VOLTAR

A linguagem DART, declaração de variáveis, desvios condicionais e funções

Aprender o que é a linguagem Dart, sua inportância para o desenvolvimento mobile híbrido e conhecer sua principal estrutura, ou seja, aprender como são declaradas as variáveis, funções, desvios condicionais e escopo de variáveis.

NESTE TÓPICO

Breve histórico da Linguagem

DART é uma linguagem de programação apresentada em 2011 por pesquisadores do Google com o intuito de substituir o Javascript, entretanto sua evolução permitiu, em 2013, integra-la com uma tecnologia de desenvolvimento Mobile Híbrida (conhecida com Flutter), em sua versão 2.0. O DART é uma linguagem de script com foco em desenvolvimento Web (LARS e BRACHA, 2011).

O desenvolvimento de aplicações móveis pode, atualmente, ser feito de duas maneiras: Com linguagens nativas (Java, Kotlin, Swift etc.), onde há uma linguagem específica para cada plataforma móvel existente (Android e iOS) ou de forma híbrida, onde um único código-fonte é capaz de gerar aplicações para mais de uma plataforma. O DART é uma linguagem para plataformas híbridas, ou seja, seu código-fonte pode ser executado em Android, iOS e navegadores (browsers) (OKEDIRAN, *et al*. 2014).

Atualmente, para desenvolvimento híbrido existem muitas tecnologias e linguagens, como Ionic, React Native, Flutter etc. O Dart, com o Framework (pacote de desenvolvimento) Flutter é uma das linguagens em rápida ascensão que tem ganhado muita notoriedade no mercado de aplicações móveis híbridas e tem tornado o desenvolvimento mobile cada vez mais

fácil, pois é uma linguagem de desenvolvimento e aprendizado muito ágil, devido à sus sintaxe simples e muito parecida com o que cada grande linguagem de programação traz de vantagem.

Declaração de variáveis

Para realizar os devidos testes e aprender a sintaxe da linguagem sem a necessidade de instalar nenhum software de desenvolvimento no momento, é possível utilizar um editor e compilador online, disponível em https://dartpad.dev, que é o mesmo que este material utilizará no momento.

As variáveis na programação servem para armazenar valores que podem ser lidos (recuperados) sempre que necessário. Por exemplo, para se calcular a área de um terreno retangular, é preciso criar variáveis que armazenam o tamanho da largura, o tamanho do comprimento e a própria variável que receberá o resultado do cálculo da área, para poder exibir ao usuário o valor final.

Na programação toda variável deve possuir um nome, normalmente do que ela representa no contexto programado e, normalmente, que tipo de informação ela guarda. Por exemplo as variáveis do tipo "String" permitem armazenar textos, do tipo "int" permitem armazenar valores inteiros, "boolea" permitem valores binários (verdadeiro ou falso) etc.

O trecho de código abaixo mostra a sintaxe de um "Olá mundo" na linguagem Dart. Repare como os comentários são colocados na linguagem, tanto em linha quando em blocos.

Repare, também, que o método "main" (principal) é o primeiro a ser executado pelo interpretador. Toda aplicação em Dart começará por este método.

```
1.  //Isso é um comentário
2.
3.  /*
4.   * Isso é um
5.   * bloco de
6.   * comentários(s)
7.   * */
8.
9.  void main() { //método principal (main):
10.     print('Olá Uninove!'); //imprimindo no console
11. }
```

