> New Pad ↺ Reset ≡ Format ⬇ Install SDK ensorcelled-gu

```
1 void main() {  
2  
3   var nota = 5;  
4  
5   nota as double;  
6  
7   if (nota is double){  
8     print('nota é double');  
9   }  
10  
11   nota as num;  
12  
13   if (nota is num){  
14     print('nota é num');  
15   }  
16  
17   if(nota is! String){  
18     print('nota não é String');  
19   }  
20  
21  
22 }
```

▶ Run

Console

nota é double
nota é num
nota não é String

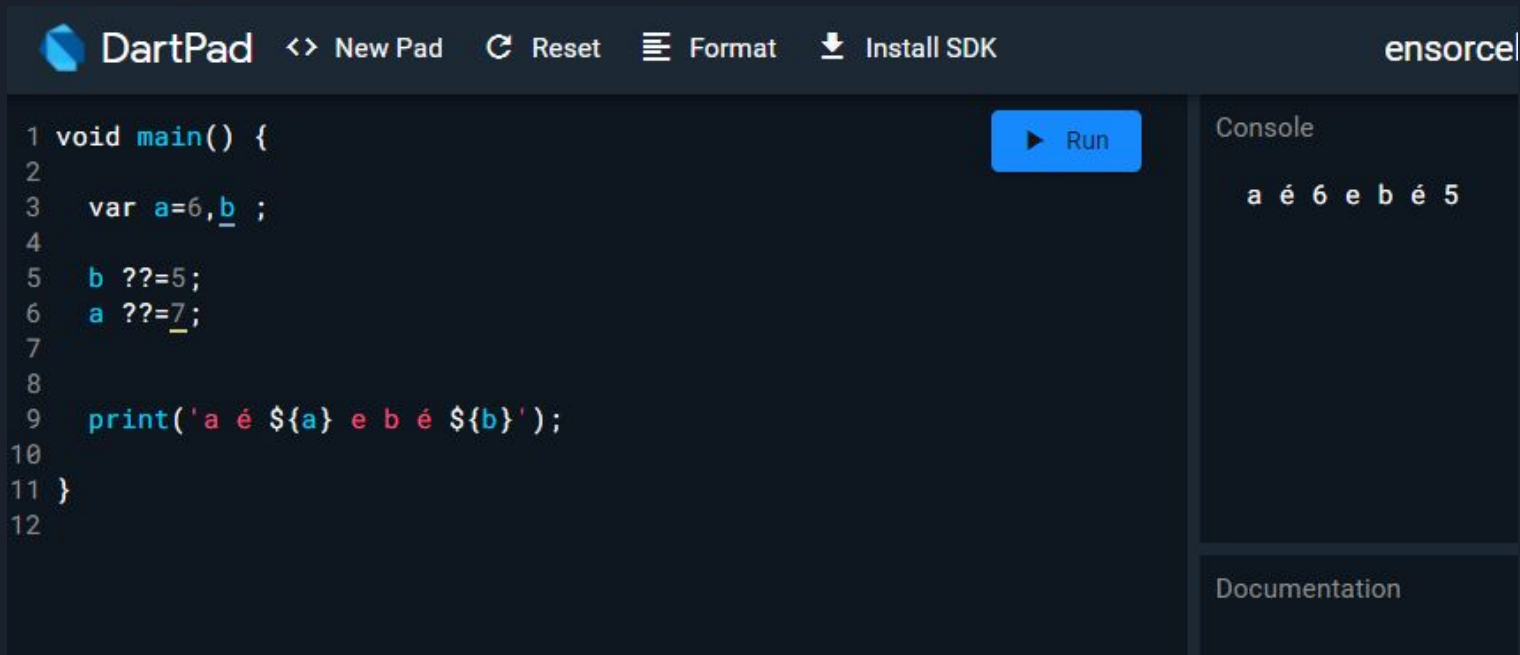
Documentation

Dart Operadores

Operadores de atribuição

(=) => Atribui valor a uma variável

(??=) => Se a variável for nula atribui o valor, senão mantém o valor anterior



The screenshot shows the DartPad web editor interface. At the top, there's a toolbar with icons for 'New Pad', 'Reset', 'Format', and 'Install SDK'. The main area contains Dart code with line numbers 1 through 12. A blue 'Run' button is positioned to the right of the code. On the right side, there's a 'Console' panel showing the output of the code, and a 'Documentation' panel below it.

```
1 void main() {  
2  
3   var a=6, b ;  
4  
5   b ??=5;  
6   a ??=7;  
7  
8  
9   print('a é ${a} e b é ${b}');  
10  
11 }  
12
```

