



Dart

Neste módulo, vamos introduzir a linguagem de programação Dart, que é utilizada no desenvolvimento de aplicativos Flutter. Discutiremos a história do Dart, suas características principais, a estrutura básica de um programa Dart e como configurar o ambiente de desenvolvimento para começar a programar com Dart. Esta aula fornece uma base sólida para entender a sintaxe e os conceitos fundamentais do Dart, preparando os alunos para as aulas subsequentes.

Introdução ao Dart

História e Características do Dart

1. História do Dart

- Desenvolvido pelo Google e lançado em 2011.
- Projetado para o desenvolvimento de aplicações web e móveis.
- Utilizado principalmente no framework Flutter para desenvolvimento de aplicativos multiplataforma.

2. Características do Dart

- Orientado a Objetos: Dart é uma linguagem orientada a objetos, onde tudo é um objeto.
- Compilação AOT e JIT: Suporta compilação Ahead-of-Time (AOT) para desempenho e Just-in-Time (JIT) para desenvolvimento rápido.

- Fortemente Tipada: Suporta tipagem estática com inferência de tipos.

Configurando o Ambiente de Desenvolvimento

1. Instalando o SDK do Dart

- Acesse dart.dev e siga as instruções de instalação para o seu sistema operacional.

2. Configurando o VS Code

- Instale o Visual Studio Code a partir de code.visualstudio.com.
- Adicione a extensão Dart no VS Code:
 - Vá em View > Extensions.
 - Pesquise por “Dart” e instale a extensão oficial.

Estrutura Básica de um Programa Dart

1. Hello World

Um exemplo simples de um programa Dart que imprime “Hello, World!”.

```
void main() {  
  print('Hello, World!');  
}
```

Explicação: A função main é o ponto de entrada do programa. O comando print exibe a mensagem no console.

2. Variáveis e Tipos de Dados

Declaração e uso de variáveis em Dart.

```
void main() {  
    String name = 'John';  
  
    int age = 30;  
  
    double height = 5.9;  
  
    bool isStudent = true;  
  
    print('Name: $name');  
  
    print('Age: $age');  
  
    print('Height: $height');  
  
    print('Is Student: $isStudent');  
  
}
```

Explicação: Dart suporta vários tipos de dados como String, int, double e bool. A interpolação de strings é usada para incluir variáveis dentro de uma string.

Nesta aula, vamos explorar funções e métodos em Dart. Funções são blocos de código que realizam tarefas específicas e podem ser reutilizados. Veremos como declarar e utilizar funções, passar argumentos, retornar valores e entender a diferença entre funções e métodos. Também discutiremos as funções anônimas e os métodos das classes.

Funções e Métodos

Declarando e Chamando Funções

1. Função Simples

```
void sayHello() {  
    print('Hello!');  
  
}
```

```
void main() {  
  
    sayHello();  
  
}
```

Explicação: A função sayHello não recebe parâmetros e apenas imprime uma mensagem.

2. Função com Parâmetros

```
void greet(String name) {  
    print('Hello, $name!');  
  
}
```

```
void main() {  
  
    greet('Alice');  
  
}
```

Explicação: A função greet recebe um parâmetro name do tipo String e imprime uma mensagem personalizada.

Retornando Valores de Funções

1. Função com Retorno

```
int add(int a, int b) {  
    return a + b;  
  
}  
  
void main() {  
  
    int result = add(3, 5);  
  
    print('Result: $result');  
  
}
```

Explicação: A função add recebe dois inteiros como parâmetros, soma-os e retorna o resultado.

Funções Anônimas e Arrow Functions

1. Função Anônima

```
void main() {  
    var list = ['apples', 'bananas', 'oranges'];  
  
    list.forEach((item) {  
  
        print('Item: $item');  
  
    });  
  
}
```

Explicação: Uma função anônima é uma função sem nome que pode ser passada como parâmetro.

2. Arrow Function

```
int multiply(int a, int b) => a * b;  
  
void main() {  
  
    print('Multiply: ${multiply(4, 5)}');  
  
}
```

Explicação: A sintaxe de arrow function é uma maneira concisa de definir funções de uma linha.