As variáveis em DART podem ser declaradas com tipo de dados ou com o tipo "dynamic" que permite qualquer tipo de valor e ela assume o tipo de sua última atribuição. Veja, o trecho de código abaixo que mostra um exemplo de atribuições de variáveis de vários tipos, incluindo o dynamic, suas respectivas impressões e concatenação de textos com interpolação. Veja linha a linha com seus respectivos comentários, acima de cada uma das declarações. Os comentários são muito importantes para documentar o código e, neste caso, para comentarmos cada uma das linhas do programa sendo executado, para melhor entendimento daquela linha, por isso serão muito utilizados nessa disciplina.

```
void main() {

  //String -> tipo texto
  String nomeCurso = "Desenvolvimento Móvel";

  //int -> tipo números reais, inteiros
  int qtdAlunos = 100;

  //double -> tipo numero decimal, de ponto flutuante
  double mediaGeral = 9.5;

  //bool -> Tipo binário, verdadeiro ou falso
  bool temVaga = true;

  print(nomeCurso); //imprime o nome do nome do curso

  //impressão de um texto concatenado com uma variável:
  print("A Uninove oferece à você o curso de " + nomeCurso);

  //interpolação de uma variável com o texto (muito usado):
  print("O curso $nomeCurso tem $qtdAlunos alunos! :D");
  print("Os discentes tiram a nota $mediaGeral em média!");

  //impressão de variáveis booleanas:
  print("Este curso tem vagas? $temVaga");

  //o tipo dynamic permite qualquer valor.
  //ele assume o valor do último tipo:
  dynamic variavelDinamica = 1;
  print(variavelDinamica);
  variavelDinamica = "teste";
  print(variavelDinamica);
}
```

O resultado da execução do trecho de código anterior pode ser visto na figura baixo, executado pelo próprio DartPad, online.

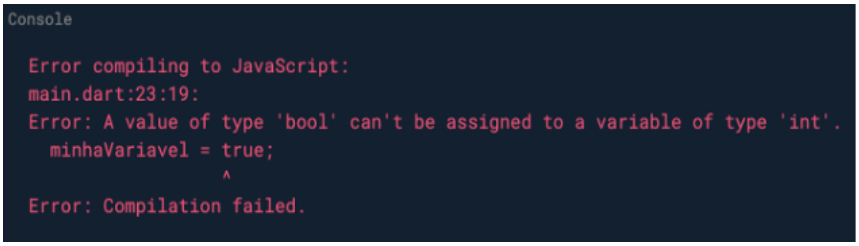
O resultado da execução do trecho de código anterior pode ser visto na figura baixo, executado pelo próprio DartPad, online.

```
Console

  Desenvolvimento Móvel
  A Uninove oferece à você o curso de Desenvolvimento Móvel
  O curso Desenvolvimento Móvel tem 100 alunos! :D
  Os discentes tiram a nota 9.5 em média!
  Este curso tem vagas? true
  1
  teste
```

O Dart implementa também o tipo "var" que é parecido com o dynamic, ou seja, a variável assumirá o valor de sua primeira atribuição. A diferença é que o dynamic poderá receber outros tipos de valores durante a execução do código, enquanto o "var" não, ou seja, ela assumirá o tipo da primeira atribuição e não poderá receber outros tipos em seguida. O trecho de código abaixo mostra isso na prática e simula o erro.

```dart
void main() {
  //trabalhando com dynamic

  //declarando uma variável Dynamic e atribuindo um valor inteiro:
  dynamic variavelDynamic = 10;

  print(variavelDynamic);

  //novo valor, de novo tipo:
  variavelDynamic = "Olá uninove! :D";
  //continua funcionando
  //agora essa variável é do tipo "String"
  print(variavelDynamic);

  print("\n"); //quebra uma linha

  //-----agora com var------
  var minhaVariavel = 33;
  print(minhaVariavel);

  //tentando trocar o tipo da variável "var":
  minhaVariavel = true; //<- dará erro nessa linha! ******
}
```

E como você pode observar, há um erro na linha onde a variável tipo "var" tenta receber um valor diferente de seu tipo original. A imagem abaixo mostra que este código possui um erro e explica-o, dizendo que "o valor tipo "bool" não pode ser atribuído a uma variável do tipo int" e mostra onde está o erro no código.



É muito importante observar os erros de compilação cuidadosamente, pois estes ajudam muito a encontrar e sanar quaisquer problemas na codificação.

Desvios condicionais

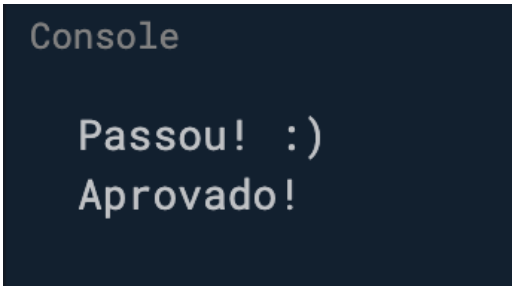
Agora que já sabemos com trabalhar com variáveis podemos aprender como trabalhar com desvios condicionais no DART. Um desvio condicional representa o desvio (alteração) da execução do código, dada uma condição.