Run

Console

a é 6 e b é 5

Documentation

Dart Operadores

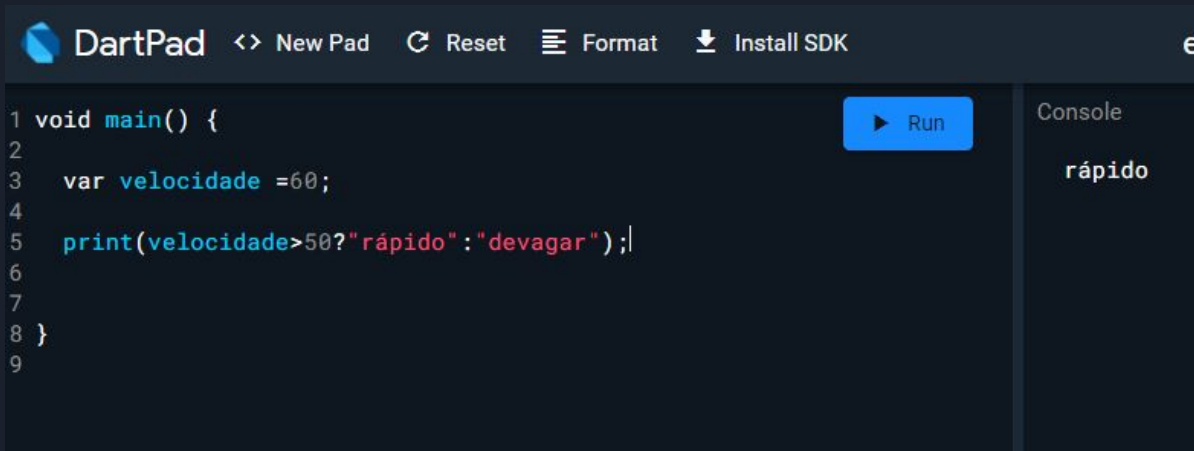
Operadores Condicionais

O operadores condicionais servem para a criação de expressões condicionais.

os operadores condicionais clássicos que conhecemos são da estrutura if-else.

No dart temos a opção de escrever um if else com a seguinte expressão.

```
condição ? expressão1 : expressão2
```



The screenshot shows the DartPad web interface. At the top, there's a toolbar with icons for 'New Pad', 'Reset', 'Format', and 'Install SDK'. Below the toolbar is a code editor with the following Dart code:

```
1 void main() {  
2  
3   var velocidade = 60;  
4  
5   print(velocidade > 50 ? "rápido" : "devagar");  
6  
7  
8 }  
9
```

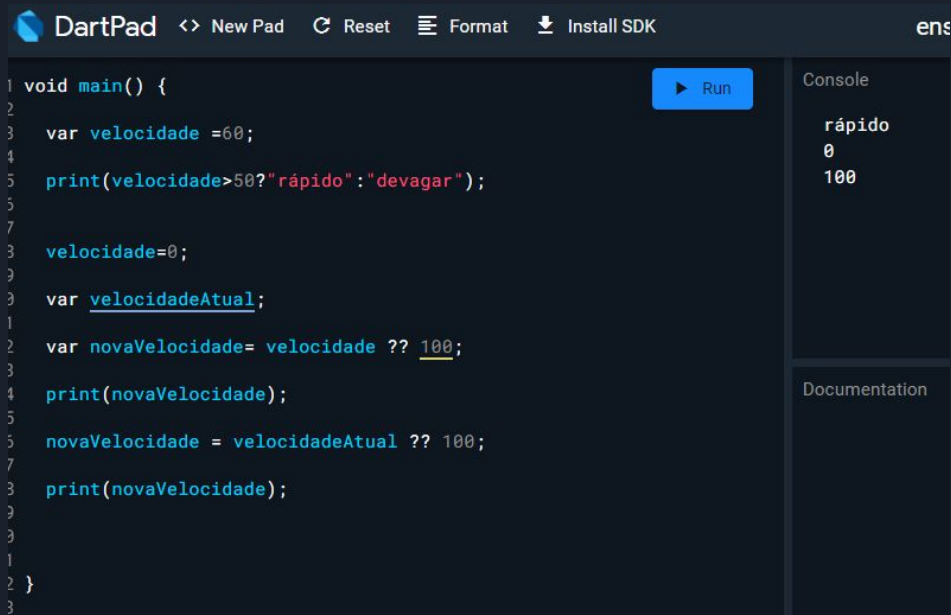
To the right of the code editor is a 'Console' panel. It contains the output 'rápido', which is the result of the conditional expression in the code. A blue 'Run' button is located to the right of the code editor.