Métodos das Classes

1. Métodos de Instância

```
class Person {  
    String name;  
  
    int age;  
  
    Person(this.name, this.age);  
  
    void introduce() {  
  
        print('Hello, my name is $name and I am $age years old!');  
  
    }  
  
}
```

```
void main() {
```

```
var person = Person('John', 25);  
  
person.introduce();  
  
}
```

Explicação: Métodos de instância são funções definidas dentro de uma classe e operam sobre instâncias dessa classe.

Nesta aula, vamos aprender sobre coleções em Dart, incluindo listas, conjuntos e mapas. Coleções são usadas para armazenar e manipular grupos de objetos. Veremos como criar e usar cada tipo de coleção, e explorar algumas operações comuns, como iteração e manipulação de elementos.

Coleções em Dart

Listas

1. Criando e Usando Listas

```
void main() {  
  List<String> fruits = ['Apple', 'Banana', 'Orange'];  
  
  print(fruits[0]); // Acessando elementos  
  
  fruits.add('Grape'); // Adicionando elementos  
  
  print(fruits);  
  
}
```

Explicação: Listas são coleções ordenadas que permitem duplicatas. Podem ser manipuladas usando vários métodos, como add.

Conjuntos

1. Criando e Usando Conjuntos

```
void main() {  
    Set<int> numbers = {1, 2, 3, 4};  
  
    numbers.add(4); // Tentando adicionar duplicata (não será adicionada)  
  
    print(numbers);  
  
}
```

Explicação: Conjuntos são coleções desordenadas que não permitem duplicatas.

Mapas

1. Criando e Usando Mapas

```
void main() {  
    Map<String, String> capitals = {  
  
        'USA': 'Washington, D.C',  
  
        'France': 'Paris',  
  
        'Japan': 'Tokyo'  
  
    };  
  
    print(capitals['France']); // Acessando valores  
  
    capitals['Germany'] = 'Berlin'; // Adicionando pares chave-valor
```



```
print(capitals);  
  
}
```

Explicação: Mapas são coleções de pares chave-valor, onde cada chave é única.

Operações Comuns

1. Iteração em Coleções

```
void main() {  
    List<String> fruits = ['Apple', 'Banana', 'Orange'];  
  
    for (var fruit in fruits) {  
  
        print(fruit);  
  
    }  
  
}
```

Explicação: Usando um loop for para iterar sobre elementos de uma lista.

2. Manipulação de Elementos

```
void main() {  
    List<int> numbers = [1, 2, 3, 4, 5];  
  
    numbers.removeAt(2); // Remover o terceiro elemento  
  
    print(numbers);  
  
}
```

Explicação: Manipulação de elementos de uma lista usando métodos como `removeAt`.

Nesta aula, vamos nos aprofundar na programação orientada a objetos (OO) com Dart. Veremos como criar e usar classes, herança, interfaces e mixins. Também discutiremos a importância dos construtores e a implementação de métodos e propriedades nas classes.

Programação OO com Dart

Classes e Objetos

1. Definindo Classes e Criando Objetos

```
class Car {  
  String model;  
  
  int year;  
  
  Car(this.model, this.year);  
  
  void displayInfo() {  
  
    print('Model: $model, Year: $year');  
  
  }  
  
}  
  
void main() {  
  
  var car = Car('Toyota', 2020);  
  
  car.displayInfo();  
  
}
```

Explicação: Definindo uma classe Car com um construtor e um método para exibir informações.

Herança

1. Implementando Herança

```
class Animal {  
  void eat() {  
  
    print('Eating...');  
  
  }  
  
}
```

```
}  
  
}  
  
class Dog extends Animal {  
  
    void bark() {  
  
        print('Barking...');  
  
    }  
  
}  
  
void main() {  
  
    var dog = Dog();  
  
    dog.eat();  
  
    dog.bark();  
  
}
```

Explicação: A classe Dog herda métodos da classe Animal e adiciona seus próprios métodos.