Os desvios condicionais estão associados as cláusulas "se, então" da programação e, no DART, há essencialmente duas formas de trabalhar com isso: O famoso e muito utilizado "if, else" e com operadores ternários.

O trecho de código abaixo mostra o cálculo de uma média entre duas notas arbitrárias e verifica, inicialmente, com um "if, else" (se, então), se o aluno foi aprovado. Em seguida é verificada a variável "aprovado" em uma condição ternária. Analise cuidadosamente o código abaixo, com seus respectivos comentários explicativos.

```
void main() {
  //variáveis
  bool aprovado;
  double nota_av1 = 10.0;
  double nota_av2 = 9.5;

  //cálculo da média
  double media = (nota_av1 + nota_av2) / 2;

  //se média é menor que 6:
  if (media < 6) {
    print("Infelizmente não passou! :(");
    aprovado = false;
  } else if (media != 10) {
    //se a média é diferente de dez
    print("Passou! :)");
    aprovado = true;
  } else {
    //se a média é igul a dez
    print("Top! :D");
    aprovado = true;
  }

  //if ternário em DART.
  //repare que a variável "info" rececе o retorno da condição
  //Sintaxe do if ternário:
  // <condição> ? [VALOR SE VERDADEIROS] : [VALOR SE FALSO]
  String info = aprovado ? "Aprovado!" : "Reprovado";

  //imprime a variável "info":
  print(info);
}
```

E o resultado da execução do trecho de código anterior pode ser visto na imagem abaixo:



O operador ternário é muito utilizado e é muito importante praticar bastante para entender bem sua sintaxe e, por isso, é lançado para você um pequeno desafio, para praticar: Tente criar uma pequena calculadora de índice de massa corpórea (IMC), utilizando a cláusula ternária para analisar a classificação da pessoa.

Uma outra forma de utilizar o desvio condicional em DART é aplicando o operador "SWITCH", (chaveado), onde as conduções estão dentro de uma série de possibilidades.

```
void main() {

  String tipoDeRoupa = "Camisetas";

  //seleção de um programa de máquina de lavar
  switch(tipoDeRoupa){
    case "Jeans":
      print("Lavagem pesada");
      break;
    case "Seda":
      print("Lavagem delicada ");
      break;
    case "Pano de chão":
      print("Lavagem pesada com alvejante");
      break;
    case "Camisetas":
      print("Lavagem diária");
      break;
    default:
      print("Verifique a etuqueta");
      break;
  }
}
```

Repare no código acima que há um bloco "deafult" (padrão) dentro do "switch". Este bloco será executado sempre se a escolha não estiver em nenhum outro tratamento, mas lembre-se que ao final de cada bloco é preciso incluir o "break" (quebra), para que a execução saia do bloco do switch, ou seja, não caia na cláusula default, sempre.

Funções

Trabalhar com funções em qualquer linguagem de programação é primordial, pois elas podem ser invocadas em qualquer ponto do código, sempre que precisarmos e evitamos, portanto, a necessidade de reescrever um trecho de código. As funções são blocos de código que podem ser invocados para realizar uma determinada ação.

As funções em DART são muito usadas quando passamos a integrar o Flutter no desenvolvimento e, por isso, este é um importante assunto para o conteúdo.

Veja o vídeo abaixo para entender como as funções, em Dart, funcionam.

O trecho de código do vídeo anterior por ser visto em:

```
void main(){
  dizNome("Uninove");
  calculaSoma(10.5, 15.20);

  double mult = calculaMultiplicacao(3.0, 5.0);
  print(mult);
}

void calculaSoma(double a, double b){
  double soma = a + b;
  print(soma);
}

double calculaMultiplicacao(double a, double b){
  double mult = a * b;
  return mult;
}

String dizNome(String nome){
  return "Olá $nome";
}
```

Mas existe uma peculiaridade em linguagens como o Dart (linguagens de script) muito interessantes sobre os parâmetros de entrada, que utilizamos muito. São os parâmetros opcionais.