Dart Operadores

Operadores Condicionais

`expressao1 ?? expressao2`

Se `expressao1` não for nulo retorna um valor, senão passa para a `expressao2` e retorna outro valor.



```
1 void main() {  
2  
3   var velocidade = 60;  
4  
5   print(velocidade > 50 ? "rápido" : "devagar");  
6  
7  
8   velocidade = 0;  
9  
10  var velocidadeAtual;  
11  
12  var novaVelocidade = velocidade ?? 100;  
13  
14  print(novaVelocidade);  
15  
16  novaVelocidade = velocidadeAtual ?? 100;  
17  
18  print(novaVelocidade);  
19  
20  
21 }  
22  
23
```

The screenshot shows the DartPad interface with a code editor on the left and a console on the right. The code in the editor demonstrates the use of the `??` operator. It starts with a `main` function containing several lines of code. The first part sets a variable `velocidade` to 60 and prints it, resulting in the output "rápido". The second part sets `velocidade` to 0 and prints it, resulting in the output "devagar". The third part uses the `??` operator to set `novaVelocidade` to either the value of `velocidade` (which is 0) or 100. Since 0 is not null, `novaVelocidade` becomes 0. This is printed, resulting in the output "0". The fourth part uses the `??` operator again, but this time with `velocidadeAtual`, which is null. Since `velocidadeAtual` is null, `novaVelocidade` becomes 100. This is printed, resulting in the output "100". The console on the right shows the output of the code: "rápido", "0", and "100".



Dart Variáveis e Constantes

Var => var no lugar do tipo de dado e atribuir um valor a ela

Object e dynamic => permitem atribuir qualquer valor a uma variável

Quando não atribuímos valor, o dart atribui null por padrão.

Final e Const

Variável que não deve mudar o valor

Final pode receber o valor em tempo de execução ou quando declarada e Const deve receber o valor quando declarada.



Template String e Arrow functions

Talvez você deve ter estranhado como concatenamos as Strings durante o essa aula.

o template String é uma alternativa a forma clássica, para concatenar strings e variáveis .

forma clássica:

```
print('Texto' + variavel)
```

Template String:

```
print('Texto ${variavel}')
```



Functions

Funções ajudam a desacoplar nosso código, mas como devemos declarar-las?

```
printAlgo(){
```

```
  print("Hello World")
```

```
}
```

```
String minhaPatente(){
```

```
  return "1º Tenente"
```

```
}
```

```
num calcularSalario(soldo,gratificacoes){
```

```
  var total =soldo + gratificacoes;
```

```
  return total
```

```
}
```

```
[tipo de retorno] [nome] ([params...]){
```

```
  códigos
```

```
  return*
```

```
}
```

*se declararmos um tipo para função precisamos indicar retorno



Arrows Functions

Arrows Functions simplificam a declaração de uma função.

Forma normal

```
int Exemplo(){
```

```
  x+y;
```

```
}
```

Arrow Function:

```
String Exemplo()=>x+y
```



Anonymous functions

```
void main()
{
  var list = ["pmerj", "pcerj", "cbmerj"];
  print("Anonymous function in Dart");
  list.forEach( (item) {
    print('${list.indexOf(item)} : $item');
  });
}
```



Anonymous functions