Interfaces e Mixins

1. Implementando Interfaces

```
abstract class Shape {  
  
    void draw();  
  
}
```

```
class Circle implements Shape {
```

```
    @override
```

```
    void draw() {
```

```
        print('Drawing a circle');
```

```
    }
```

```
}
```

```
void main() {
```

```
    var circle = Circle();
```

```
    circle.draw();
```

```
}
```

Explicação: A classe Circle implementa a interface Shape e define o método draw.

2. Usando Mixins

```
mixin CanFly {
```

```
    void fly() {
```

```
        print('Flying...');
```

```
    }
```

```
class Bird with CanFly {
```

```
void main() {
```

```
var bird = Bird();  
  
bird.fly();  
  
}
```

Explicação: O mixin CanFly adiciona funcionalidade de voo à classe Bird.

Construtores

1. Construtores Nomeados

```
class Person {  
  String name;  
  
  int age;  
  
  Person(this.name, this.age);  
  
  Person.named(String name) {  
  
    this.name = name;  
  
    this.age = 0; // Default age  
  
  }  
  
}  
  
void main() {  
  
  var person1 = Person('Alice', 30);  
  
  var person2 = Person.named('Bob');  
  
  print('Person1: ${person1.name}, ${person1.age}');
```

```
print('Person2: ${person2.name}, ${person2.age}');  
  
}
```

Explicação: Construtores nomeados permitem a criação de diferentes formas de inicializar objetos da classe Person.

Este conteúdo fornece uma base sólida para iniciantes em Dart aprenderem a configurar seu ambiente de desenvolvimento, entender a estrutura básica de um programa Dart, trabalhar com funções e métodos, manipular coleções e aplicar conceitos de programação orientada a objetos. Para mais detalhes, consulte a documentação oficial do Flutter: [Flutter Documentation](#).

Materiais Extras

Você pode realizar o download do arquivo contendo os materiais extras utilizados ao longo das aulas por meio do seguinte link: <https://drive.google.com/file/d/1mg7lqMl8Pt2zl0rHlsFS0Qew00YN-sEX/view?usp=sharing>.

Conteúdo Bônus

Confira a live “Conceitos fundamentais da linguagem Dart,” transmitida pelo canal Flutterando TV no YouTube, como parte da 25ª edição do Flutterando Meetup. Apresentada por John Kevid, desenvolvedor Flutter na F-team, essa live aborda os principais conceitos da linguagem Dart, fundamental para o desenvolvimento com Flutter. Através desse conteúdo, é possível entender melhor a linguagem que dá suporte ao Flutter, explorando suas características e vantagens. Essa live é ideal para quem deseja aprofundar-

se na linguagem e elevar seu nível como desenvolvedor, agregando conhecimento sobre uma ferramenta poderosa como o Dart.

Referências Bibliográficas

BOYLESTAD, R. L.; NASHELSKY, L. **Dispositivos Eletrônicos e Teoria de Circuitos**. 11. ed. Pearson, 2013.

DEITEL, P. J.; DEITEL, H. M. **Ajax, Rich Internet Applications e Desenvolvimento Web para Programadores**. Pearson, 2008.

DUARTE, W. **Delphi para Android e iOS: Desenvolvendo Aplicativos Móveis**. Brasport, 2015.

FELIX, R.; SILVA, E. L. da. **Arquitetura para Computação Móvel**. 2. ed. Pearson, 2019.

LEE, V.; SCHNEIDER, H.; SCHELL, R. **Aplicações Móveis: Arquitetura, Projeto e Desenvolvimento**. Pearson, 2005.

MARINHO, A. L.; CRUZ, J. L. da. **Desenvolvimento de Aplicações para Internet**. 2. ed. Pearson, 2019.

MOLETTA, A. **Você na Tela: Criação Audiovisual para a Internet**. Summus, 2019.

SILVA, D. (Org.) **Desenvolvimento para dispositivos móveis**. Pearson, 2017.

Ir para exercício