O vídeo abaixo mostra como isso funciona e suas respectivas possibilidades. Lembre-se que é muito importante entender bem este conceito pois, quando começarmos a criar nossos primeiros aplicativos, usaremos muito (muito mesmo) este recurso.

Parâmetros opcionais na linguagem Dart

O código abaixo reproduz o código exemplificado no vídeo:

```
1.  void main() {
2.    criaBotao("Calcular", cor: "Azul", altura: 8.0, largura: 20.0);
3.  }
4.
5.  void criaBotao(String texto, {String cor, double largura, double altura}){
6.    print(texto);
7.    print(cor ?? "Verde");
8.    print(largura ?? 10.0);
9.    print(altura ?? 5.0);
10. }
```

Entretanto, existe uma outra propriedade muito interessante do Dart e bastante útil, também, que é não apenas passar um parâmetro de forma opcional, mas como passar uma função inteira como parâmetro. O vídeo abaixo mostra como isso pode ser feito, inclusive com funções anônimas.

Passando funções como parâmetros em Dart

O código abaixo mostra o código do vídeo anterior com alguns comentários para entendermos melhor o código-fonte.

```
void main() {
  criaBotao("Calcular", acaoClick, cor: "Azul", altura: 8.0, largura: 20.0);

  print("------");

  //criando uma função anônima como parâmetro
  criaBotao("Salvar", () {
    print("Essa é a ação do botão salvar");
  }, cor: "Roxo");

  print("------");

  //chamando a função passando uma função sem ação:
  criaBotao("Botão X", (){}, cor: "Ciano", largura: 10.0, altura: 3.0);
}

void acaoClick() {
  print("Essa é a ação de click!");
}

void criaBotao(String texto, Function click,
    {String cor, double largura, double altura}) {
  print(texto);
  print(cor ?? "Verde");
  print(largura ?? 10.0);
  print(altura ?? 5.0);
  click();
}
```

Apenas para fecharmos este assunto, temos que entender bem, também o conceito de escopo de variáveis, ou seja, entender que as variáveis declaradas dentro dos métodos são exclusivas dos métodos e as declaradas fora do método são globais naquele contexto (arquivo, por exemplo). O vídeo abaixo ilustra melhor essa ideia:

Não deixe de treinar bastante estes vídeos para entender bem a essa introdução à linguagem Dart, que é a base do desenvolvimento em Flutter. O Dart é uma linguagem muito simples e gostosa de trabalhar.

Quiz

Exercício Final

A linguagem DART, declaração de variáveis, desvios condicionais e funções

INICIAR

Referências

BRACH, Gilard, LARS, Bak. Dart: a new programming language for structured web programming. **GOTO Conference**, 10 out. 2011. Disponível em: https://gotocon.com/dl/goto-aarhus-2011/slides/GiladBracha_and_LarsBak_OpeningKeynoteDartANewProgrammingLanguageForStructuredWebProgramming.pdf. Acesso em: 12 nov. 2020.

DART. **Dart documentation**. Site. Disponível em: https://dart.dev/. Acesso em: 12 nov. 2020.

OKEDIRAN, O. O. *et al*. Mobile operating systems and application development platforms: a survey. **Int. J. Advanced Networking and Applications**. v.6, n.1, p. 2195-2201, july-aug. 2014. Disponível em: https://www.ijana.in/download%206-1-9.php?file=V6I1-9.pdf. Acesso em: 12 nov. 2020.

WINDMILL, Eric. **Flutter in action.** Nova Iorque: Manning publications, 2020. *E-book*. Disponível em: https://learning.oreilly.com/library/view/flutter-in-action/9781617296147/, Acesso em: 12 nov. 2020.

SINHA, Sanjib . **Quick start guide to Dart programming:** create high performance applications for the web and mobile. Lompoc, CA, EUA: Apress, 2019. *E-book*. Disponível em: https://learning.oreilly.com/library/view/quick-start-guide/9781484255629/. Acesso em: 12 nov. 2020.

Avalie este tópico

☆☆☆

Índice

Biblioteca (https://www.uninove.br/conheca-a-uninove/biblioteca/sobre-a-biblioteca/apresentacao/)
Portal Uninove (http://www.uninove.br)
Mapa do Site

Ajuda?

PRÓXIMO
Orientação a Objetos com PADRT (https://ava.un idCurso=)